TUGboat

Volume 32, Number 3 / 2011
TUG 2011 Conference Proceedings

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group.

Memberships and Subscriptions
2011 dues for individual members are as follows:

- Ordinary members: $95.
- Students/Seniors: $65.

The discounted rate of $65 is also available to citizens of countries with modest economies, as detailed on our web site.

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site.

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is $100 per year, including air mail delivery.

Institutional Membership
Institutional membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group, as well as providing a discounted group rate and other benefits. For further information, see `http://tug.org/instmem.html` or contact the TUG office.

Addresses
TEX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Telephone
+1 503 223-9994

Fax
+1 206 203-3960

Web
`http://tug.org/`
`http://tug.org/TUGboat/`

Electronic Mail
(Internet)

General correspondence, membership, subscriptions:
`office@tug.org`

Submissions to *TUGboat*, letters to the Editor:
`TUGboat@tug.org`

Technical support for TEX users:
`support@tug.org`

Contact the Board of Directors:
`board@tug.org`

Have a suggestion? Problems not resolved?
The TUG Board wants to hear from you:
Please email `board@tug.org`.

[printing date: December 2011]

Printed in U.S.A.

2011 Conference Proceedings

TEX Users Group
Thirty-second Annual Meeting
River Valley Technologies
Trivandrum, Kerala, India
October 19–21, 2011

TUGboat

COMMUNICATIONS OF THE TEX USERS GROUP

TUGboat Editor Barbara Beeton
Proceedings Editor Karl Berry

Volume 32, Number 3 • 2011
Portland • Oregon • U.S.A.

TUG 2011: TeX in the eBook era

Trivandrum ▪ Kerala ▪ India

October 19–21, 2011

Sponsors

TeX Users Group ▪ River Valley Technologies ▪ DANTE e.V. ▪ Spoken Tutorials ▪ CSTUG
with assistance from individual contributors. *Thanks to all!*

Bursary committee

Steve Peter, chair ▪ Jana Chlebikova ▪ Bogusław Jackowski ▪ Alan Wetmore

Conference committee

Kaveh Bazargan ▪ Karl Berry ▪ Krishna GS ▪ Robin Laakso ▪ CV Radhadkrishnan
With thanks to Namboodiri of RVT for the drawings.

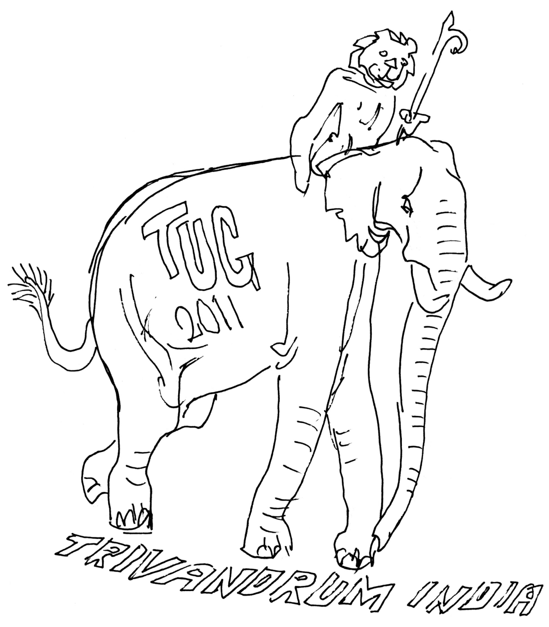
Participants

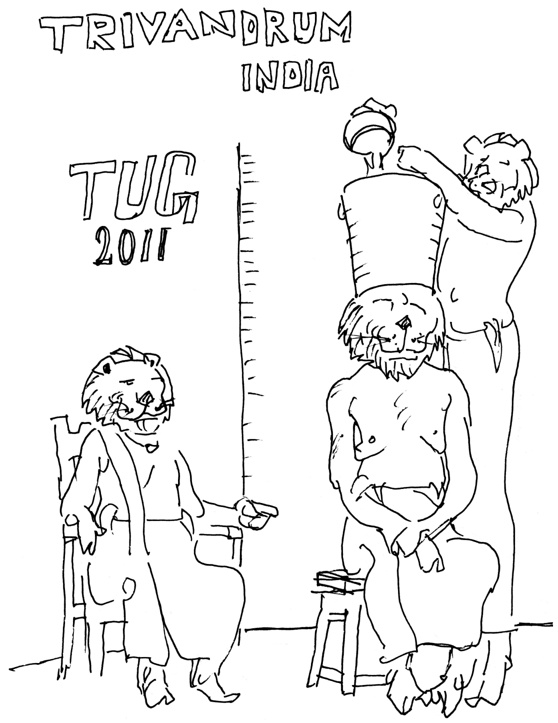
Pavneet Arora, Bolton, Canada
Kaveh Bazargan, River Valley Technologies
Barbara Beeton, American Mathematical Society
Dave Crossland, Wimborne, UK
Jean-luc Doumont, Principiae
L. Ganesh, TNQ Books and Journals
Steve Grathwohl, Duke University Press
Brian Housley, GCCS GmbH
Jean-Michel Hufflen, University of Franche-Comté
Anu Jexline, Mathematical Sciences Publishers
Manjusha Joshi, Pune, India
Jestin Joy, Rajagiri School of Engineering and Technology
Stefan Kottwitz, Germany
Reinhard Kotucha, Germany
Chandrashekhar Kumar, Bangalore
Suresh Kumar, SPi Global
Vinoth Kumar S, Scientific Publishing Services
Alagu Lakshmanan, Scientific Publishing Services
Saravanan M, SPi Global
S. Mahalakshmi, TNQ Books and Journals
Bob Margolis, Hampshire, UK
Frank Mittelbach, LaTeX3 Project
Rajiv Monsurate, Tamilnadu
Kannan Moudgalya, IIT Bombay
Ross Moore, Macquarie University
K. Murali, Transforma Pvt Ltd
Ilangovan N, Scientific Publishing Services
Josy Pullockara, Indian Institute of Science
CV Radhakrishnan, River Valley Technologies
Thomas Ratajczak, German Federal Armed Forces
Rishi, River Valley Technologies
Chris Rowley, LaTeX3 Project
Sukumar Sankar, TNQ Books and Journals
M. Sankaran, Transforma Private Ltd
Neeraj Saxena, Aptara
Johny Sebastian, Aptara
Karel Skoupý, Lingea & River Valley Technologies
Alistair Smith, Sunrise Setting Ltd
Petr Sojka, Masaryk University
Paulo Ney de Souza, BooksInBytes
Henrikas Stankus, VTeX
Vytautas Statulevicius, VTeX
Sehar Tahir, Aptara
B. Thiagarajan, TNQ Books and Journals
Sigitas Tolušis, VTeX
Suki Venkat, TNQ Books and Journals
Didier Verna, EPITA R & D Laboratory
Boris Veytsman, George Mason University
David Walden, E. Sandwich, MA
Alan Wetmore, US Army Research Laboratory
Dominik Wujastyk, Austria

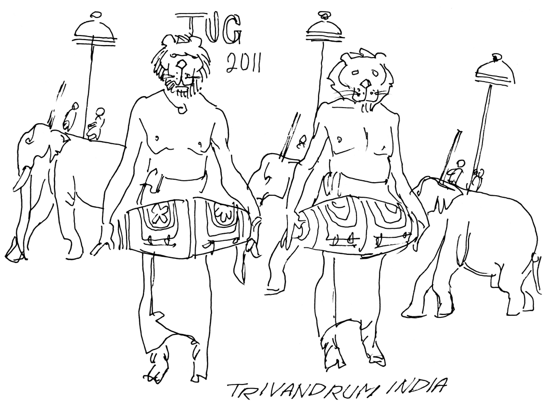
TUG 2011 — program and information

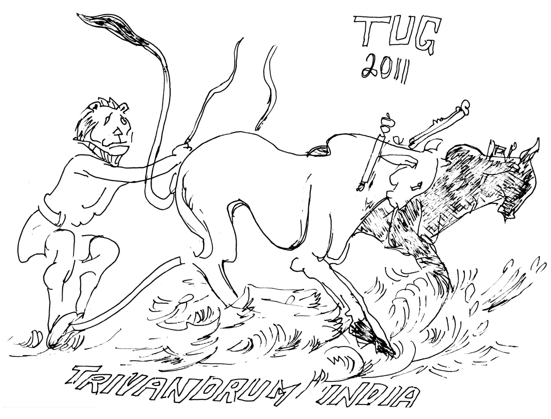
Wednesday October 19	8:30 am	*registration*	
	9:30 am	Barbara Beeton, TeX Users Group	*Welcome*
	9:35 am	Ross Moore, Macquarie University	*Further advances toward Tagged PDF for mathematics*
	10:10 am	Rishi, River Valley Technologies	*Creating magical PDF documents with pdfTeX*
	10:45 am	*break*	
	11:20 am	CV Radhakrishnan, River Valley Tech.	*TeX4ht — A Swiss army knife for TeX*
	12:15 pm	Karel Skoupý, Lingea	*Data structures in ε-TeX*
	12:30 pm	*lunch*	
	1:40 pm	Kaveh Bazargan, River Valley Tech.	*Why TeX is more relevant now than ever*
	2:15 pm	Alan Wetmore, US Army Research Laboratory	*e-Readers and LaTeX*
	2:40 pm	*break*	
	3:15 pm	Boris Veytsman and Michael Ware, George Mason Univ. and Brigham Young Univ.	*Ebooks and paper size: Output routine hacking made easy*
	3:50 pm	Rishi	*Automated generation of ePub from LaTeX*
	4:25 pm	q&a	
Thursday October 20	9:35 am	Jean-Michel Hufflen, University of Franche-Comté	*A comparative study of methods for bibliographies*
	10:10 am	Brian Housley, GCCS GmbH	*Making a package for flexible letter & page headings*
	10:45 am	*break*	
	11:20 am	Didier Verna, EPITA R & D Lab.	*Toward LaTeX coding standards*
	11:55 pm	Frank Mittelbach, LaTeX3 Project	*LaTeX3 architecture*
	12:30 pm	*lunch*	
	1:40 pm	Dave Crossland, Wimborne, UK	*Freeing fonts for fun and profit*
	2:15 pm	Boris Veytsman & Leyla Akhmadeeva, Bashkir State Medical University	*Towards evidence-based typography: Experiment design*
	2:40 pm	*break*	
	3:15 pm	Karel Skoupý	*Typesetting fancy multilingual phrase books with LuaTeX*
	3:50 pm	Dominik Wujastyk, Austria	*Typesetting Sanskrit in various alphabets: XƎLaTeX, TEC files, hyphenation, and even XML*
	4:25 pm	q&a	
Friday October 21	9:35 am	Pavneet Arora, Canada	*Typesetting with masonry*
	10:10 am	Jean-luc Doumont, Principiae	*Integrating TeX and PDF seamlessly in pdfTeX*
	10:45 am	*break*	
	11:20 am	Sukumar Sankar, S. Mahalakshmi and L. Ganesh, TNQ Books and Journals	*An XML model of CSS3 as an XLaTeX-TeXML-HTML5 stylesheet*
	11:45 am	S.K. Venkatesan, TNQ	*On the use of TeX as a general markup language for HTML5*
	12:05 pm	Kannan Moudgalya, IIT Bombay	*LaTeX training through spoken tutorials*
	12:30 pm	*lunch*	
	1:40 pm	Stefan Kottwitz, Germany	*Bringing together TeX users online: From Usenet to Web 2.0 and beyond*
	2:15 pm	Manjusha Joshi, India	*A dream of computing and LaTeXing together: A reality with SageTeX*
	2:40 pm	*break*	
	3:15 pm	Petr Sojka, Masaryk University	*Why TeX math search is more relevant now than ever*
	3:50 pm	Dominik Wujastyk, Austria	*My father's book: Typesetting and publishing a family memoir*
	≈ 4:30 pm	*end*	

(Drawings courtesy of Namboodiri of River Valley Technologies.)









TUG 2011 in India

Barbara Beeton

For the second time (the first was in 2002), the TUG annual conference was held in Kerala, India. Since I wasn't able to attend the first one, this was an exciting new experience.

The principal accommodations were in a hotel just off the beach near Trivandrum. The view toward the shore was quite enchanting, and the waves, which built up across the entire expanse of the Indian Ocean, made me wonder why there weren't any surfers — but that isn't the Indian way.

The conference itself took place in the facilities of River Valley Technologies, an interesting coach ride inland. (On the way, one unexpected sight was the iconic "dangerous bend" sign "in the wild"! Thanks to Reinhard Kotucha, a photo was obtained and forwarded to Don Knuth as a souvenir.) The buildings and grounds of River Valley have been designed and constructed to be minimally dependent on external utilities while providing an inviting and accessible workplace. The garden provides fresh fruit and vegetables for the canteen, and water is collected from rain, stored, used and recycled. "Natural" air conditioning is provided by broad roof overhangs and open windows for air circulation; only the conference room, directly under the roof, requires powered air conditioning. The River Valley web site (`http://river-valley.com/new-office-campus`) has photos and a good description of the facilities, and it is well worth viewing.

A TUG conference isn't just location. The real attraction is seeing old friends, making new ones, and learning what has been going on in the TeX world. This year's conference wasn't a large one, but it was interesting. The theme, eBooks, is a "hot" topic these days, and one of my goals was to find out what tools are available right now that can be applied to the Math Society's books to make them available in electronic form. (Interest has even been expressed in the possibility of putting this material onto small devices such as cell phones and PDAs; my conclusion is that this isn't going to happen very soon — not even if a magnifying glass is supplied à la the compact OED.) The consensus seems to be that things are looking up with respect to moderate-sized e-readers, but the situation is still far from ideal.

Several talks captured my attention in particular. I don't mean to slight any of the others; in fact, all the talks had something important to say, and we are fortunate that they are available on video: `http://river-valley.tv/conferences/tug-2011`.

Alan Wetmore's demonstration of several actual e-book devices was a real eye-opener. While text looked quite good, and this paper-and-ink devotee can even see an attraction for some situations, the math still needs work. Watching a display equation pour down the side of the screen like honey from a pitcher must be seen to be believed.

The problem of communicating published math to readers with impaired vision is a thorny one. Ross Moore has been working for several years on techniques for converting LaTeX files to tagged PDF. Adobe has apparently said that they will accept MathML, and now that PDF is an international standard, the goal should be to hold them to that implied promise. More work is needed, but this appears to be an achievable goal.

For decades, I've used the "light box" technique to determine whether two allegedly identical pages really are the same. Rishi described what I would call a "virtual light box" that overlays two PDF pages so that any small variation glares out from the screen. Using this technique, River Valley has cut its need for printed proof from four sheets per page to one sheet per four pages. (You do the arithmetic.) It really is magic, and I want a copy; it seems to be available now from the `river-valley.com` site.

Dominik Wujastyk's talk about designing, typesetting and publishing his father's memoirs was a touching lesson in what can be done by someone with real interest in a project, but very little funding. The result, passed around through the audience, is a fine example that aspires to Don Knuth's exhortation to create beautiful books.

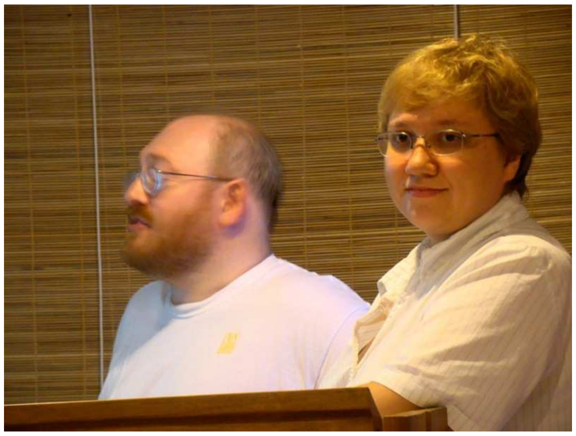
More contemporary in flavor are the multilingual phrase books created by Karel Skoupý. Very colorful, they are well designed, well organized, and I would really enjoy using them if I understood the languages involved.

This issue is a composite — talks for which papers were delivered along with other, regular articles. We've done this before, and expect to do it again when conditions warrant, so that delays are kept to a minimum. A final note: Although Axel Kielhorn did not attend the meeting, his article was so close to the topic that it seemed appropriate to include it among the talks.

Thanks to Kaveh, Radhakrishnan and Krishna for their kind invitation to hold the conference at River Valley, and to everyone else involved in making everything work so well.

◇ Barbara Beeton
`http://tug.org/TUGboat`
`tugboat (at) tug dot org`

Photos courtesy of: Vytas Statulevicius, Jean-luc Doumont, Sehar Tahir, Frank Mittelbach and Vidhya GS.


Boris Veytsman, Leyla Akhmadeeva.

Suresh Kumar, Saravanan M, Rajiv Monsurate.

Chandrasekhar Kumar, Vidhya, Stefan Kottwitz, Manjusha Joshi, Leyla Akhmadeeva, Sehar Tahir, Barbara Beeton, Kaveh Bazargan, Johny Sebastian.

Radhakrishnan CV, Kaveh Bazargan, Neeraj Saxena.

Lunch in the River Valley canteen.

Krishna.

Vidhya, Brian Housley.

Ross Moore.

Barbara Beeton, Ross Moore, Paulo Ney de Souza.

Jean-luc Doumont.

Sigitas Tolusis and wife, Renata, on an elephant ride.

Petr Sojka.

Post-conference back-water tour.

Jean-Michel Hufflen.

Vidhya and Radhakrishnan.

TEX online communities — discussion and content

Stefan Kottwitz

Abstract

On the Internet there are various platforms where TEX users meet for discussion. In this article, such systems will be compared with a particular focus on usability and content development.

1 Introduction

It all began in the 1980s with mailing lists such as `texhax`,[1] and Usenet. Around 1990, the Usenet group `comp.text.tex`[1] emerged, and continues today to be a place where TEX hackers gather.

On the continuously developing Internet, TEX user groups created mailing lists, and built home pages and software archives. Web forums turned up and lowered the barrier for beginners and occasional TEX users to get support.

Today, TEX's friends can also follow blogs and news feeds, and take part in vibrant question and answer sites.

These various systems offer different features, which make some particularly useful for discussion and others useful for information look-up and content creation.

2 Classic discussion systems

2.1 Mailing lists

Subscribers to mailing lists discuss a certain topic via email. The topic can be broad, such as TEX in general, or very specific, such as a particular LATEX 2ε package. The list's server receives emails from subscribers and reflects them to all other subscribers.

Mailing lists have the advantage that they can be used on every device with a mail client, so are accessible on tablets and smartphones, and even offline, just going online during receiving and sending.

However, there are caveats:

- Following a general (LA)TEX list can be difficult because of high traffic.
- Subscribing to quite a few specialized lists can be overwhelming.
- If a user doesn't know yet which package might solve his problem, it may be hard to find the right list.

Focused mailing lists are great for organizations, developers, and authors, but less so for a casual user.

A well known example of a mailing list for general TEX questions and discussion in English language is `texhax`.[2] It has been online since the 1980s, has hundreds of subscribers and offers a public archive.

About 50 further lists, most dealing with a specific topic, can be found on the TUG home page.[3]

There are further specialized TEX mailing lists hosted by various providers.

2.2 Usenet groups

Usenet is a discussion system on the Internet, distributed by thousands of servers world wide. It emerged around 1980. In Usenet, articles are logically organized in hierarchies of subjects and arranged in threads. It can be accessed via a dedicated newsreader client or can be accessed via web interfaces, such as Google Groups or mail gateways.

The first TEX group `comp.text.tex`[4] was established about 1990, and it is still active today with about 1000 posts each month. Its language of discussion is English, but there are further groups in other languages, including `de.comp.tex.tex`[5] in German since 1992, `fr.comp.text.tex`[6] in French since 1992 and `es.comp.lenguajes.tex` in Spanish since 1996 (although the latter is not used any more).

Usenet has some advantages — it is distributed on many thousands of servers, and is thus redundant which makes censoring hardly possible. Furthermore, it has been around many years and a lot of experienced users participate. However, though the Usenet as a whole is structured, the TEX group itself has no further structuring, except thread subjects.

There are feature-rich dedicated Usenet clients, although many people also use it via Google Groups. This brings us to a potential problem — some nice features depend on Google Groups:

- How could we access the `comp.text.tex` archive if Google stops providing it? Remember, Deja News stopped the original Usenet search service in 2001, before the archive was sold to Google who reopened it.
- How could we access it via the web if Google Groups disappears?

3 Web based communication

3.1 Blogs, feeds and aggregators

There are various blogs maintained by users, user groups and companies. They offer knowledge and news, though they can be hard to find and follow. Feed aggregators provide a solution for this problem.

[1] Reviewed by Jim Hefferon in "Which way to the forum?", *TUGboat* 32:2, 2011

[2] `http://lists.tug.org/texhax`

[3] `http://lists.tug.org`

[4] `http://groups.google.com/group/comp.text.tex`

[5] `http://groups.google.com/group/de.comp.text.tex`

[6] `http://groups.google.com/group/fr.comp.text.tex`

They offer convenient access by aggregating posts of dozens TeX blogs into one list, which can be read online or via a feed. Two are outstanding, `texample.net`[7] and `planet.dante.de`.[8] Both offer a chronological list of posts from most TeX blogs.

3.2 Web forums

A web forum is an HTML-based discussion forum on the Internet, usually hosted on one server, unlike Usenet. Forum posts are logically organized into categories and subcategories and arranged in threads, usually chronologically. The forum can natively be accessed via web browser on any Internet capable computer, thus also on tablets and smart phones. Posts can make use of markup such as HTML, BBCode or Markdown, and LaTeX syntax-highlighting is usually available. Web forums support file attachments and inline images, useful for displaying TeX and LaTeX output.

Web forums are usually moderated, and thus are spam-free and afford some measure of quality control.

`latex-community.org`[9] is a well frequented web forum[1] for TeX and LaTeX, covering all topics. It has been online since Jan 20, 2008. At this writing, it has 7673 registered users and 14,087 threads containing 52,762 posts are available for browsing and via the forum search feature. The forum is organized into 5 categories with 38 subforums.

In addition to LaTeX-specific web forums, there are also various LaTeX subforums on many technology and math/science discussion sites.

Another LaTeX forum is `golatex.de`,[10] though it is in German. An outstanding feature is its LaTeX wiki, which uses the GNU Free Documentation License.

A challenge — building a knowledge base

Besides communication — how can we improve the content of online TeX resources? This means reliable archiving, good searching and browsing access, quality measuring, elimination of redundancies, and cross-linking.

3.3 Q&A sites and advanced web applications

So-called Q&A web sites are specialized in strict question & answer format. They are intended both for experts and for general user support. Like web forums, such sites are hosted on a server or server farm. The complete archive is stored as a database enriched with extra information such as quality scores, content related tags and links to related information.

`tex.stackexchange.com`,[11] now also known as TeX.SX,[1] is a TeX-dedicated site on the network of Stack Exchange Q&A sites. These sites offer a very dynamic web interface with assisted editing, tooltips, good search and browsing features. The site's content is free under the CC BY-SA license;[12] regular database dumps are freely available for download on `clearbits.net`.[13]

TeX.SX has been publicly online since November 11, 2010. Today there are 7,300 registered users, more than 11,000 questions, and about 20,000 answers, and it is quickly growing.

In August 2011, Stack Exchange Inc. became an institutional member of the TeX Users Group, initiated by TeX.SX.

Compared to other systems, TeX.SX offers some outstanding features:

Tagging: Questions can be marked by one or several tags. This allows browsing by subject, filtering, feed subscribing, and more search features.

Voting: Users can vote posts up or down. So the best solution (or at least the most popular) will be displayed at the top, most easy to see.

Reputation system: Users earn reputation score if other users vote up their posts. This allows community moderation: the more reputation the more moderation features are available for the user.

Community edits: All posts can be edited by all users, either directly by users with high reputation score, or by edit suggestions which need to be confirmed. This improves quality: mistakes can be corrected and answers can be improved.

Duplicate control: When a user creates a question, possible duplicates are suggested. Users can flag existing duplicates. This leads to the best solution, with an automatic FAQ system.

Database exploring: The database dump can be browsed by SQL queries online.[14] This provides statistical features; complex queries can filter and connect content and attributes.

Open API: Programmers have developed applications for various special purposes, and for Android and iOS.

Meta site: There is a companion Q&A site with similar features, where users can discuss moderation, usage and any questions about the site

[7] `http://www.texample.net/community/`
[8] `http://planet.dante.de`
[9] `http://www.latex-community.org`
[10] `http://www.golatex.de`
[11] `http://tex.stackexchange.com`
[12] `http://creativecommons.org/licenses/by-sa/3.0/`
[13] `http://www.clearbits.net`
[14] `http://data.stackexchange.com`

		Mailing lists	Usenet	Web forums	Q&A
Usability	Reading, writing	✓	✓	✓	✓
	Markup, inline graphics			✓	✓
	Attachments	✓	✓	✓	✓
	Commenting, annotating				✓
	Deleting own posts			✓	✓
	Community deleting				✓
	Editing own posts			✓	✓
	Community editing				✓
Interfaces	Native web access			✓	✓
	Articles, blogs			✓	✓
	Tool-tips				✓
	Assisted editing				✓
	Feeds		✓	✓	✓
	Twitter posts				✓
	Chat				✓
	Statistics				✓
	Open API		✓		✓
Availability	Redundancy		✓		
	Archive on server	✓	✓	✓	✓
	Full public archive	✓			✓
Quality	Accepted solutions				✓
	Community voting				✓
	Duplicate elimination				✓
	Automatic FAQ extraction				✓
	Community edits				✓
Moderation	By moderators			✓	✓
	By the community				✓
	Mod election by community				✓
	Meta & moderation site				✓
Content access	Full text search	✓	✓	✓	✓
	Topic categories			✓	✓
	Quality sorting				✓
	Database queries				✓
Filtering by	Topics			✓	✓
	User-defined terms		✓		
	Consensus score				✓
	User score		✓		

Table 1: Feature comparison of online systems

and how it works. This keeps the focus of the main site on TEX-related content.

Chat site: A chat with features closely related to the main TEX site allows free discussion of more complex problems.

4 Comparing systems

Table 1 shows which features are available on which systems, for mailing lists, Usenet groups, web forums, and Q&A sites. It is a rough comparison based on the mentioned TEX examples for each platform. Some points are debatable though. For example, on the Usenet authors may cancel messages, though on the distributed network this is clearly not reliable, and today there are web gateways for non-web services such as Usenet and mailing lists.

Conclusion

For discussion, Usenet groups, mailing lists, and web forums are great. On Q&A sites, mixing discussion with content is undesirable, however there are separate discussion sites and chats as companions to the main site.

For content building and for developing TEX knowledge bases, dedicated sites with a proper free license are recommended.

◇ Stefan Kottwitz
Hamburg, Germany
`stefan (at) texblog dot net`
`http://texblog.net`

LaTeX training through spoken tutorials

Kannan M. Moudgalya

Abstract

Spoken Tutorials, a combination of screencast and voice over, are meant for self-learning of free and open source software (FOSS) systems. The pedagogy involved in creating spoken tutorials for LaTeX is explained. A checklist, instructions for conductors and activity-based instructions for learners, along with the spoken tutorial are what is needed to conduct the two hour SELF FOSS Study Workshops. As one finds out during the workshop how to learn from these tutorials, one can complete the learning at home if two hours are insufficient.

This method of learning has been shown to be effective. As it allows the conducting of these workshops without the domain experts, this methodology is scalable: we expect to conduct 500 workshops in a period of six months. The students are trained free of cost. The honorarium expenses for the conductors of these workshops work out to Rs. 25 per student per software package.

1 Introduction

A spoken tutorial is a an audio-video tutorial that explains an activity performed on the computer. An expert explains the working of the software by demonstrating it on the screen, along with a running commentary. Screencast software makes a movie of the entire activity, both the screen and the spoken part. This movie is the spoken tutorial. It is of ten minutes duration. One can reproduce the commands shown in the tutorial side by side and thus use it as an effective instructional tool.

We have been using this methodology to create a series of tutorials in open source software families, such as LaTeX, Scilab, GNU/Linux, ORCA, Python, LibreOffice and PHP/MySQL. We have selected the duration of a typical spoken tutorial to be about ten minutes long. Although only a small topic can be covered in ten minutes, by stringing them together, one can come up with study plans that are capable of teaching advanced topics as well.

Our approach involves the creation of a script before creating the video. It is possible to translate the script into other languages and use them for dubbing, while screen shots continue to be in English. For example, see a tutorial with Tamil audio at [1]. This will help those who are weak in English, while not compromising on the employability.

Spoken tutorials can also be used to bridge the digital divide: topics such as buying train tickets online, locating low cost agricultural loans, and locating information on first aid and primary health care can be covered. One target audience for a spoken tutorial is a remote child, working alone at midnight without anyone to help her.

As the spoken tutorials are created for self-learning, it is possible to conduct workshops even without domain experts. This allows scaling up workshops using additional instructional methodologies.

The above mentioned points have been explained in [4]. The motivation behind this effort is available at [6]. An early work in this area is [3].

The initial part of this article focuses on creation of spoken tutorials on LaTeX and Xfig. The rest of this paper is devoted to the methodology of conducting workshops using these tutorials.

2 Creation of LaTeX Spoken Tutorials

We created the first few Spoken Tutorials in early 2007 to teach LaTeX to our students. We had just completed a textbook [2]. Had we not used LaTeX, it would have taken several more years to complete it. We wanted our students also to benefit from LaTeX. We also wanted to contribute something back to the community. We created the following tutorials:

1. What is compilation (9:17)
2. Letter writing (8:19)
3. Report writing (16:14)
4. Mathematical typesetting (24:46)
5. Equations (23:42)
6. Tables and figures (25:12)
7. Bibliographies (8:20)
8. Inside story of bibliographies (24:44)
9. LaTeX on Windows (27:20)
10. Updating MiKTeX on Windows (15:31)
11. Beamer (34:00)

The numbers inside the brackets indicate the time duration in minutes and seconds. Although we wanted the tutorials to be 10 minutes long, some of them turned out to be longer. In those days, we did not have a method to estimate the length of a spoken tutorial before it was made. The original tutorial on bibliography was longer than we wanted, so we renamed it as the "inside story" and re-did a shorter one lasting only 8:20.

In order to make LaTeX available to Windows users, we made tutorials 9 and 10 in the above list. As our group had not used LaTeX on Windows, we had to learn it first before making a tutorial. Finally, we created the spoken tutorial on Beamer. We now insist on LaTeX-created slides to accompany all new spoken tutorials.

Block diagrams are an important requirement for any scientific writing. For this purpose, we created: introduction to Xfig (12:38), feedback diagram in Xfig (12:02) and mathematics in Xfig (15:20). The last tutorial explains a procedure to embed mathematical formulae in figures, using Xfig and LaTeX.

There are a few reasons for not using a special-purpose LaTeX editor for Unix systems:

1. We wanted to teach LaTeX in the most pristine form, without getting bogged down by the editor's details.
2. There is a good amount of effort involved in learning any special-purpose LaTeX editor.
3. As we don't use one, it would have taken a considerable amount of our time and effort to explain the working of a specialised editor.
4. As Unix users are reasonably comfortable with the terminal, our approach would not create much difficulty.

The situation is quite different for a Windows user, who is generally not comfortable in working with the terminal. As most of our audience use Windows, we did not mind putting in some effort in learning a specialised LaTeX editor for Windows and creating tutorials 9 and 10 above.

All the spoken tutorials on LaTeX explain a three step process: 1) Creating a source file with an editor; 2) Compiling the source with `pdflatex`; 3) Viewing the resulting PDF file with a PDF browser.

The recorded area was divided to simultaneously show the editor, terminal and the PDF browser. These three were arranged in a non-overlapping manner in the early tutorials, such as letter writing, mathematics and equations. In the letter writing tutorial, a given source file was explained along with an illustration of the effects of changes in the commands. In the tutorial on Beamer presentations, however, additional commands and text were copied from another file and the effects demonstrated. A screen shot of this is shown in Fig. 1.

Showing three or four non-overlapping windows made the fonts small. This was addressed in the Xfig tutorial, where the overlapping requirement was done away with, so as to achieve large font sizes. A screen shot of this is shown in Fig. 2. In all of the above, only a small portion of the Desktop was recorded to keep the recording size small.

All the tutorials mentioned above are available at `http://spoken-tutorial.org`, along with all the required source, style, `bst` and PDF files.

Spoken tutorial is based on demonstrations. We want a learner to reproduce whatever is shown in the tutorial. As it is *active learning*, it is effective. To ensure that this happens, we insist that 75% of every tutorial is devoted to demonstrations and not more than 25% is concerned with theory. These limits generally work well in all except perhaps the first tutorial in a series. To solve this problem, we recommend that the introductory tutorial be created after all other tutorials in the series are completed. Armed with these tutorials, one can also make the introductory tutorial consist of 75% demonstrations by playing these tutorials. This strategy has been followed in creating [6].

3 Workshops using Spoken Tutorials

Spoken tutorials are created for self-learning. These are also available for free download. Although one can learn from these tutorials without any external assistance, organised workshops are the most effective way if we want to train a large number of people in a short time.

Ours differ from conventional workshops in many ways: instructional material, conduct of the workshop, time duration of the workshop and the conductor of the workshop, to name a few. We will explain these in detail now.

3.1 Instructional material

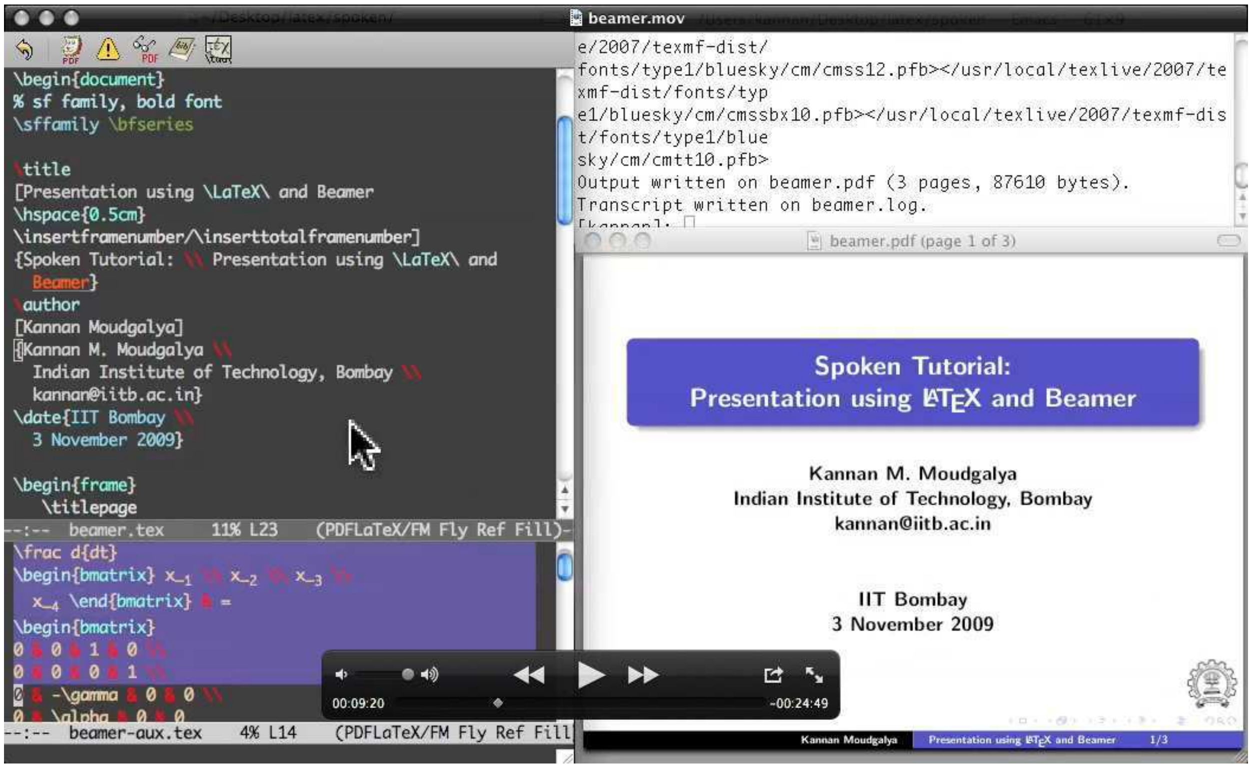
In order to conduct the workshops one needs the spoken tutorials and the associated files, as mentioned earlier. One can download these from our website free of cost. If required, we also provide them on CDs, once again free. In addition to these, we provide the following three documents:

A checklist that has to be completed the day before the workshop is conducted. This forces the organiser of the workshop to verify for every PC (a) that the spoken tutorials and resource files are copied; (b) that the software to be taught in the workshop is installed correctly; (c) whether the headphones are working; (d) whether the spoken tutorial can be seen and heard through the headphones. In case of online tests, the organiser should also check the working of the Internet. A copy of such a list is given in Fig. 3.

The second document is the instruction sheet for the conductor of the workshop. This important activity is covered in detail in the next section.

The rest of this section is devoted to the third document, which contains instructions to the learners. This instruction sheet is meant to be used with the spoken tutorials. We have reproduced the initial part of the instruction sheet meant for learning LaTeX on GNU/Linux system in Fig. 5.

These instruction sheets are for self-learners. In view of this, the initial part of the instructions are activity based, or equivalently contain verbs. For instance, the first few instructions have the following



beamer.mov

```
\begin{document}
% sf family, bold font
\sffamily \bfseries

\title
[Presentation using \LaTeX\ and Beamer
\hspace{0.5cm}
\insertframenumber/\inserttotalframenumber]
{Spoken Tutorial: \\ Presentation using \LaTeX\ and
  Beamer}
\author
[Kannan Moudgalya]
{Kannan M. Moudgalya \\
  Indian Institute of Technology, Bombay \\
  kannan@iitb.ac.in}
\date{IIT Bombay \\
  3 November 2009}

\begin{frame}
  \titlepage
--:--  beamer.tex      11% L23    (PDFLaTeX/FM Fly Ref Fill)-
```

```
\frac d{dt}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\
  x_4 \end{bmatrix} & =
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & -\gamma & 0 & 0 \\
0 & \alpha & 0 & 0
--:--  beamer-aux.tex    4% L14    (PDFLaTeX/FM Fly Ref Fill
```

```
e/2007/texmf-dist/
fonts/type1/bluesky/cm/cmss12.pfb></usr/local/texlive/2007/te
xmf-dist/fonts/typ
e1/bluesky/cm/cmssbx10.pfb></usr/local/texlive/2007/texmf-dis
t/fonts/type1/blue
sky/cm/cmtt10.pfb>
Output written on beamer.pdf (3 pages, 87610 bytes).
Transcript written on beamer.log.
[kannan]:
```

beamer.pdf (page 1 of 3)

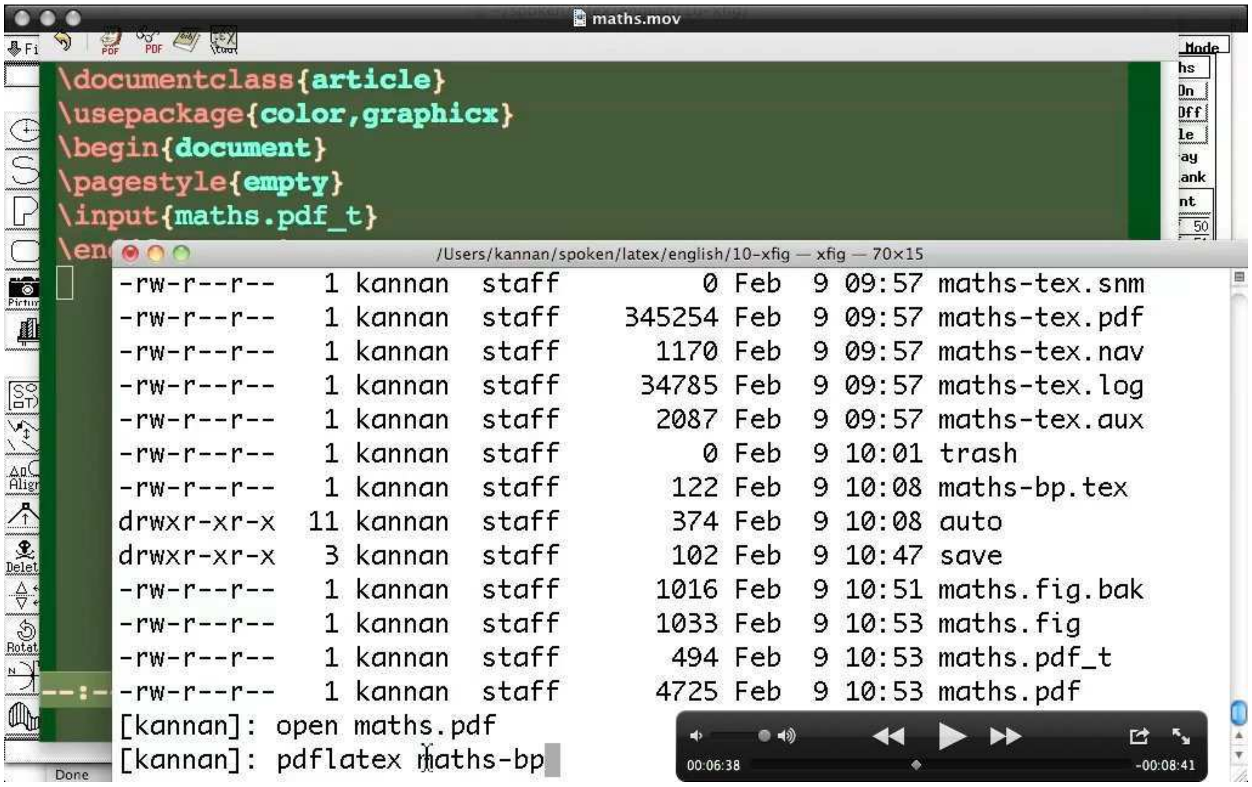
Spoken Tutorial:
Presentation using LaTeX and Beamer

Kannan M. Moudgalya
Indian Institute of Technology, Bombay
kannan@iitb.ac.in

IIT Bombay
3 November 2009

Kannan Moudgalya | Presentation using LaTeX and Beamer | 1/3

Figure 1: Two files, terminal, PDF browser shown in non-overlapping fashion in the Beamer Spoken Tutorial [5]



maths.mov

```
\documentclass{article}
\usepackage{color,graphicx}
\begin{document}
\pagestyle{empty}
\input{maths.pdf_t}
\en
```

/Users/kannan/spoken/latex/english/10-xfig — xfig — 70×15

```
-rw-r--r--   1 kannan  staff       0 Feb  9 09:57 maths-tex.snm
-rw-r--r--   1 kannan  staff  345254 Feb  9 09:57 maths-tex.pdf
-rw-r--r--   1 kannan  staff    1170 Feb  9 09:57 maths-tex.nav
-rw-r--r--   1 kannan  staff   34785 Feb  9 09:57 maths-tex.log
-rw-r--r--   1 kannan  staff    2087 Feb  9 09:57 maths-tex.aux
-rw-r--r--   1 kannan  staff       0 Feb  9 10:01 trash
-rw-r--r--   1 kannan  staff     122 Feb  9 10:08 maths-bp.tex
drwxr-xr-x  11 kannan  staff     374 Feb  9 10:08 auto
drwxr-xr-x   3 kannan  staff     102 Feb  9 10:47 save
-rw-r--r--   1 kannan  staff    1016 Feb  9 10:51 maths.fig.bak
-rw-r--r--   1 kannan  staff    1033 Feb  9 10:53 maths.fig
-rw-r--r--   1 kannan  staff     494 Feb  9 10:53 maths.pdf_t
-rw-r--r--   1 kannan  staff    4725 Feb  9 10:53 maths.pdf
[kannan]: open maths.pdf
[kannan]: pdflatex maths-bp
```

Figure 2: Overlapping of Emacs editor, terminal, and Xfig allows the use of large fonts [7]

PC No.	Is the PC booting?	Can log into PC? (if applicable)	These can be separated in 2 PCs, possibly in 2 rooms				
			PC for spoken tutorial				PC for test
			FOSS loaded?	Spoken Tut. copied?	Plays in VLC?	Audio works?	Internet works?
PC 1							
PC 2							
⋮							
PC 50							

Figure 3: A sample checklist to be completed before starting a SELF FOSS Study Workshop

Getting Started

1. 04:17: Perform the following calculations on the scilab command line:
   $$\text{phi} = \frac{\sqrt{5}+1}{2} \qquad \text{psi} = \frac{\sqrt{5}-1}{2}$$
   Find 1/phi and 1/psi
2. 6:06: Verify Euler's identity: Is $e^{\pi i}+1$ close to zero? Compare with $\cos(\pi)+i\cdot\sin(\pi)$

Figure 4: Assignments with timing for the *Getting Started* Spoken Tutorial on Scilab

verbs: click, locate, copy, open, gedit, right click, etc. Theory is introduced only slowly.

These instructions are based on the time when a certain activity has to be carried out. For example, the instructions numbered 7, 8 and 9 in Fig. 5, respectively, are supposed to be carried out at 1:57, 2:04 and 3:04 min.

There could be instructions to point out the difference in the activity. For example, the instruction at 3:04 says not to invoke the command `skim`, but to use `evince`. This methodology can also be used to correct minor mistakes, if any, in the spoken tutorial.

These instructions are detailed. For example, in the invocation of `evince` there is a space before the ampersand (`&`) symbol to help emphasize it. Normally, such detailed instructions are unnecessary for Unix users. But for Windows users who may not know the difference between a terminal and an editor and the concept of background jobs, the instructions need to be detailed.

Finally, there are assignments that need to be done at a specific time. This is clear from a sample assignment question of a Scilab spoken tutorial workshop, given in Fig. 4.

3.2 Conductor of the workshop

We do not need a domain expert to conduct these workshops. In a large country like India, one may have to conduct these workshops in thousands of places. It would be difficult to get domain experts to even visit such a large number of places, let alone *lecturing* in these workshops.

We insist that the conductor of the workshop *not* answer any domain-dependent questions. There are two reasons for this:

1. We are not sure about the capability of the conductors of these workshops. We definitely do not want them to give wrong answers.
2. Answering domain-dependent questions could take up a lot of time. As a result, a conductor may not be able to handle more than a few students. It would be impossible even for an expert to answer all the questions of twenty students, the recommended ratio in our methodology.

The students are supposed to follow the steps exactly as given in the instruction sheet and the spoken tutorial. Two types of difficulties can arise:

1. If the students have difficulty in following any instruction, the conductor of the workshop should point out the mistake made by the student. If the mistake is a serious one, the conductor could even ask the student to start the tutorial from scratch. As the tutorials are short, one will not have to spend a lot of time in repetition.
2. It is possible for a student to try out some changes of their own. They can do this so long as they do not encounter any problems. If they experience any difficulty, they are recommended to go to the next tutorial. The idea is that there are enough things (that work) to learn, before trying something of one's own.

To handle these two difficulties, the conductor of the workshop need not be a domain expert. As a corollary, a person who is trained to conduct a workshop on a topic (say, LaTeX), can conduct a workshop on another topic (say, Scilab) also.

The procedure indicated above suggests a definite set of things to learn during the workshop. The learner may not have the freedom to learn whatever

Spoken Tutorial Based LaTeX
Workshop on Linux
Spoken Tutorial Team
IIT Bombay
5 August 2011

First tutorial: `What is Compilation?`

These detailed instructions are intended mainly for Windows users, who may have to use GNU/Linux for learning LaTeX. GNU/Linux users will already know most of this.

1. Click the `Places` button in the top left hand corner and then click the `Home Folder`. The folder that opens is called your `home` folder.
2. Please locate the folder `LaTeX_Workshop` that is available on the desktop. The sub-folder `01-compilation` contains the following files that you need for this tutorial: `hello.tex` and `compiling.mov`.
3. Please copy `hello.tex` from this folder to your `home` folder.
4. Open the `terminal` using the command `Ctrl-Alt-t`, by pressing all these three keys simultaneously.
5. Open the file that you copied above into the editor using the command

   ```
   gedit hello.tex &
   ```

   Do not forget the symbol ampersand (`&`) at the end of the command, obtained by pressing `shift 7`. Please leave spaces exactly as given above.
6. Right click on `compiling.mov`, point the cursor on `Open With` and select `VLC Media Player`. Now listen to this spoken tutorial.
7. As shown in the video at 1:57min, compile from the terminal the file `hello.tex` using the command

   ```
   pdflatex hello.tex
   ```

   Note that `pdflatex` is *one* command. Please do not leave a space between `pdf` and `latex`.
8. Pause the video at 2:04min. You should now be able to give the command `pdflatex hello.tex` and get a file `hello.pdf`. If there is any difficulty in this step, please listen to the tutorial from 1:57min to 2:04min once again.
9. The video talks about a PDF viewer called `skim` at 3:04min.
   - Please do not attempt to use `skim` — it is *not* available on GNU/Linux.

   You have to use the PDF viewer `evince` instead. Give the following command from the `terminal` to open the PDF file:

   ```
   evince hello.pdf &
   ```

   Once again, do not forget the `&` symbol in the above command.

Figure 5: A sample of instructions for a SELF FOSS Study Workshop on LaTeX. To make it suitable for self-study, all initial instructions are activity based.

they want. But there are enough new things to learn in any case during the course of the workshop. This topic is explained further in the next section.

3.3 Duration of the workshop

We recommend a duration of two hours for these workshops. A workshop typically has about ten spoken tutorials. In a two hour period, one can learn four tutorials. More importantly, students will figure out how to learn from spoken tutorials. Thus, students can learn the remaining tutorials on their own.

So, why do we restrict the workshop to two hours? There are many reasons for this:

1. This will allow the same facility to be used for other workshops or for other people or both.
2. As our workshops are conducted free of cost, it is not clear how many students are really interested in the workshop. We do not want to host any uninterested students longer than absolutely necessary.
3. The students who are interested in appearing in the online exams have to necessarily complete the tutorials on their own. This improves the quality of learning and hence can indeed help provide better learning than conventional workshops.
4. In government supported training programmes, not only do the students pay nothing, but there are organisational expenses as well. For example, the conductors of the workshop have to be paid an honorarium for their time and effort. Such expenses are reduced by minimising the workshop duration. If any workshop is conducted for a longer period, there are also demands for a break and financial support for refreshments.

Not all students who undergo the two hour workshops have computers at home. So, if a college wants to offer their premises for learning, we have no objection. We tell them, however, that such sessions are not a part of our training programme.

A comment about the quality of learning in the two hour workshop is in order. In the previous section, the highly regimented procedure of the workshop has been discussed. Although no green field type of learning is possible, in no other method can one learn in two hours:

1. how to write letters using LaTeX,
2. how to write reports,
3. writing mathematics and equations, and
4. and introduction to presentations using Beamer.

We guarantee all of the above in a two hour workshop.

The two hour workshops discussed above are called *Spoken Tutorial based education and learning* *through Free FOSS Study Workshops* or SELF FOSS Study Workshops. The word *Free* in the above denotes *free of cost*, unlike the word free that comes in FOSS, which denotes freedom.

As mentioned earlier, the students who undergo training through these workshops do not pay anything. As of now, we actually *spend* about Rs. 500 for every twenty students as honorarium expenses for the organisers. Thus, the cost of training one student on one software package is Rs. 25. We are now planning to do away with this honorarium expense.

The SELF FOSS Study Workshops have been extremely effective and also popular. In a workshop on GNU/Linux conducted in an engineering college in Alwar, Rajasthan, by their own student volunteer, the average marks went up by 85% after the workshop, although the post workshop test was tougher than the pre-workshop test. As a matter of fact, every student passed the second test and received a certificate of completion.

4 Conclusion and future work

This article has presented an instructional methodology for conducting large number of FOSS workshops on software systems, such as LaTeX, Scilab, Python, PHP/MySQL and GNU/Linux. Using this scalable method, we expect to conduct about 500 workshops in a period of six months, with an average number of participants in each workshop of about 50.

Although the current workshops are organised only for college students, we hope to extend them to secondary schools as well. We will pursue this activity as soon as instructional material on LibreOffice is ready. Office seems to be the most important software for schools.

A more interesting question is whether it is possible to create instructional material to teach LaTeX for school students. It is likely that the methodology of conducting the workshop will work for schools also. What is not clear, however, is whether a special type of instructional material has to be created exclusively for schools.

We also desire to teach more advanced topics on LaTeX to the students who successfully complete the basic training explained in this work. This requires more spoken tutorials. We hope to get help from the TeX community for this purpose.

Finally, it will be useful to create more rigorous evaluation methods to check the efficacy of learning. At present, we use only multiple choice questions. It is not clear how to administer exams to check the LaTeXing capability of the large number of students who may take the exam. To preserve scalability, we need automatic evaluation methods.

References

[1] Tamil dubbing: T. Vasudevan and Priya. Report Writing in LaTeX, Seen on 4 Nov. 2011. Video available at `http://spoken-tutorial.org/How-to-write-a-Report-using-LaTeX-Tamil`.

[2] K. M. Moudgalya. *Digital Control*. John Wiley & Sons Ltd., Chichester, 2007.

[3] K. M. Moudgalya. Spoken tutorials. In *Technology for Education, T4E 2009*, pages 17–23, Bangalore, August 2009. IEEE.

[4] K. M. Moudgalya. Spoken Tutorial: A Collaborative and Scalable Education Technology. *CSI Communications*, 35(6):10–12, September 2011. Available at `http://spoken-tutorial.org/CSI.pdf`.

[5] K. M. Moudgalya. Presentation using LaTeX and Beamer, 3 November 2009. Video available at `http://spoken-tutorial.org/Latex_beamer_english`.

[6] K. M. Moudgalya. What is a Spoken Tutorial, 8 March 2011. Video available at `http://spoken-tutorial.org/What_is_a_Spoken_Tutorial`.

[7] K. M. Moudgalya. Embedding Maths in Xfig, 9 Feb. 2011. Video available at `http://spoken-tutorial.org/Xfig_Feedback_Diagram_with_Maths`.

◇ Kannan M. Moudgalya
Dept. of Chemical Engineering
IIT Bombay, Powai
Mumbai 400 076, India
`kannan (at) iitb dot ac dot in`
`http://spoken-tutorial.org`

e-Readers and LaTeX

Alan Wetmore

Abstract

2011 has seen many e-readers arrive on store shelves; a new generation of "touch screen" devices that include the Nook Simple Touch, Kobo eReader Touch, and a higher resolution iRiver Story HD. They all have the capability of loading user created content, so the question arises: how well can they support my legacy documents? The answer just might be, surprisingly well. After we understand the capabilities and some of the limitations we will explore how we can re-purpose older documents and prepare new LaTeX documents for use with these e-readers.

1 Introduction

There are lots of new e-reader machines this year. I've been trying out three: a Nook, a Kobo, and an iRiver. When I was invited to give a talk at the conference I decided to tell you about some of things I discovered as I explored using these machines.

I have been interested in e-readers for a while, but they all seemed to make it nearly impossible to bring your own documents to them. Recently there have been advances on more direct support of e-readers published in the latest *TUGboat* by William Cheswick [1] and Hans Hagen [2] as well as papers at this conference by Boris Veytsman and Rishi. Based on all of this work I am looking forward to seeing some truly powerful capabilities arriving soon.

How we can use them: collect our legacy documents; carry our class notes; read a manual while working away from the computer or in the field; expand our markets, ...

1.1 A quick tour of the machines

Common elements are:

Size Height and width similar to a medium sized paperback book, but quite thin.

Screen a touch screen for selecting and navigating. About 6×8 inches (150×200 mm); either 600×800 or 768×1024 pixels. In 1991 these were fair to good resolutions for a PC or low-end workstation. There were a lot of 1024×768 X terminals out there.

Bezels surrounding the screen, giving us a way to hold the reader without obscuring the text as well as a convenient place for the buttons that navigate the menus and page through the document.

Buttons at least a power switch, often page turning buttons, sometimes a keyboard.

Syncing usually through a USB cable to a PC, sometimes via wireless to their proprietary "Bookstore". Usually there is also a program for our computer that manages our purchases on the device but the memory of the e-reader can also appear as an external disk with folders where we can copy our own files.

Memory expansion usually done through a removable microSD card (SD in the iRiver). We can copy files onto the card when the e-Reader is tethered via USB and the microSD card shows up as another drive, or the card can be removed from the reader and loaded and modified using a standard card reader.

"Bookstores" are a universal feature, everyone wants to "sell" you books, magazines, and anything else they can think of. Some bookstores make it easier than others to get the books you create on the "shelf".

Speed is one of the main things that affects our experience; how "snappy" is the menu navigation, how long does it take to process our document and display it so we can start reading, and how quickly and smoothly can we "turn the pages"? All three seem acceptable to me.

Library organization All of the readers organize your "Library" and let you sort the display by author or title. In addition there is usually a search function that includes author, title, and keywords from the metadata.

Navigation The readers all make it easy to page through a book using swipes on the touchscreen and or dedicated page turning buttons. In addition they have additional capabilities for jumping to particular pages.

Document formats While all of the readers can deal with quite a few formats we will concentrate on just two — ePub and PDF. ePub because it is usually the best supported format; it is what everyone but the Kindle sells in their store. PDF because we have large legacy document collections and produce them as a matter of course. ePub has extensive support in the readers for bookmarking and note taking to support our reading.

How can we make ePub? There are two ways; first by passing through an HTML intermediary, and second by processing a PDF file. In both cases we can use a tool called `calibre` to make the conversion to ePub.

1. LaTeX $\rightsquigarrow$ HTML $\rightsquigarrow$ ePub
   LaTeX2HTML (LaTeX) $\rightsquigarrow$ HTML
   `calibre` (HTML) $\rightsquigarrow$ ePub

I haven't explored this route very much, but it could be a feasible solution. The drawback is that the conversion to HTML is not particularly robust with respect to using arbitrary LaTeX package files to enhance our documents.

2. LaTeX ⇝ PDF ⇝ ePub
   PDFLaTeX (LaTeX) ⇝ HTML
   `calibre` (PDF) ⇝ ePub
   This approach is discussed further in section 3.3.

2 PDF metadata

Metadata in PDF files is used by the e-readers to fill out author information in the list of documents available. You can use a PDF manipulator such as Acrobat or you can include the information directly using the `hyperref` package when you create your PDF file with PDFLaTeX.

```
\usepackage[%
   bookmarks=true% style guide
  ,pdfauthor={Alan Wetmore}%
  ,pdfcreator={pdfLaTeX article.cls}%
  ,pdfkeywords={e-Readers,TUG2011}%
  ,pdfsubject={e-Readers}%
  ,pdftitle={e-Readers and LaTeX}%
]{hyperref}
```

3 The most important features

3.1 Standards compliance

It turns out that even though PDF and ePub are reasonably well defined "standards", e-readers are not particularly consistent with how they consume and display these files. When we feed these devices the same files, they can produce substantially different displays and they expose different navigation and viewing options for us to use when we explore and consume our documents.

3.2 Legacy PDF files

Many of us have large collections of PDF files that we have accumulated over the years. The older ones were of course generated without regard to reading on anything other than paper, A4 or letter-size for the most part, or on our computer screens. Many of our gizmos and gadgets now ship with nothing more than an abbreviated "Quick Start Guide"; sometimes with a CD in the package with a longer manual, sometimes with or without a hint that there is a longer manual hidden somewhere on a manufacturer's web site that we can download. In section 4, we'll explore a recently published textbook to see how well we can read it on the various e-readers.

3.3 Converting PDF to ePub with calibre

One popular and powerful tool for converting between the various e-book formats is `calibre` [5]. This is a free and open source e-book library management application; it is much more than conversion software. In addition to organizing documents on your computer, it interfaces with your reader device when you plug it in, and then copies your documents to and fro.

`calibre` doesn't extract a lot of structure information from PDF files, and so doesn't generate particularly good ePub files as a result. My experiments suggest that this is not yet a very fruitful path; perhaps the `poppler` library is now ready to be used to improve this.

Alternatively, LaTeX2HTML might well be a better starting point for generating ePub format documents.

3.4 Generating new PDF files

We will want to see what options we have for typesetting new PDF files to be used with our e-readers. In section 6 we'll take a look at what a couple of very simple choices with some standard packages can do for us.

4 Exploring a textbook

We will be using the text *Principles of Uncertainty* [3] by Joseph P. Kadane of Carnegie Mellon University. This is a large (499 pages) textbook of a traditional size; it was produced by our friend Heidi Sestrich using LaTeX. It is also available as a PDF file from the author's web site.

4.1 The original document

The original document was typeset on US letter (8.5 × 11 inch) size paper with margins and crop marks for the final production size of approximately 6 by 9 inches. It was typeset with pdfTeX using LaTeX and `hyperref`. There is plenty of moderately high level mathematical notation involved to exercise the e-reader's rendering engines. All of the readers successfully load the book and allow us to read it, with varying degrees of flexibility and robustness. I'll be concentrating on Chapter 4 of the book; the demonstrations are viewable on the `river-valley.tv` web site with the conference videos.

4.2 Nook

As for PDF files, loading and viewing a PDF file looks great at first; the full page is displayed on the screen with mathematics intact. However, when you zoom in for a closer look, things quickly break down. The zoom control uses the same seven font size buttons as the ePub viewer, but — instead of zooming — the Nook adjusts the font size and reflows the document. Plain text doesn't fare too badly, but mathematics is very corrupted, with the layout destroyed and many missing symbols.

During the conference Ross Moore supplied some "tagged pdf" files with demonstrations of mathematics; when loaded onto the Nook all the mathematics in these survived the zooming tests. This might be one route to PDF files which are usable on all of the devices.

4.3 Kobo

When viewing PDF files, a double tap to the screen zooms the image to 200%; there are also seven zoom levels available through a slider control. When the image is zoomed, you can drag it around the screen to change the viewport, with an overview widget that shows which part of the page you are on. Unfortunately there is no simple page advance mechanism in zoomed mode, e.g., tapping at the margins doesn't advance the page. You must drag the image to expose either a left or right margin or activate the more complete navigation widget, or double tap again to return to full page mode and then tap at the margin.

4.4 iRiver

The original test document didn't display at all well, appearing as broken pages with overlong lines. However, the version cropped to the text box appeared correctly with mathematics intact. The iRiver has a zoom control for PDF files that sort of works; alternatively, one of the buttons brings up a rotate menu allowing the image to rotate 90° to landscape mode that expands the width of the image to fill the wider screen by applying a one-third magnification. For a well-crafted document this will usually be enough zoom to make out the details of sub- and super-scripts.

5 Idiosyncrasies of the devices

5.1 Nook

One interesting thing about the Nook is the simple ability to load and use your own images for screensavers. I've loaded mine with a collection of iconic TUG images and meeting posters.

5.2 Kobo

After copying files to the SD card memory, when you eject the memory card from the computer, the Kobo spends some time "processing" the files. It is likely that this includes scanning for meta-data and rebuilding the list of books.

5.3 iRiver

The iRiver doesn't have a touch screen, instead it has lots of buttons arranged mostly as a `QWERTY` keyboard plus a few more navigation and control keys. It also uses full size SD cards instead of the microSD format that the other devices use.

6 Generating new PDF files

As promised, we will now explore what happens as we process some text with a couple of options. The first thing we will do is use the `12pt` option to get larger text as the starting point on the device. This should reduce the need for magnifying text that caused problems for the Nook, while maintaining maximum compatibility with LaTeX packages and options.

First we will use the `geometry` package to choose our "paper" size.

6.1 Letter paper size

Using the `geometry` package to see where things go, we start with the standard 8.5 by 11 inch `papersize`. We are also using the `margin=1in` option. This results in 95 character lines and oversize margins that waste considerable screen space.

6.2 The screen option

Adding the `screen` option from the `geometry` package generates output for a 4×3 screen ratio, which matches our screen rotated 90°. It uses a 225 mm by 180 mm `papersize`. This results in 100 character lines.

6.3 An epaper option

Setting the `papersize` to 100 mm by 125 mm results in 54 character lines. This is the final choice for the `epaper` option. The following line can be added to the internal database in the package file to add our new epaper option.

```
\@namedef{Gm@epaper}#1{%
     \Gm@setsize{#1}(100,125){mm}}%
       %for e-readers
```

And for a final consideration, we'll increase the margins to 2mm. Until the `geometry` package has been updated with a suitable `epaper` option we can use the following in our preamble.

```
\usepackage[%
    papersize={100mm,125mm},%
    margin=2mm,%
    includeheadfoot%
    ]{geometry}
```

7 The future

We can expect more generations of these devices in the next few years. Will they have better PDF engines? We can hope. We can also hope that software updates to the current machines improve the PDF engines. Maybe they will make use of internal PDF links so our table of contents and cross references will work.

Will MathML be accepted into a future revision of ePub? Probably, but will it actually be supported by the readers? That is a much less certain outcome.

Will searching within a document be improved? The current capabilities are simply too slow for searching through reference materials, and without live links in the index, that doesn't really offer much of an option.

While e-readers seem to be popular for best sellers and light reading, they don't as yet replace real textbooks or the more robust PDF capabilities of real computers or powerful tablets. But perhaps they are more suited to consuming smaller chunks of material on the order of individual lessons from the Khan Academy [4].

References

[1] Cheswick, William: *iTEX — Document formatting in an ereader world*, *TUGboat* **32**:2, 2011, 158–162. `http://tug.org/TUGboat/tb32-2/tb101cheswick.pdf`.

[2] Hagen, Hans: *E-books: Old Wine in New Bottles*, *TUGboat* **32**:2, 2011, 152–158. `http://tug.org/TUGboat/tb32-2/tb101hagen.pdf`.

[3] Kadane, Joseph P.: *Principles of Uncertainty*, CRC Press, 2011. `http://uncertainty.stat.cmu.edu`.

[4] *Khan Academy*. `http://www.khanacademy.org`.

[5] Schember, John: *Calibre Quick Start Guide*. `http://calibre-ebook.com/about`.

◇ Alan Wetmore
US Army Research Laboratory
alan dot wetmore (at) gmail dot com

Ebooks and paper sizes: Output routines made easier

Boris Veytsman and Michael Ware

Abstract

The idea of a book being a collection of pages is so ingrained that modern electronic book readers often try to faithfully reproduce this feature — up to elaborate simulations of page turns. However, traditional pages are not necessary and often inconvenient in electronic books. It is often easier to scroll the text than to turn the pages, and text reflowing makes the use of folios a rather strange way to refer to the position inside a text.

We argue that it is more natural to paginate electronic books according to their logical structure, when a "page" corresponds to a sectional unit of the book. This leads to rather long pages, with the height of the page depending on the length of the corresponding unit.

We discuss how to implement these pages in TeX and provide a basic introduction to output routines in TeX for a beginning TeXnician. We also provide exercises for a slightly more advanced reader.

1 Introduction

TeX-based systems have been creating high-quality electronic books (ebooks) for decades, with PDF becoming the dominant format in recent years. While there are many macro packages that optimize TeX's PDF output for electronic reading (with links, etc.), the basic paradigm of TeX-produced ebooks is still very much tied to the ideas of a physical book — the document is formatted into a series of identically-sized pages with the position of floating environments and graphics adjusted so as to avoid the page breaks. This type of ebook works well both in print and on screens that are similar in size to a piece of paper. However, the proliferation of mobile devices with small screens presents a new opportunity and challenge for ebook creators.

Typical PDF pages cannot be displayed in their entirety on small screens. To read a full-page document on a smartphone, one has to zoom in until the text is sufficiently large, and then pan around through the document. This makes for a difficult reading experience, and has caused ebook creators to largely abandon fixed-layout schemes like PDF in favor of reflowable formats such as HTML. This "just-in-time" layout strategy allows the page layout software to adjust the font size to a readable level and the line width so that one only has to scroll in one direction to view all of the text.

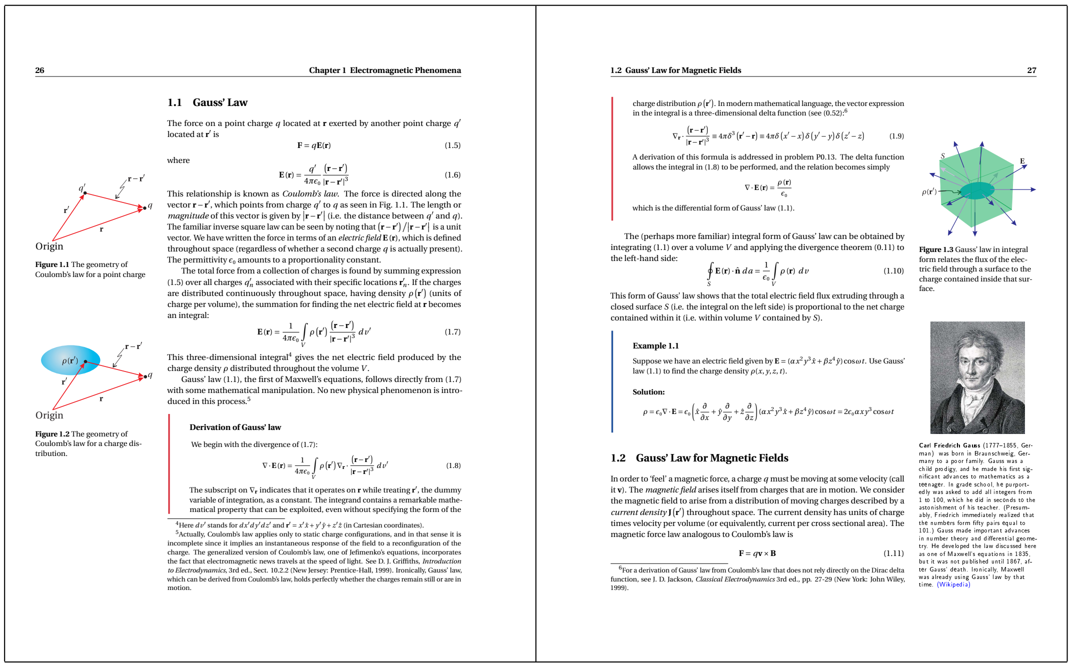
Current reflow schemes work well for documents that are primarily text-based, but they typically have limited support for technical documents with an abundance of equations and figures. Since equations and figures are a critical part of a technical text, the current generation of ebook readers does not provide a workable solution for most technical documents. Some progress has been made in browser-based HTML with technologies such as MathJax (see `http://www.mathjax.org/`) and we may eventually have a dedicated ebook reader that uses TeX as its layout engine (Bazargan, 2009). However, it seems likely that we will continue to use PDF documents for some time for technical ebooks, so the question remains: how can TeX-created documents be optimized for a smooth ebook reading experience?

To make these issues more concrete, consider the layout of the textbook pages shown in Fig. 1. This is a typical textbook layout, with figures in the margins near the referring text, footnotes at the bottom of each page, and headings atop each page to give the reader information about their location in the text. These features make for an easy reading experience on paper or on a large screen, but can get in the way when viewed on even a moderately sized e-reader such as an iPad. In this reading environment, a reader needs to scroll horizontally to view figures, and the flow of the text is interrupted on each page to show the footnotes and header information.

Since a PDF document has a fixed layout,[1] it is necessary to choose the page size parameters in advance in a way that works for the intended screen size. An obvious first approach is to make the TeX page size match the size of the intended reading device and minimize the margins around the text (Cheswick, 2011). However, this has some drawbacks. As the page size gets smaller, it becomes increasingly difficult to lay out non-text elements such as figures and larger equations. Since the pages can be quite small, there are a lot of page breaks, TeX's float placement routines tend to leave many pages with awkward white space, and the floats are far from the referring text.

A better approach is to make the pages fit the target screen width, but be very tall relative to the intended screen size. Each logical division of the text (say, a section) is placed on its own page. In this scenario, figure placement and equation layout are easy for TeX, since the page breaks essentially disappear. The reader can view the full width and scroll vertically through the content of a section, and flipping

[1] One can reflow the text portion of a document in newer PDF specifications, but images do not reflow.



26 Chapter 1 Electromagnetic Phenomena

1.1 Gauss' Law

The force on a point charge q located at $\mathbf{r}$ exerted by another point charge q' located at $\mathbf{r}'$ is

$$\mathbf{F} = q\mathbf{E}(\mathbf{r}) \tag{1.5}$$

where

$$\mathbf{E}(\mathbf{r}) = \frac{q'}{4\pi\epsilon_0}\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3} \tag{1.6}$$

This relationship is known as *Coulomb's law*. The force is directed along the vector $\mathbf{r}-\mathbf{r}'$, which points from charge q' to q as seen in Fig. 1.1. The length or *magnitude* of this vector is given by $|\mathbf{r}-\mathbf{r}'|$ (i.e. the distance between q' and q). The familiar inverse square law can be seen by noting that $(\mathbf{r}-\mathbf{r}')/|\mathbf{r}-\mathbf{r}'|$ is a unit vector. We have written the force in terms of an *electric field* $\mathbf{E}(\mathbf{r})$, which is defined throughout space (regardless of whether a second charge q is actually present). The permittivity ϵ_0 amounts to a proportionality constant.

The total force from a collection of charges is found by summing expression (1.5) over all charges q'_n associated with their specific locations $\mathbf{r}'_n$. If the charges are distributed continuously throughout space, having density $\rho(\mathbf{r}')$ (units of charge per volume), the summation for finding the net electric field at $\mathbf{r}$ becomes an integral:

$$\mathbf{E}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0}\int_V \rho(\mathbf{r}')\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3}\,dv' \tag{1.7}$$

This three-dimensional integral[4] gives the net electric field produced by the charge density ρ distributed throughout the volume V.

Gauss' law (1.1), the first of Maxwell's equations, follows directly from (1.7) with some mathematical manipulation. No new physical phenomenon is introduced in this process.[5]

Figure 1.1 The geometry of Coulomb's law for a point charge

Figure 1.2 The geometry of Coulomb's law for a charge distribution.

Derivation of Gauss' law

We begin with the divergence of (1.7):

$$\nabla\cdot\mathbf{E}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0}\int_V \rho(\mathbf{r}')\nabla_{\mathbf{r}}\cdot\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3}\,dv' \tag{1.8}$$

The subscript on $\nabla_{\mathbf{r}}$ indicates that it operates on $\mathbf{r}$ while treating $\mathbf{r}'$, the dummy variable of integration, as a constant. The integrand contains a remarkable mathematical property that can be exploited, even without specifying the form of the

[4]Here dv' stands for $dx'dy'dz'$ and $\mathbf{r}' = x'\hat{x} + y'\hat{y} + z'\hat{z}$ (in Cartesian coordinates).

[5]Actually, Coulomb's law applies only to static charge configurations, and in that sense it is incomplete since it implies an instantaneous response of the field to a reconfiguration of the charge. The generalized version of Coulomb's law, one of Jefimenko's equations, incorporates the fact that electromagnetic news travels at the speed of light. See D. J. Griffiths, *Introduction to Electrodynamics*, 3rd ed., Sect. 10.2.2 (New Jersey: Prentice-Hall, 1999). Ironically, Gauss' law, which can be derived from Coulomb's law, holds perfectly whether the charges remain still or are in motion.

1.2 Gauss' Law for Magnetic Fields 27

charge distribution $\rho(\mathbf{r}')$. In modern mathematical language, the vector expression in the integral is a three-dimensional delta function (see (0.52):[6]

$$\nabla_{\mathbf{r}}\cdot\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3} \equiv 4\pi\delta^3(\mathbf{r}'-\mathbf{r}) \equiv 4\pi\delta(x'-x)\,\delta(y'-y)\,\delta(z'-z) \tag{1.9}$$

A derivation of this formula is addressed in problem P0.13. The delta function allows the integral in (1.8) to be performed, and the relation becomes simply

$$\nabla\cdot\mathbf{E}(\mathbf{r}) = \frac{\rho(\mathbf{r})}{\epsilon_0}$$

which is the differential form of Gauss' law (1.1).

The (perhaps more familiar) integral form of Gauss' law can be obtained by integrating (1.1) over a volume V and applying the divergence theorem (0.11) to the left-hand side:

$$\oint_S \mathbf{E}(\mathbf{r})\cdot\hat{\mathbf{n}}\,da = \frac{1}{\epsilon_0}\int_V \rho(\mathbf{r})\,dv \tag{1.10}$$

This form of Gauss' law shows that the total electric field flux extruding through a closed surface S (i.e. the integral on the left side) is proportional to the net charge contained within it (i.e. within volume V contained by S).

Figure 1.3 Gauss' law in integral form relates the flux of the electric field through a surface to the charge contained inside that surface.

Example 1.1

Suppose we have an electric field given by $\mathbf{E} = (\alpha x^2y^3\hat{x} + \beta z^4\hat{y})\cos\omega t$. Use Gauss' law (1.1) to find the charge density $\rho(x,y,z,t)$.

Solution:

$$\rho = \epsilon_0\nabla\cdot\mathbf{E} = \epsilon_0\left(\hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z}\right)(\alpha x^2y^3\hat{x} + \beta z^4\hat{y})\cos\omega t = 2\epsilon_0\alpha xy^3\cos\omega t$$

Carl Friedrich Gauss (1777–1855, German) was born in Braunschweig, Germany to a poor family. Gauss was a child prodigy, and he made his first significant advances to mathematics as a teenager. In grade school, he purportedly was asked to add all integers from 1 to 100, which he did in seconds to the astonishment of his teacher. (Presumably, Friedrich immediately realized that the numbers form fifty pairs equal to 101.) Gauss made important advances in number theory and differential geometry. He developed the law discussed here as one of Maxwell's equations in 1835, but it was not published until 1867, after Gauss' death. Ironically, Maxwell was already using Gauss' law by that time. (Wikipedia)

1.2 Gauss' Law for Magnetic Fields

In order to 'feel' a magnetic force, a charge q must be moving at some velocity (call it $\mathbf{v}$). The *magnetic field* arises itself from charges that are in motion. We consider the magnetic field to arise from a distribution of moving charges described by a *current density* $\mathbf{J}(\mathbf{r}')$ throughout space. The current density has units of charge times velocity per volume (or equivalently, current per cross sectional area). The magnetic force law analogous to Coulomb's law is

$$\mathbf{F} = q\mathbf{v}\times\mathbf{B} \tag{1.11}$$

[6]For a derivation of Gauss' law from Coulomb's law that does not rely directly on the Dirac delta function, see J. D. Jackson, *Classical Electrodynamics* 3rd ed., pp. 27-29 (New York: John Wiley, 1999).

Figure 1: Some typical pages of a textbook formatted for paper printing. (See `http://optics.byu.edu/`)

to the next "page" moves to the next logical division of the text. This layout has the added benefit of getting rid of many old typography problems, such as widows, orphans, badly placed floating material, etc.

However, since the text of each section is of a different length while TEX maintains a fixed page length, each page has white space at the bottom that a user must scroll through. It would be nice if TEX provided a way to make each page just the right size for its content, and fortunately it does.

Figure 2 shows the same text as Fig. 1 formatted using the "tall page" approach described above. The figures have been moved inline with the text using some straightforward modifications to the margin figure macros, and the width of the text is now appropriate for full-width viewing on a tablet device. The informational header is still retained at the top of the page and the footnotes appear at the bottom, but they no longer interrupt the flow of the text. Now the reader can view the full content of this section of the textbook simply by scrolling vertically. To view the next section of the book, the reader simply moves to the next "page" of the ebook.

With some planning, it is very reasonable to design a document class that can be converted from a traditional paper layout like Fig. 1 to an ebook layout like Fig. 2 with a single command switch.[2] This allows an author to easily provide multiple layouts that are appropriate for both paper and on-screen reading. To do this, it is necessary to understand how paper size is treated in TEX.

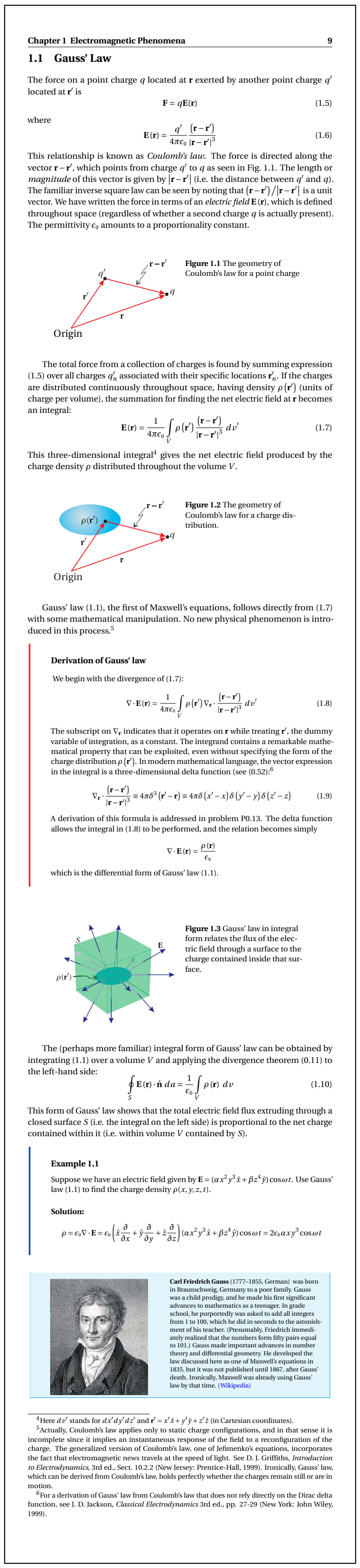
2 Paper length in TEX

It might be surprising for a beginning TEXnician to learn the extent to which the "classical" TEX cares about actual paper dimensions: namely, it does not care about them at all — and does not even know about them. Of course, paper dimensions are mentioned in many TEX macro packages — for example, LATEX `geometry` package (Umeki, 2010), but in many cases they are just used to calculate the dimensions of the text area.

One can argue that this feature corresponds to the rôle of TEX as a compositor: in a classic printing shop a compositor puts words into a matrix, but the choice of the paper on which the imprint on was done by another artisan. A more prosaic explanation is that the printers used during the time TEX was written had no means to change the paper dimensions, so it made little sense to control them in a typesetting program.

DVI drivers, however, could deal with paper size, for example, through the use of PostScript commands as arguments of `\special`s in `dvips`. Since `pdftex` is both a TEX engine *and* a (PDF) driver, it has commands `\pdfpageheight` and `\pdfpagewidth`, which deal with paper dimensions directly. In this section

[2] For example, the code that made the example pages in Figs. 1 and 2 is available at `optics.byu.edu`.



Chapter 1 Electromagnetic Phenomena 9

1.1 Gauss' Law

The force on a point charge q located at $\mathbf{r}$ exerted by another point charge q' located at $\mathbf{r}'$ is

$$\mathbf{F} = q\mathbf{E}(\mathbf{r}) \qquad (1.5)$$

where

$$\mathbf{E}(\mathbf{r}) = \frac{q'}{4\pi\epsilon_0}\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3} \qquad (1.6)$$

This relationship is known as *Coulomb's law*. The force is directed along the vector $\mathbf{r}-\mathbf{r}'$, which points from charge q' to q as seen in Fig. 1.1. The length or *magnitude* of this vector is given by $|\mathbf{r}-\mathbf{r}'|$ (i.e. the distance between q' and q). The familiar inverse square law can be seen by noting that $(\mathbf{r}-\mathbf{r}')/|\mathbf{r}-\mathbf{r}'|$ is a unit vector. We have written the force in terms of an *electric field* $\mathbf{E}(\mathbf{r})$, which is defined throughout space (regardless of whether a second charge q is actually present). The permittivity ϵ_0 amounts to a proportionality constant.

Figure 1.1 The geometry of Coulomb's law for a point charge

The total force from a collection of charges is found by summing expression (1.5) over all charges q'_n associated with their specific locations $\mathbf{r}'_n$. If the charges are distributed continuously throughout space, having density $\rho(\mathbf{r}')$ (units of charge per volume), the summation for finding the net electric field at $\mathbf{r}$ becomes an integral:

$$\mathbf{E}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0}\int_V \rho(\mathbf{r}')\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3}\,dv' \qquad (1.7)$$

This three-dimensional integral[4] gives the net electric field produced by the charge density ρ distributed throughout the volume V.

Figure 1.2 The geometry of Coulomb's law for a charge distribution.

Gauss' law (1.1), the first of Maxwell's equations, follows directly from (1.7) with some mathematical manipulation. No new physical phenomenon is introduced in this process.[5]

Derivation of Gauss' law

We begin with the divergence of (1.7):

$$\nabla\cdot\mathbf{E}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0}\int_V \rho(\mathbf{r}')\nabla_{\mathbf{r}}\cdot\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3}\,dv' \qquad (1.8)$$

The subscript on $\nabla_{\mathbf{r}}$ indicates that it operates on $\mathbf{r}$ while treating $\mathbf{r}'$, the dummy variable of integration, as a constant. The integrand contains a remarkable mathematical property that can be exploited, even without specifying the form of the charge distribution $\rho(\mathbf{r}')$. In modern mathematical language, the vector expression in the integral is a three-dimensional delta function (see (0.52):[6]

$$\nabla_{\mathbf{r}}\cdot\frac{(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3} \equiv 4\pi\delta^3(\mathbf{r}'-\mathbf{r}) \equiv 4\pi\delta(x'-x)\,\delta(y'-y)\,\delta(z'-z) \qquad (1.9)$$

A derivation of this formula is addressed in problem P0.13. The delta function allows the integral in (1.8) to be performed, and the relation becomes simply

$$\nabla\cdot\mathbf{E}(\mathbf{r}) = \frac{\rho(\mathbf{r})}{\epsilon_0}$$

which is the differential form of Gauss' law (1.1).

Figure 1.3 Gauss' law in integral form relates the flux of the electric field through a surface to the charge contained inside that surface.

The (perhaps more familiar) integral form of Gauss' law can be obtained by integrating (1.1) over a volume V and applying the divergence theorem (0.11) to the left-hand side:

$$\oint_S \mathbf{E}(\mathbf{r})\cdot\hat{\mathbf{n}}\,da = \frac{1}{\epsilon_0}\int_V \rho(\mathbf{r})\,dv \qquad (1.10)$$

This form of Gauss' law shows that the total electric field flux extruding through a closed surface S (i.e. the integral on the left side) is proportional to the net charge contained within it (i.e. within volume V contained by S).

Example 1.1

Suppose we have an electric field given by $\mathbf{E} = (\alpha x^2 y^3\hat{x} + \beta z^4\hat{y})\cos\omega t$. Use Gauss' law (1.1) to find the charge density $\rho(x,y,z,t)$.

Solution:

$$\rho = \epsilon_0\nabla\cdot\mathbf{E} = \epsilon_0\left(\hat{x}\frac{\partial}{\partial x}+\hat{y}\frac{\partial}{\partial y}+\hat{z}\frac{\partial}{\partial z}\right)(\alpha x^2 y^3\hat{x} + \beta z^4\hat{y})\cos\omega t = 2\epsilon_0\alpha x y^3\cos\omega t$$

Carl Friedrich Gauss (1777–1855, German) was born in Braunschweig, Germany to a poor family. Gauss was a child prodigy, and he made his first significant advances to mathematics as a teenager. In grade school, he purportedly was asked to add all integers from 1 to 100, which he did in seconds to the astonishment of his teacher. (Presumably, Friedrich immediately realized that the numbers form fifty pairs equal to 101.) Gauss made important advances in number theory and differential geometry. He developed the law discussed here as one of Maxwell's equations in 1835, but it was not published until 1867, after Gauss' death. Ironically, Maxwell was already using Gauss' law by that time. (Wikipedia)

[4]Here dv' stands for $dx'dy'dz'$ and $\mathbf{r}' = x'\hat{x} + y'\hat{y} + z'\hat{z}$ (in Cartesian coordinates).

[5]Actually, Coulomb's law applies only to static charge configurations, and in that sense it is incomplete since it implies an instantaneous response of the field to a reconfiguration of the charge. The generalized version of Coulomb's law, one of Jefimenko's equations, incorporates the fact that electromagnetic news travels at the speed of light. See D. J. Griffiths, *Introduction to Electrodynamics*, 3rd ed., Sect. 10.2.2 (New Jersey: Prentice-Hall, 1999). Ironically, Gauss' law, which can be derived from Coulomb's law, holds perfectly whether the charges remain still or are in motion.

[6]For a derivation of Gauss' law from Coulomb's law that does not rely directly on the Dirac delta function, see J. D. Jackson, *Classical Electrodynamics* 3rd ed., pp. 27-29 (New York: John Wiley, 1999).

Figure 2: The content from the pages in Fig. 1 formatted in an "ebook friendly" layout.

```
\vsize=500cm
\pdfpageheight=500cm
\hrule
\vskip 1in
\centerline{\bf A SHORT STORY}
\vskip 6pt
\centerline{\sl by A. U. Thor}
\vskip .5cm
Once upon a time, in a distant  galaxy called
\"O\"o\c c, there lived a computer named
R.~J. Drofnats.

Mr.~Drofnats---or ‘‘R. J.,’’ as
he preferred to be called---was happiest when
he was at work typesetting beautiful documents.
\vskip 1in
\hrule
\vfill\eject
\bye
```

Figure 3: Long page in plain TEX: A simple example.

we discuss how to employ these commands to set up page height for electronic books.

We start from a simple example shown on Figure 3. It differs from the classical TEX story (Knuth, 1994) by two lines: the command `\vsize=500cm` tells TEX that the "galley" is very long, and the command `\pdfpageheight=500cm` sets the paper height to the same value.

Exercise 1: Actually it is a rather poor idea to set both these lengths to the same value. Why?

When we compile this file with `pdftex`, we get a very long page with a lot of white space at the bottom. As discussed in the previous section, we would like to get rid of the extra white space and make the page size fit the text. To do this we need to set `\pdfpageheight` dynamically. For this would like to know the height of the text at the moment the page is "shipped out". This means changing the output routine of TEX.

3 Output routines

As TEX processes a document, it arranges a block of material (text, equations, figures, etc.) until the block's size is near a predetermined target (stored in `\pagegoal`), or it runs out of material. At this point, TEX hands the collected material off to an output routine that does some final manipulation and adds page numbers, headers, footers, etc. to the page. The page is then shipped out to the final document, and TEX moves on to the next page.

Traditionally, output routines are considered one of the hardest parts of TEX — probably because

```
\output={\shipout\box255}
```

Figure 4: World's simplest output routine.

```
\output={%
  \pdfpageheight=\pagetotal
  \advance\pdfpageheight by 2in
  \shipout\box255}
\vsize=500cm
\hrule
\vskip 1in
\centerline{\bf A SHORT STORY}
\vskip 6pt
...
```

Figure 5: Modified plain TEX example.

they *are* hard. This section is intended to be a gentle introduction. A more rigorous introduction can be found in the book (Eijkhout, 2007), and the comprehensive tutorial in the papers (Salomon, 1990a; Salomon, 1990b; Salomon, 1990c).

For our purposes, we need two facts about output routines: first, that the output page is contained in box 255, and second, that the current text height is contained in the value `\pagetotal`. The first fact leads to the world's simplest output routine (Figure 4). The second one suggests the following modification of this routine: let us set up the paper height to `\pagetotal` plus 2 inches to allow for 1 inch top and bottom margins. This leads to the modified example shown on Figure 5. This example produces a page of the height determined by the material on the page — exactly what we wanted!

Of course our output routines are rather toy-like. A self-respecting output routine should include headers, footers, folios, footnotes, etc. We can take plain TEX output routine and patch it with the code setting `\pdfpageheight`, but here we shall leave this as an exercise to the reader.

Exercise 2: Try to patch the plain TEX output routine in the way described above.

Instead we turn to the LATEX output routine. It is very powerful and complex, and may look intimidating for a beginning (or expert) TEXnician. Fortunately, there are packages that allow for patching this routine without looking too deeply into its code. In this paper we will use one such package, everyshi (Schröder, 2001). This package allows one to *add* code to the LATEX output routine. Thus we can write down our patch as shown on Figure 6.

Unfortunately, this solution has a flaw. To see it, try to compile a file shown on Figure 7. It has rather lengthy footnotes — and the output routine patch in Fig. 6 cuts them off!

```
\textheight500cm
\usepackage{everyshi}
\EveryShipout{%
  \pdfpageheight=\pagetotal
  \advance\pdfpageheight by 2\topmargin
  \advance\pdfpageheight by 2in}
```

Figure 6: LATEX output routine patched.

```
\documentclass{article}
\usepackage{everyshi,lipsum}
\pagestyle{empty}
\textheight500cm
\EveryShipout{%
  \pdfpageheight=\pagetotal
  \advance\pdfpageheight by 2\topmargin
  \advance\pdfpageheight by 2in}
\begin{document}

\lipsum[3-5]
\pagebreak

This line has a footnote\footnote{\lipsum[6-8]}.
\lipsum[1]

And this line too\footnote{\lipsum[12]}.
\pagebreak

\end{document}
```

Figure 7: A LATEX file with footnotes.

What happens to the footnotes? It turns out that `\pagetotal` is the height of the text part of the page, and footnotes are not accounted for here. To resolve this issue, we note that the dimension `\pagegoal` keeps the height of the page *minus* the height of footnotes: this is the height of the page "accessible for the text". Therefore the height of the footnotes is `\textheight` *minus* `\pagegoal`. Once we understand this, we can easily modify our code as in Figure 8 to account for the footnotes.

Exercise 3: Compile the code in Figure 8 and check that it solves our problem.

Exercise 4: Ross Moore suggested a different solution to the problem of footnotes, based on the fact that minipages include the footnotes' height into the total height. Try to implement it.

Exercise 5: Change `\pagebreak` to `\newpage` in Figure 8. What happens? Why?

```
\documentclass{article}
\usepackage{everyshi,lipsum}
\pagestyle{empty}
\textheight500cm
\EveryShipout{%
  \pdfpageheight=\pagetotal
  \advance\pdfpageheight by 2in
  \advance\pdfpageheight by 2\topmargin
  \advance\pdfpageheight by \textheight
  \advance\pdfpageheight by -\pagegoal}
\begin{document}

\lipsum[3-5]
\pagebreak

This line has a footnote\footnote{\lipsum[6-8]}.
\lipsum[1]

And this line too\footnote{\lipsum[12]}.
\pagebreak

\end{document}
```

Figure 8: A corrected LaTeX output routine.

4 Conclusion

Using the procedure described above, one can create well-formatted content with essentially no superfluous white space. This seems an excellent approach for producing technical electronic content where on-the-fly layout engines will not work.

Finally, we note that the current state of PDF viewers on mobile devices still leaves something to be desired. For example, on the iPad, we have not found an app that properly handles PDF links. Also, the current generation of apps have limited ability to control the zoom state of a page. Some require one to scroll through a PDF vertically, and apps that let you flip through pages with gestures typically scale each page to fit entirely on the screen; there were no options for fixing the page width to fit. For a long page, this produces a thin strip of unreadable text which then must be manually zoomed to read. Nevertheless, these annoyances seem likely to be fixed as new apps are continually produced.

Acknowledgements

We are grateful to Frank Mittelbach who found an error in our code, to Ross Moore who suggested an alternative solution for the problem of footnotes and to other participants of TUG 2011 for many interesting suggestions and comments.

References

Bazargan, Kaveh. “TeX as an ebook reader”. *TUGboat* **30**(2), 272–73, 2009. `http://river-valley.tv/tex-as-an-ebook-reader`, `http://tug.org/TUGboat/30-2/tb95bazargan.pdf`.

Cheswick, William. “TeX and the iPad”. *TUGboat* **32**(2), 158–163, 2011. `http://river-valley.tv/tex-and-the-ipad`, `http://tug.org/TUGboat/32-2/tb102cheswick.pdf`.

Eijkhout, Victor. *TeX by Topic*. Lulu, 2007. `http://eijkhout.net/texbytopic/texbytopic.html`.

Knuth, Donald Ervin. *The TeXbook*. Computers & Typesetting A. Addison-Wesley Publishing Company, Reading, MA, 1994. Illustrations by Duane Bibby.

Salomon, David. “Output Routines: Examples and Techniques. Part I: Introduction and Examples”. *TUGboat* **11**(1), 69–85, 1990a.

Salomon, David. “Output Routines: Examples and Techniques. Part II: OTR Techniques”. *TUGboat* **11**(2), 212–236, 1990b.

Salomon, David. “Output Routines: Examples and Techniques. Part III: Insertions”. *TUGboat* **11**(4), 588–605, 1990c.

Schröder, Martin. *The everyshi package*, 2001. `http://mirrors.ctan.org/macros/latex/packages/ms`.

Umeki, Hideo. *The geometry package*, 2010. `http://mirrors.ctan.org/macros/latex/contrib/geometry`.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
`borisv (at) lk dot net`

◇ Michael Ware
Department of Physics and Astronomy, N283 ESC
Brigham Young University
Provo, UT 84602
`ware (at) byu dot edu`

LaTeX to ePub

Rishi T.

Abstract

We have developed a workflow to generate ePub [1] from a LaTeX document. This workflow has two main parts. The first part converts the document sources in LaTeX format to XML. We have been using this part of the workflow for many years. The second part generates ePub from the XML documents thus created.

This workflow is completely automated and makes use of TeX4ht, XSLT and ANT scripts.

1 Evolution of our XML-to-ePub workflow

At River Valley we have been engaged in the task of perfecting a workflow for the generation of high quality ePub directly from XML sources. Since we are primarily dealing with scientific, technical, and medical (STM) books with complex mathematical formulae, the original sources of these contents will invariably have been authored in TeX. As our ePub workflow demands XML as its input, we use TeX4ht to convert the TeX sources into XML.

Two years back, we developed an XML to ePub filter, but before long, we were forced to abandon it as it suffered from several deficiencies owing to its poor design and use of inappropriate technologies for processing XML sources. One of the major handicaps of this filter was that it required repeated manual intervention to edit the XML sources to suit its rigid input format. This experience forced us to review the design of the filter from the ground up, and develop a new one flexible enough to meet the needs of the evolving ePub specifications, and to be customizable enough for processing the XMLs of different DTDs. As the future of publishing seems to be moving more and more towards ePub, we thought it appropriate to invest more time and effort on it. Now the development team is happy that, at last, it can provide a robust solution.

The latest workflow is mainly based on XSLT [2] and ANT scripts [3]. Our main concerns about the workflow were the following.

- It should be user-friendly.
- Even a novice developer should be able to maintain it.
- It should be highly customisable without modifying the core area.
- It should require no manual intervention.
- It must be an XML-based and cross-platform solution.

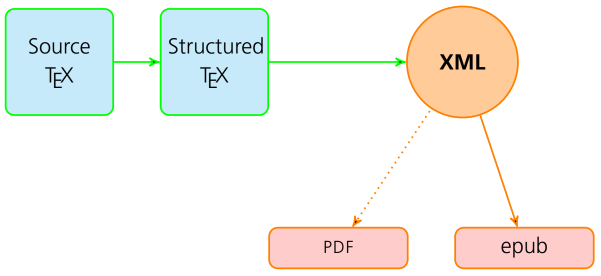


Figure 1: Schematic diagram of our workflow

2 Workflow

A simple schematic diagram of our workflow is given in figure 1. It can be described as follows.

1. Create a structured TeX document from the author's source document. Structured TeX means a TeX document, where the details are tagged clearly. An example of how author details are coded is given below:

```
\author{%
  \fnm{Rajagopal}
  \snm{CV}
}
\address{%
  \orgname{River Valley Technologies}
  \city{Trivandrum}
  \cnty{India}
}
```

   Structuring is done with the aid of TeX4ht and some scripts written in Vim.
2. This structured LaTeX document is converted to an XML format, which follows Elsevier's book DTD (`book521.dtd`).[1] TeX4ht is used for the TeX-to-XML conversion.
3. Next, bitmapped equations (images of equations) are created for all MathML tags. Images are used rather than MathML tags since current e-book readers do not support MathML rendering.
4. Then the XSLT style sheet is applied on this XML document and an ePub is created.

2.1 Working method

2.1.1 Input

(1) XML files, bitmapped equations, and any external entities (such as figures) loaded in the XML files. (2) A hub file. This is an XML file that includes the metadata and the list of XML files that should be

[1] In our experience, this is one of the best DTDs, covering almost all types of STM content as far as a standard book is concerned.

```
<files>
<title>Field Guide</title>
<author>Yakov</author>
<cover name="cover/cover.jpg"/>
<stylesheet name="epub-stuff/fg-spie.css"/>

<folder name="fg21"/>
<color fcolor="#238acb;" rcolor="#002395;"/>

<prelims>
  <file name="prelims/cover.xhtml"/>
  <file name="prelims/half-title-page.xhtml"/>
</prelims>

  <file name="spiebk-fg21-b01.xml"/>
  <file name="..."/>
  <file name="spiebk-fg21-r01.xml"/>
  ...
</files>
```

Figure 2: An example `hub.xml`

converted to ePub. The files are listed in the same order as they should be in the ePub. An example `hub.xml` is shown in Figure 2.

The source files are kept in another folder inside the working folder. In general, that folder has the same name as the project for which the ePub is to be generated.

All the source files can be either copied to the project folder or can be in different subfolders inside it. For example, one may create subfolders with chapter numbers and copy the figures and bitmapped equations of that particular chapter to that folder.

2.1.2 Process

To make the process simpler, we use the (GNU) `make`, a utility which executes commands grouped under a specific target in a file called `makefile` or `Makefile`. Separate targets are declared for each function. A single target that carries out the whole process is also available in the `Makefile`. For example,

```
make epub
```

will create an ePub, validate it and display an error log if there are any errors.

The resources of an ePub consist mainly of XHTMLs, graphic objects and several other auxiliary files. During debugging, if we have made any changes in the XHTML files directly, we need again to zip the files into an ePub format, and for this we run the command:

```
make zip
```

A complete list of our `Makefile` targets is in Table 1.

Target	Action
`file`	opens `makefile.in` to input the project id for which we need to create an ePub
`epub`	creates the ePub
`hub`	opens the hub file
`zip`	zips the files in an ePub format, assuming that all the files required for an ePub are available
`check`	validates the ePub
`renumber`	renumbers the IDs
`err`	opens the error log
`view`	opens the ePub in Lucidor (an ePub viewer)
`ncx`	opens `toc.ncx`
`opf`	opens `content.opf`

Table 1: List of targets in our `Makefile`

2.1.3 Files

The files `toc.ncx`, `content.opf` etc. mentioned in the table are generated through the XSLT style sheet. Some log files for debugging will also be generated.

3 Features

TEX to ePub through XML. The source file is a TEX file. This is converted into an XML file through an automated conversion process. The XML file generated conforms to Elsevier's book DTD. Since the primary source is TEX, TEX4ht [5] is used for TEX-to-XML conversion. During this process, one gets numerous opportunities to appreciate the power of TEX4ht and its highly configurable features for processing complex TEX documents into XML.

Conversion using XSLT. XSLT is the style sheet language recommended for XML and this is a declarative language used for the conversion of XML documents. We carry out the conversion to the ePub format from the XML using XSLT.

Minimal use of images. Except for complex math formulae, all the in-line math formulae are represented in ePubs using their HTML equivalents. For example, we can handle H_2SO_4, $E = mc^2$ etc., in HTML, whereas $\sqrt[n]{24}$ and similar formulae that do not have equivalent HTML are set as images.

Use of `dvipng`. For creating images of complex in-line and multi-line formulae, we use the application `dvipng` [6], and the images created look as beautiful as they are in the DVI.

Importing XHTML files. The real data for the ePub file comes directly from XML. However, if one has any other information (e.g., copyright pages, advertisements, call for papers etc.) which cannot be coded as XML due to DTD constraints, they can be used to create equivalent XHTML files and import them directly.

Compatibility. We have tried our best to create ePubs that are compatible with all e-book readers such as iPad, NOOK, Lucidor, Firefox etc.

Cross-platform solution. Since the conversion process uses TEX and XML technologies only, we can very well claim that this is a cross-platform solution.

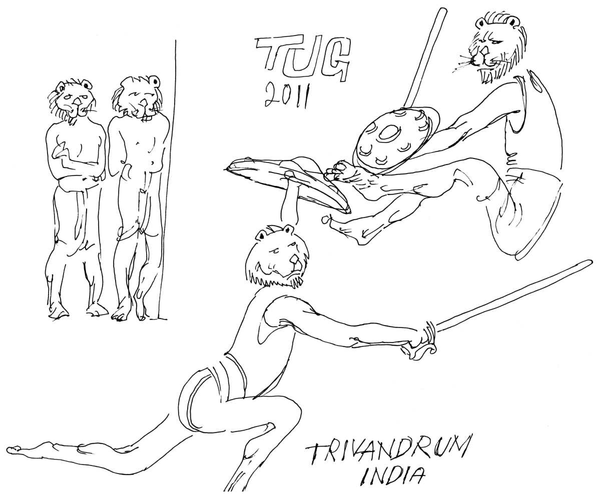
4 Challenges

Making the ePub compatible with different e-book readers posed some challenges.

References

[1] `http://en.wikipedia.org/wiki/EPUB`
[2] `http://en.wikipedia.org/wiki/XSLT`
[3] `http://ant.apache.org`
[4] `http://en.wikipedia.org/wiki/XML`
[5] `http://en.wikipedia.org/wiki/TeX4ht`
[6] `http://sourceforge.net/projects/dvipng`

◇ Rishi T.
River Valley Technologies
rishi (at) river-valley dot com
http://www.river-valley.com



A dream of computing and LaTeXing together: A reality with SageTeX

Manjusha Joshi

Abstract

Researchers search for some computational packages to find their results. At the time when they have the desired output, they begin worrying about how to insert it in their LaTeX document. They have to keep track of their output, formatting and inserting it at the appropriate places in the document.

The SageTeX package is a blessing in these situations. It calls the powerful open source maths server Sage, to compute and embed the result into a TeX document.

1 Sage: Math server

Sage [1] can handle numeric calculations, symbolic calculations, and plotting 2D–3D functions.

Sage is based on the scripting language Python. Sage is free and open source software which is intended to provide an alternative to proprietary programs such as Mathematica, Matlab, Maple and Magma.

1.1 Numerical outputs

Sage can handle multiprecision calculations.

```
factorial(100)
```

(100)! = 9332621544394415268169923885626670049
071596826438162146859296389521759999322991560
894146397615651828625369792082722375825118521
0916864000000000000000000000000

1.2 Symbolic calculations

Sage can also handle symbolic computations.

$$\int e^x x\,dx$$

can be computed in Sage using this command:

```
integral(exp(x)*x,x)
```

and then the output can be inserted in the TeX document like this:

```
$\sage{integral(exp(x)*x,x)}$
```

and the output will be generated like this: $(x-1)e^x$.

1.3 Graphics

Sage can produce graphics, both 2D and 3D. Fig. 1 shows a twisted torus generated in Sage.

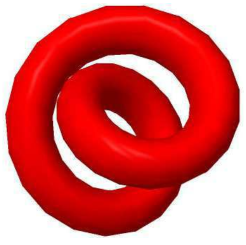


Figure 1: A twisted torus generated in Sage

2 Necessity of computed outputs in LaTeX?

Here are a few of the places where it can be useful to dynamically compute values from directly within a LaTeX document:

- In research papers
- In questions on exams
- Answers in solution sets in a book
- E-learning systems

3 Calculations from mathematical software

Researchers may need varies types of computations. Here is the list of related free software with the main focus on this job:

- R: Statistics, data handling
- Scilab: Numerical Computing
- Maxima: Symbolic Computing, Graph Theory
- GAP: Abstract Algebra
- Pari-gp: Number Theory
- Singular: Commutative Algebra

Sage can integrate any of these packages. The usual way of inserting results in a `.tex` file is by drawing it with some other software and inserting it as a figure.

Here is a figure of a vector field computed and drawn through the software Scilab [2]:

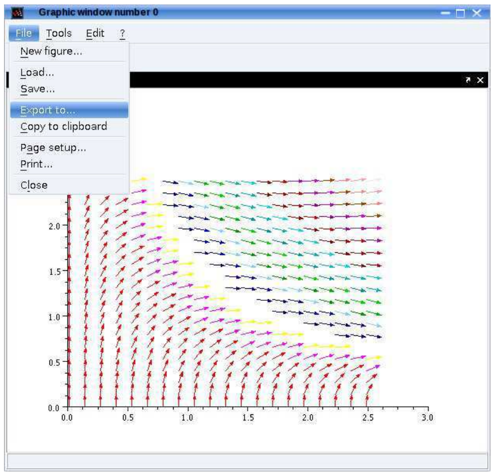
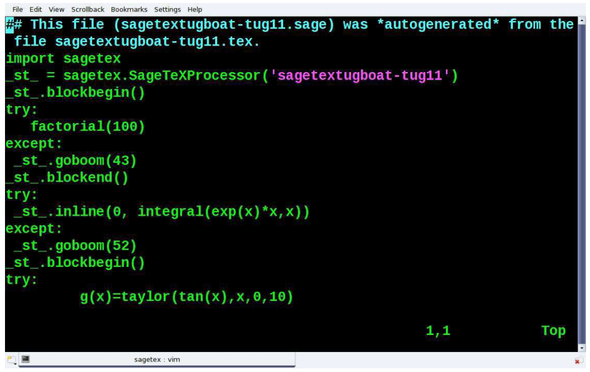


Figure 2: Exporting figure

4 Challenges

- While writing the paper first think of the places where the author needs to insert the computed outputs/figures.



```
## This file (sagetextugboat-tug11.sage) was *autogenerated* from the
 file sagetextugboat-tug11.tex.
import sagetex
_st_ = sagetex.SageTeXProcessor('sagetextugboat-tug11')
_st_.blockbegin()
try:
   factorial(100)
except:
 _st_.goboom(43)
_st_.blockend()
try:
 _st_.inline(0, integral(exp(x)*x,x))
except:
 _st_.goboom(52)
_st_.blockbegin()
try:
         g(x)=taylor(tan(x),x,0,10)

                                                       1,1           Top
```

Figure 3: Generated `.sage` file

- In each case one needs to switch between TEX and the software for computing, and search for the place in the TEX file where it has been decided to insert the outputs.
- While pasting calculations from other software, the author may miss some part of the answer. Loss of important calculated data is possible.
- Some files also may be lost, simply because the author may forget where the different output files have been stored, or because the software stores them in an unknown location.
- This laborious process does not help the author trying to compose the paper.

5 With SageTEX

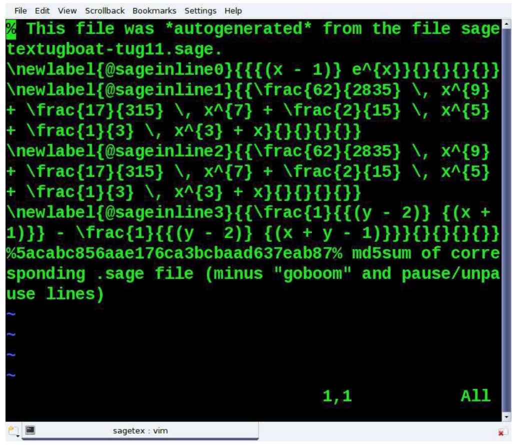
While writing research papers one can insert appropriate examples which can be calculated or drawn immediately by calling Sage directly from the LATEX compiler, and then be inserted at the appropriate places. Thus, one can maintain some orderliness in thinking.

Pictures, calculations become part of the `.tex` file. The author need not given special attention for picture insertion, etc. Also, it is not required to collect such files separately.

6 Work flow

To use Sage through LATEX, just use the package `sagetex` in the `.tex` file. (Of course you need to have Sage installed.) The sequence for compilation is as follows:

1. Start with your file `abc.tex` and `sagetex.sty`.
2. `pdflatex abc.tex` which contains the usual `\usepackage{sagetex}`.
3. This generates a `.sage` file in the same folder.
4. `sage abc.sage` to generate a `.sout` file. An example is shown in fig. 4.
5. `pdflatex abc.tex` once more.



```
% This file was *autogenerated* from the file sage
textugboat-tug11.sage.
\newlabel{@sageinline0}{{{(x - 1)} e^{x}}{}{}{}{}}
\newlabel{@sageinline1}{{\frac{62}{2835} \, x^{9}
+ \frac{17}{315} \, x^{7} + \frac{2}{15} \, x^{5}
+ \frac{1}{3} \, x^{3} + x}{}{}{}{}}
\newlabel{@sageinline2}{{\frac{62}{2835} \, x^{9}
+ \frac{17}{315} \, x^{7} + \frac{2}{15} \, x^{5}
+ \frac{1}{3} \, x^{3} + x}{}{}{}{}}
\newlabel{@sageinline3}{{\frac{1}{{(y - 2)} {(x +
1)}} - \frac{1}{{(y - 2)} {(x + y - 1)}}}{}{}{}{}}
%5acabc856aae176ca3bcbaad637eab87% md5sum of corre
sponding .sage file (minus "goboom" and pause/unpa
use lines)
~
~
~
~
                                      1,1           All
```

Figure 4: Generated `.sout` file

7 Sage commands

There are a few commands with which the document can communicate with Sage in variety of ways.

To insert only the output from Sage, use the command `\sage{...}`:

```
\sage{factorial(100)}
```

To plot a function through sage, the command is `\sageplot{...}`:

```
\sageplot{plot(exp(x),-5,5)}
```

It is also possible to display Sage commands in the `.tex` file as verbatim text and at the same time pass these commands to Sage. This is done with the `sageblock` environment:

```
\begin{sageblock}
g(x)=taylor(tan(x),x,0,10)
\end{sageblock}
```

After this, `g(x)` is known to Sage with the definition declared in the `sageblock`.

Then the TEX command `\sage{g(x)}` will pass the value of `g(x)` and the computed output will be inserted, as in:

```
$$\tan(x)=\sage{g(x)}$$
```

7.1 Predefined graphs

Sage predefines many graphs commonly used in graph theory. One can have complete graphs on a given number of vertices. Here is the famous Petersen graph (fig. 5):

```
    import sets
    g=graphs.PetersenGraph()
```

The LATEX command to display such graphs:

```
\sageplot{ g.plot().show() }
```

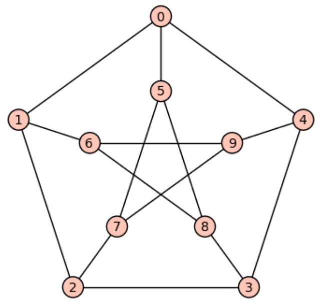


Figure 5: Petersen graph output.

7.2 Dynamic input to Sage

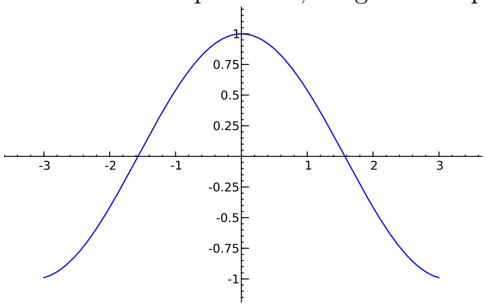
The input to Sage can be determined at the time of LaTeX compilation. For example, one can generate output from Sage after collecting input with `\typein`: the author can insert functions to be plotted at the time of compilation and plot the graphs. With this, at each compilation one can generate different graphs. This can be useful with online exam systems, since we want to generate different figures or randomly generated matrices in the exam paper.

As another example, one can have a collection of examples one of which is inserted randomly; required calculations and figure choice will be inserted on the fly at compilation time.

Here is an example of how one can get input and send it to Sage at the time of compilation:

```
\typein[\function]{Enter function name}
Here is the graph of $\function$:
\sageplot{plot(\function, -3,3)}
```

With the input ‘`cos`’, we get the expected:



8 Pregenerating results from Sage

For submission of a paper or book, it is not necessary for Sage to be installed on the publisher’s systems. The author should send the generated `.sout` file along with the `.tex` file for publishing.

9 Installation of sagetex

The package `sagetex` is available on CTAN [3], and also from `http://www.sagemath.org/`. Install it either in a `texmf` tree or in your working document folder.

If Sage is properly installed on the system and available then no additional settings are required. More installation information can be found at [4].

Also, one can use Sage remotely from the Internet while still using the `sagetex` package to compute and insert outputs. More details about this can be found at [5].

References

[1] `http://www.sagemath.org/`

[2] `http://www.scilab.org/`

[3] `http://mirror.ctan.org/macros/latex/contrib/sagetex`

[4] `www.sagemath.org/doc/installation/sagetex.html`

[5] `http://www.sagenb.org/`

◇ Manjusha Joshi
Pune, India
`manjusha dot joshi (at) gmail dot com`

Multi-target publishing

Axel Kielhorn

1 One road leads to one target

The usual target format of my documents was paper: ISO A4, ISO A5 or sometimes 3,5" × 5". My workflow led to an intermediate PDF file which was fine for reading on the screen, especially the smaller formats.

But then mobile devices appeared. The screen was too small to read A4 or even A5 documents. With some effort it was possible to create a document that was readable on *one* mobile device without excessive scrolling.

Having a format that reflows according to the size of the display with a user defined font size would be desirable. Such a format is ePub. It is simply a ZIP archive with a predefined structure and a few XML files that contain the actual content. A CSS file is used to control the appearance.

2 A detour

Luckily there is a program that reads LaTeX and writes ePub: Pandoc [5] (licensed under the GPL).

Unless the LaTeX file is too complicated, Pandoc will understand and convert it. But what is too complicated? The easiest way to find out is to convert a file from LaTeX to LaTeX and see what survives.

```
pandoc -r latex -t latex -o source-pd.tex
       source.tex
```

2.1 A rough road

Pandoc uses UTF-8 encoded files. This shouldn't be problem for most English speakers since they usually only use the first 127 characters of that encoding. But that is a naïve assumption. Even English speakers need non-ASCII characters for foreign words and punctuation characters. LaTeX offers many ways to enter these characters, but the only way that doesn't cause problems is to write them as UTF-8 characters. Thus `\^o` should be written as `ô` and `\o` as `ø`. A small few lines of `sed` will help with the conversion.

3 Back to square minus one

Is LaTeX really the starting point? Or should we see LaTeX as *one* backend and the LaTeX file just as an intermediate product?

4 An unusual direction: Markdown instead of markup

Markdown is a markup language developed by John Gruber [1] which looks as if no markup is present:

> A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions.

The start of this article originally looked like the following in Markdown:

```
# One road leads to one target

The usual target format of my documents was
paper: ISO A4, ISO A5 or sometimes 3,5"
$\times$ 5". My workflow led to an intermediate
file which was fine for reading on the screen,
especially the smaller formats.

But then mobile devices appeared.  The screen
was too small to read A4 or even A5 documents.
With some effort it was possible to create a
document that was readable on *one* mobile
device without excessive scrolling.
```

This text was created from the original with:

```
pandoc -r latex -t markdown -o Ziele-tug.md
       Ziele-tug.tex
```

Markdown is a very limited language. The man page describing the language has only 16 pages. The "Not So Short Introduction to LaTeX 2_ε" has ten times that number.

Converting from a complex language like LaTeX to a simple language like Markdown is difficult. Thus it is understandable that Pandoc only interprets a tiny amount of LaTeX markup. Since it doesn't understand TeX it uses regular expressions to parse the file. This will require additional empty lines in some cases where it is not required by TeX, otherwise the parser misses sectioning commands or environments.

Therefore it is best to convert a document to Markdown once and do all the future editing in Markdown.

5 A new road to an old target: Generating PDF from Markdown via LaTeX

```
pandoc -r markdown -t latex -o source.tex
       source.md
```

5.1 The `default.latex` file

The `default.latex` file distributed with Pandoc (in, e.g., `/usr/local/share/pandoc-X.Y/templates`) is a minimal example. With a little bit of LaTeX knowledge it can be customized to support the layout you need. A version adapted for German users is included in the supplementary material [2]. Modifications are marked with `-ak-`. A more elaborate file using the KOMA-Script class is included as well.

Editor's note: First published in *Die TeXnische Komödie* 3/2011, pp. 21-32; translation by the author.

To call Pandoc with a custom template, use the command line:

```
pandoc -r markdown -t latex
       --template=./custom.latex
       -o source.tex source.md
```

5.2 A Shortcut

The fastest way to turn a Markdown file into PDF is:

```
markdown2pdf --template=./custom.latex src.md
```

This will generate an intermediate LaTeX file and call pdfLaTeX to create the PDF.

With the options `--xetex` or `--luatex`, you can select a different engine. The template detects the engine and selects the appropriate code via `ifxetex` and `ifluatex`.

5.3 Postprocessing

The generated LaTeX file is surprisingly good. It matches files written by novice users.

Of course there may be some overfull and underfull hboxes that need further attention.

6 A new target ahead: ePub

The original desire was to create an ePub file in addition to the PDF file. The following command will do that:

```
pandoc -r markdown -t epub
       --epub-cover-image=cover-image.gif -s
       -o Source.epub Source.md
```

The text will be split into separate files according to the structure of the document. Thus it is easy to post-process the file with an ePub editor like Sigil [6].

Version 1.8.1.2 added the option to include a cover image (as shown above), thus reducing the need for post-processing.

7 The road to OpenOffice

"May I have this as a Word file?" Who doesn't know this question? Let's meet in the middle of the road with a LibreOffice file.[1]

```
pandoc -r markdown -t odt
       --reference-odt=./reference.odt -s
       -o source.odt source.md
```

The file `reference.odt` will be used as a template for the formatting of the document. If you want to change the design, you should modify the file supplied with Pandoc to make sure the internal style names match the ones used by Pandoc.

[1] `Writer2LaTeX` can convert LibreOffice files into LaTeX.

If you get an error when opening the `odt` file complaining about a corrupt file, you need to update Pandoc — a bug prior to version 1.8.1.3 led to the creation of invalid files when images were included.

Including images is still problematic. The images are in the final document, but they have to be rescaled.

8 Travel preparations

A small `sed` program removes some markup and converts LaTeX characters to UTF-8:

```
s/\\LaTeX/LaTeX/g
s/\\TeX/TeX/g
s/\\ConTeXt/ConTeXt/g
s/\\begingroup//
s/\\endgroup//
s/\\^o/ô/
s/\\o/ø/
```

Call this program on the command line with:

```
sed -f tex2mdtex.sed Source.tex
    >Source-clean.tex
```

The result can be converted to Markdown with:

```
pandoc -r latex -t markdown -s
    -o Source-clean.md  Source-clean.tex
```

9 Road signs

9.1 Sectioning commands

Markdown supports six hierarchy levels for sectioning commands. The number of # signs indicates the level. There has to be an empty line in front of the sectioning command.

```
# Top level

## Second level

### Third level

#### *Important information* hidden in
    the fourth level
```

An alternative form of sectioning commands only supports two levels:

```
First Level
===========

Second and last level
---------------------
```

9.2 Block Quotations

Markdown uses email conventions for quoting blocks of text. Lines starting with a `>` character are treated as block quotations.

```
> This is a block quotation
>
> > And this is a block quotation
> > inside a block quotation.
>
>
> The > sign is only needed in the first
line of the quotation.
```

A special kind of quotation is a quotation from a program. This is usually printed in a monospaced font. If a line starts with four spaces, it is treated as a verbatim text.

```
␣␣␣␣\documentclass[a4paper]{ltugboat}
␣␣␣␣\usepackage[utf8]{inputenc}
```

If you don't want to indent every line, you can use a delimited block, which begins with 3 or more tilde (~) characters and ends with at least the same number of tilde characters. If the code already contains a row of tilde characters, use more to delimit the quotation.

```
~~~~~~~~
This is a program listing
~~~~
Header preceded by tildes
~~~~
Body preceded by tildes
~~~~~~~~
```

9.3 Lists

There are several list types in Markdown that we already know from LaTeX:

9.3.1 The `itemize` list

The `itemize` list is started with a bullet character (*, + or -).

```
* one
* two
* three
        - three a
        - three b
* four
```

If a list entry contains several paragraphs, the paragraphs should be indented with four spaces or one tab.

```
* one
* two
* three
        - three a
        - three b
* four

    As usual, we hide important information
    in the fourth item.

    To be really sure, the 4 space rule is only
    mentioned in the last paragraph.
```

9.3.2 The `enumerate` list

An ordered list is like a bullet list, but it starts with an enumerator (`1.`, `(1)`, or `i.`) instead. The enumerators need not be in the correct order, even if that looks funny.

This kind of enumeration automatically loads the `enumerate` package to get custom enumerators. The generic enumerator `#.` uses the enumerators defined by the document class and avoids loading an additional package.

```
1. one
2. two
4. three
        a) three a
        b) three b
5. four
    Hiding important information ...
```

9.3.3 The `description` list

Sadly these animals from the German `lshort` haven't made it into the English version. Therefore I will introduce them here.

The term described is on a line of its own; the description follows in the next lines. The description is started with a colon or tilde, indented by one or two spaces. A term may have multiple descriptions, and each description may have one or more paragraphs.

```
Gelse
  : a small animal, living east of the
    Semmering, that chases tourists away.

Gemse
  : a large animal, living west of the
    Semmering, chased away by tourists.

    A long paragraph discussing
    whether it should be Gemse or Gämse.

Gürteltier
  ~ A medium sized animal. It only appears here
    because it has a long name.

  ~ In Austria, Gürteltiere are usually seen
    only in zoological gardens.
```

```
---------------------------------------------------------------------
 Centered      Default         Right    Left
 Header        Header        aligned    aligned
-------------  -------  -------------   ---------------------
First          row              12.0    Example of a row that
                                        spans multiple lines.

Second         row               5.0    Here's another one. Note
                                        the blank line between rows.
----------------------------------------------------------------------
```

Figure 1: A multiline table

9.3.4 Numbered lists with references

Usually a new list starts with number 1. If you want your items numbered throughout the document, Markdown offers a special list marker that is not reset. These list markers can be used as a reference later, or earlier. This is comparable to the caption counters used by LaTeX, but doesn't use the `\label`/`\ref` mechanism and does not require a second LaTeX pass.

```
(@Statement) Here I state something.

The statement (@Statement)
will be proved in (@Proof).

(@Proof) This is the proof.
```

9.4 Tables

Starting with version 1.8.1.2 Pandoc uses the `ctable` package to create tables. When entering tables it is best to use spaces instead of tabs to align the columns. There are three kind of tables.

A simple table:

```
  Right   Left     Center   Default
-------   ------  --------  -------
     12   12         12          12
    123   123       123         123
     ab   ab         ab          ab

Table: A simple table
```

The table header and the table rows must be written on one line. The alignment is defined by the dashed line below the header.

- If the dashed line is flush with the header text on the right side but extends beyond it on the left, the column is right-aligned.
- If the dashed line is flush with the header text on the left side but extends beyond it on the right, the column is left-aligned.

```
+------------+---------+----------------------+
| Fruit      | Price   | Advantages           |
+============+=========+======================+
| Bananas    | $3.14   | - built-in wrapper   |
|            |         | - bright color       |
+------------+---------+----------------------+
| Oranges    | $2.82   | - cures scurvy       |
|            |         | - tasty              |
+------------+---------+----------------------+
```

Figure 2: A grid table

- If the dashed line extends beyond the header text on both sides, the column is centered.
- If the dashed line is flush with the header text on both sides, the default alignment is used (in most cases, this will be left).

A table must be terminated by an empty line.

You can provide a caption starting with the string `Table:`, or just the character `:`. Any `Table` and the colon will be removed from the output. The caption may appear either before or after the table. When a caption is used, the table will be set in a `table` environment, otherwise it will appear in the body text.

Multiline tables allow headers and table rows to span multiple lines of text. The rows must be separated by empty lines. An example is shown in figure 1.

A grid table is shown in figure 2. The cells may contain arbitrary block elements, including lists.

9.5 Title

Information about the title, author and publication can be given at the beginning of the file.

```
% Multi-target publishing
% Axel Kielhorn
% TUGboat Volume vv
```

Long titles may be broken into several lines.

```
% Viele Ziele\
  (Multi-target publishing)
% Axel Kielhorn
  Babel Fisch (Trans.)
% TUGboat Volume vv
```

The \ in the first line will be translated to \\ in the LaTeX output.

9.6 Footnotes

A footnote consists of two parts, the footnote marker and the footnote text.

```
This is a footnote marker[^1]
and this is another footnote[^fussnote]

[^1]: Here is the footnote text

[^fussnote]: This footnote is slightly longer.

    It contains a second paragraph.
```

9.7 Inline formatting

Italic text is surrounded by *one* * or _. Bold text is surrounded by *two* * or _. If you want to emphasize only a part of a word, you have to use * because _ is often used as part of a name.

```
This text was emphasized _with underlines_
and this *with asterisks*.
For __bold text__ you need **two** characters.
```

Superscripts are surrounded by ^, subscripts by ~ characters:

```
H~2~O is water, 2^10^ is 1024.
2^2^^2^ is 2^22^.
```

The output of the last line may be unexpected, but note that these are *text* super- and subscript commands, not math commands.

9.8 Math(s)

Inline math is surrounded by $ characters. It is processed by LaTeX, thus everything allowed in LaTeX is permitted.

```
$2^{2^2} != 2^{22}$
```

When using a different output format, the result depends on the capabilities of that format.

Display math can be entered as raw LaTeX.

9.9 Raw LaTeX

Everything between a `\begin` and an `\end` will be copied verbatim to the LaTeX output and ignored in all other formats.

9.10 Raw HTML

Markdown was designed to create HTML. Therefore it is possible to include raw HTML, which will be ignored by non-HTML based formats.

9.11 Links

It is not surprising that a language designed to create web pages supports hyperlinks. Everything included between angle brackets is considered a link.

```
<http://johnmacfarlane.net/pandoc/>
```

A link may appear in a paragraph.

```
Documentation may be found on the [pandoc
web site](http://johnmacfarlane.net/pandoc/).
```

9.12 Pictures

A picture is included by providing a link to that picture and starting that link with a `!`.

```
![A blue picture](blau.jpg "Blue picture")
```

If the picture appears on a line of its own, it will be set in a `figure` environment and the text in the square brackets will be used as a caption text. Otherwise, it will be included in the body text.

```
The ![red square](rot.png "red square") appears
in the body text.
```

There is no way to scale pictures, they need to be in the correct size and resolution. This causes problems when the same picture is used for the web (72 dpi) and printing (300 dpi).

10 Next exit: ConTeXt

Pandoc is able to generate ConTeXt files. This is an easy way to convert LaTeX files to ConTeXt.

With a filter module, ConTeXt is able to directly process Markdown by calling Pandoc with parts of the document. See the Pandoc Extra Wiki [4] for details.

11 Large documents ...

The default settings for Pandoc is to create a document without section numbers and without a table of contents. This is fine as long as you write a short document, but when the size exceeds a few pages it would be better to have the sections numbered and a short overview over the contents:

```
pandoc -r markdown -t latex --number-sections
       -o md-test.tex md-test.md
```

numbers the sections, and with

```
pandoc -r markdown -t latex --toc
       -o md-test.tex md-test.md
```

a table of contents will be generated. Of course you can combine the options.

12 ... to full books

When you use the report, book, or memoir class, Pandoc will use the \chapter command as the highest sectioning level. It doesn't know about more exotic classes, like scrbook. If you want to use these, you have to request the chapters yourself:

```
pandoc -r markdown -t latex --chapters
       -o md-test.tex md-test.md
```

You can split the source text into several files and combine them when calling pandoc. This isn't as elaborate as the \include mechanism in LaTeX, but considering the speed of current computers, there is little need for \include/\includeonly. For a very long document you have to call Pandoc with a very long command line.

```
pandoc -r markdown -t latex
       --number-sections --toc
       --template=./report.latex
       -o md-test.tex
       md-test-intro.md
       md-test-ch1.md
       md-test-ch2.md
       md-test-ch3.md
```

13 Conclusion

With Pandoc, it is straightforward to create PDF files with LaTeX without knowing anything about LaTeX. All of the TeX processing can be hidden by using markdown2pdf and the various options in a shell script (or batch file). For example, I wrote an engine file for TeXShop that calls markdown2pdf instead of TeX. "That's too difficult" is no longer a reason not to use LaTeX.

Pandoc, or rather the Markdown language, has its limitations. If you need several kinds of foot- or endnotes, several bibliographies or lots of math, Pandoc is certainly underpowered. But if you don't need these features, Pandoc is an easy way to write structured documents without a high learning threshold.

The main work goes into the design of the template files, which should be created and maintained by a LaTeX expert.

14 Acknowledgements

This article originally appeared in the journal *Die TeXnische Komödie* [3], in German; the translation here is by the author. Some text used in this article is copied from the Pandoc–Markdown man page.

15 Appendix: Supplementary material

The following are included in the supplementary material you can download from the TUG server [2]:

md-test-tug.md The examples from this article.

md-test-tug.tex Converted LaTeX.

md-test.pdf Set with pdfLaTeX.

md-test.epub Converted to ePub.

tex2mdtex.sed A sed script to make general LaTeX documents palatable to Pandoc.

pandoc.pdf Pandoc man page.

pandoc-markdown.pdf Man page describing the Markdown syntax used in Pandoc.

markdown2pdf.pdf markdown2pdf man page.

engines A folder with engine files for TeXShop. (They will be in the next official release.)

◇ Axel Kielhorn
Lesumstraße 10
D-27283 Verden
Germany
A dot Kielhorn (at) web dot de

References

[1] John Gruber. *Daring Fireball: Markdown*, 2004. http://daringfireball.net/projects/markdown/.

[2] Axel Kielhorn. *Supplementary material*, 2011. http://tug.org/TUGboat/tb32-3/tb102kielhorn-supp.zip.

[3] Axel Kielhorn. Viele Ziele — Multi-Target Publishing. *Die TeXnische Komödie*, 3:21–32, 2011.

[4] John MacFarlane. *Pandoc Extras*, 2011. http://github.com/jgm/pandoc/wiki/Pandoc-Extras.

[5] John MacFarlane. *Pandoc — About pandoc*, 2011. http://johnmacfarlane.net/pandoc/.

[6] Strahinja Marković. *Sigil — A WYSIWYG ebook editor*, 2011. http://code.google.com/p/sigil/.

On the use of TeX as an authoring language for HTML5

S.K. Venkatesan

Abstract

The TeX syntax has been fairly successful at marking up a variety of scientific and technical literature, making it an ideal authoring syntax. The brevity of the TeX syntax makes it difficult to create overlapping structures, which in the case of HTML has made life so difficult for XML purists. We discuss S-expressions, the TeX syntax and how it can help reduce the nightmare that HTML5 markup is going to create. Apart from this we implement a new syntax for marking up semantic information (microdata) in TeX.

1 Introduction

The brevity of TeX syntax has made it fairly successful at marking up a variety of scientific and technical literature. On the one hand, modern markup languages such as (X)HTML and XML have verbose syntax which is not only difficult to author but also produces non-treelike structures such as overlapping structures that need to be checked for well-formedness. On the other hand, TeX and its macros are difficult to parse and validate, compared to XML with a DTD or schema. Many XML versions of TeX have been proposed such as TeXML [3] and XLaTeX [5] that are intrinsically close to (La)TeX. The main advantage of such a system is that one can introduce a validator using a DTD or schema to check the syntax before passing it to the TeX engine.

However, XML syntax is difficult to author and in fact is prone to producing overlapping structures that need to be avoided for it to be well-formed, and as a result these XML versions have not become popular for authoring. In this article, we propose something that is quite the reverse, i.e., TeX as an authoring syntax for both XML and HTML.

2 TeX, S-expressions and XML

Let us look at the following TeX code:

```
\title[lang=en]{Title of
  a \textit{plain} article}
```

The same code in a Lisp-like S-expression would be:

```
(title (@ (lang="en")) ("Title of a ")
  (italic "plain") ("article"))
```

or if one would like to treat elements and attributes in the same way:

```
(title (@lang="en") ("Title of a ")
  (italic "plain") ("article"))
```

The difference between the above two S-expressions is that the former introduces a deliberate asymmetry between attributes and elements, whereas the latter treats attributes on a par with elements. However, both S-expressions can be considered as an improvement on XML as they allow further nesting within attributes. The corresponding XML code would be:

```
<title lang="en">Title of
  a <italic>plain</italic> article</title>
```

In both TeX and XML syntax, further nesting of structures is not possible within attributes, which makes TeX ideal for authoring XML or HTML5.

There are further similarities between the TeX and SGML/HTML syntaxes. Attribute minimization used in HTML, like not quoting attribute values, is very much practiced in TeX syntax, more as a rule rather than the exception; e.g.,

```
\includegraphics[width=2cm]{myimage.gif}
```

Unlike SGML/HTML, TeX typically uses a comma as the separator between attributes, instead of the word-space used in SGML/HTML. TeX also uses complete skipping of attribute values, similar to the commonly used HTML code: `<option selected>`. Quite like TeX, HTML also has the practise of shrinking multiple spaces to a single space. All of these similarities make it clear that authoring HTML in TeX would be an ideal proposition.

3 Overlapping markup in HTML

Since HTML is marked up by humans, there tend to be many situations with overlapping elements or other eccentric markup which do not confirm to a well-formed SGML or XML syntax. Consider the HTML markup:

```
<p>Text with <i>unique <b>and</i>
strong formatting</b> issues</title>
```

A utility like HTML Tidy [6] or TagSoup [1] can convert this into well-formed markup such as:

```
<p>Text with <i>unique </i><b><i>and</i>
strong formatting</b> issues</title>
```

However, it is not always clear what should be done with such a non-standard markup. The HTML5 specification defines clearly how such a non-standard markup should be interpreted [7] but the HTML implementations in browsers currently deal with it differently from each other.

W3C has been insisting for some time that the next generation of markup should be XML-compliant like XHTML+MathML+SVG profiles, with other intricacies such as namespaces. However, more than 99% of HTML pages in the wild are invalid, according to the HTML4 DTD or schema. This being the case, W3C gave up on the idea of an XML solution and moved on to HTML5 with added elements and features, such as MathML, SVG and video, audio and additional microdata elements.

Given the experience with HTML4, it can be safely predicted that the more features one adds to HTML, the greater the scope for non-standard markup such as overlaps and entanglement that can create a great deal of difficulty for browsers and users.

We will consider here, e.g., Microsoft's interpretation of MathML in HTML5. Microsoft has been pushing for certain agenda in MathML3 (although I must say with great relief that much of it has not been accepted by the MathML committee). Based on their own experience with OML, a subset of OOXML markup, they would like to add formatting features in MathML such as bold, italic and paragraph elements inside MathML. Consider the following markup:

```
<math><b><mi>r</mi></b>=<mfenced><mi>x</mi>
  <mi>y</mi></mfenced></math>
```

the corresponding pure MathML coding would be:

```
<math><mi mathvariant="bold-italic">r</mi>
  <mo>=</mo><mfenced><mi>x</mi>
  <mi>y</mi></mfenced></math>
```

Mixing elements from different namespaces is one of the side effects one can expect in HTML5. It is not clear if MathML elements could be included within SVG elements or vice versa. One can expect such new non-standard markups to be created that will be quite difficult for browsers to handle.

New elements such as `<section>` have been introduced, so one can expect more confusion:

```
<section><h2>Section title</h2>
  <section><h1>Another section title</h1>
  </section>
</section>
```

The intended meaning of `<h1>` or `<h2>` is not clear from the above markup, and you could say either 'I mean what I say' or 'I say what I mean', with our own impressionistic interpretations.

In this article we do not want to convey the impression that everything about HTML5 is out of the wild west; rather, it is a rich arena that needs to be authored carefully, because there are so many pitfalls. In fact, HTML5 introduces new features like MathML, SVG, video and audio features that are essential for further enrichment of basic content [4]. The important reason for using a TeX-like system is that it doesn't allow one to see the output if there are errors in the code and one can only produce well-formed code.

4 TeX as an input format for HTML5

In this section we would like introduce LaTeX environment for authoring HTML5. Many of these features have been introduced before, say, e.g., in XLaTeX and other concepts.

4.1 Main structural elements of the document

HTML5 has introduced new content elements that bring it closer to the standard LaTeX classes. We propose the following TeX macros.

No.	HTML	LaTeX	Description
1	`<article>#1 </article>`	`\begin{article} #1 \end{article}`	article
			headings:
2	`<h1>#1</h1>`	`\Ha{#1}`	— level one
3	`<h2>#1</h2>`	`\Hb{#1}`	— level two
4	`<h3>#1</h3>`	`\Hc{#1}`	— level three
4	`<h4>#1</h4>`	`\Hd{#1}`	— level four
5	`<p>#1</p>`	`\p{#1}`	paragraph
6	`<span>#1</span>`	`\s{#1}`	text span

4.2 Simple formatting elements

We propose the following TeX macros for HTML formatting elements:

No.	HTML	LaTeX	Description
1	`<b>#1</b>`	`\B{#1}`	bold
2	`<i>#1</i>`	`\I{#1}`	italic
3	`<b><i>#1</i></b>`	`\BI{#1}`	bold-italic
4	`<tt>#1</tt>`	`\M{#1}`	text
5	`<sup>#1</sup>`	`\sp{#1}`	superscript
6	`<sub>#1</sub>`	`\sb{#1}`	subscript

4.3 MathML elements

We propose the following TeX macros for MathML formatting elements:

No.	MathML	LaTeX	Description
1	`<mrow>#1</mrow>`	`{#1}`	grouping
2	`<mi>#1</mi>`	`{#1}`	variables
3	`<mo>#1</mo>`	`{#1}`	operators
4	`<mn>#1</mn>`	`{#1}`	numbers
5	`<mtext>#1</mtext>`	`\mbox{#1}`	monospace
6	`<mfrac>#1#2</mfrac>`	`\frac{#1}{#2}`	fraction
7	`<msup>#1#2</msup>`	`{#1}^{#2}`	superscript
8	`<msub>#1#2</msub>`	`{#1}_{#2}`	subscript
9	`<mover>#1#2</mover>`	`{#1}^{#2}`	over
10	`<munder>#1#2</munder>`	`{#1}_{#2}`	under

No.	SVG	LaTeX	Description
1	`<circle cx="#1" cy="#2" r="#3" style="stroke:#4; stroke-width:#5;fill:#6;"/>`	`\circle[x=#1,y=#2,r=#3 s=#4,sw=#5,f=#6]`	circle
2	`<ellipse cx="#1" cy="#2" rx="#3" ry="#4" style="stroke:#5; stroke-width:#6;fill:#7;"/>`	`\ellipse[x=#1,y=#2,rx=#3, ry=#4,s=#5,sw=#6,f=#7]`	ellipse
3	`<rect x="#1" y="#2" width="#3" height="#4" style="stroke:#5; stroke-width:#6;fill:#7;"/>`	`\rect[x=#1,y=#2,w=#3, h=#4,s=#5,sw=#6,f=#7]`	rectangle

Table 1: Proposed TeX macros for SVG formatting elements.

No.	Microdata	LaTeX	Description
1	itemscope	`\s[is=on]`	top element that indicates descendants are in scope
2	itemtype	`\s[it=http:// data-vocabulary.org/Person]`	property URL
3	itemid	`\s[iid=p0312]`	unique ID of the person
4	itemprop	`\s[ip=name]`	name of the person
5	itemref	`\s[ir=http:// www.ctan.org/pub/article]`	reference URL

Table 2: Proposed TeX macros for HTML5 microdata.

4.4 SVG elements

We propose the TeX macros in table 1 for SVG formatting elements. These can be implemented using LaTeX graphics packages such as TikZ [2].

4.5 Microdata attributes

Since microdata (semantic) attributes can be added to any of the basic HTML elements, we need to be able to add attributes to any of the HTML5 TeX macros as well. Table 2 shows how these microdata attributes for `<span>` element are indicated using TeX macro `\s` defined in §4.1.

5 MuLTiFlow

We have created a WYSIWYG editor for authoring HTML5, released under the GPL v3 license. It can be installed either as a Firefox addon or as a standalone program. The project is hosted at `http://sourceforge.net/projects/multiflow` and is also available through the Firefox addon network. At present this editor uses HTML5 and UTN28 markup for authoring complex equations, but it will use the proposed TeX syntax for authoring HTML5 from version 1.1 onwards.

References

[1] John Cowan, TagSoup: A SAX parser in Java for nasty, ugly HTML, `http://home.ccil.org/~cowan/tagsoup`.

[2] Andrew Mertz and William Slough, Graphics with PGF and TikZ, *TUGboat* 28:1 (2007), 91–99, `http://tug.org/TUGboat/tb28-1/tb88mertz.pdf`.

[3] Oleg Parashchenko, TeXML: Resurrecting TeX in the XML world, *TUGboat* 28:1 (2007), 5–10, `http://tug.org/TUGboat/tb28-1/tb88parashchenko.pdf`.

[4] Mark Pilgrim, HTML5: Up and Running, Dive into the Future of Web Development, O'Reilly Media, 2010.

[5] John Plaice and Yannis Haralambous, XLaTeX, a DTD/schema which is very close to LaTeX, *TUGboat* 24:3 (2003), 369–376, `http://tug.org/TUGboat/tb24-3/haralambous.pdf`, `http://omega.enstb.org/xlatex`.

[6] Dave Raggett, HTML Tidy, `http://tidy.sourceforge.net`.

[7] W3C, HTML5 Working draft, `http://www.w3.org/TR/html5/introduction.html#syntax-errors`.

◇ S.K. Venkatesan
TNQ Books and Journals
Chennai, India
skvenkat (at) tnq dot co dot in

An XML model of CSS3 as an XƎLaTeX-TeXML-HTML5 stylesheet language

S. Sankar, S. Mahalakshmi and L. Ganesh

Abstract

HTML5 [1] and CSS3 [2] are popular languages for Web development. However, HTML with CSS is prone to errors and difficult to port, so we propose an XML version of CSS that can be used as a standard for creating stylesheets and templates across different platforms and pagination systems. XƎLaTeX [3] and TeXML [4] are some examples of XML that are close in spirit to TeX that can benefit from such an approach. Modern TeX systems like XƎTeX and LuaTeX use simplified `fontspec` macros to create stylesheets and templates. We use XSLT to create mappings from this XML-stylesheet language to `fontspec`-based TeX templates and also to CSS3. We also provide user-friendly interfaces for the creation of such an XML stylesheet.

1 Comparison of OpenOffice and CSS stylesheets

Nowadays, most modern applications have implemented an XML package format including an XML implementation of stylesheets: InDesign has its own IDML [5] (InDesign Markup Language) XML package format and Microsoft Word has its own OOXML [6] format, which is another ISO standard format. As they say ironically, the nice thing about standards is that there are plenty of them to choose from. However, instead of creating one more non-standard format, we will be looking to see how we can operate closely within current standards. Below is a sample code derived from OpenOffice document format:

```
<style:style style:name="Heading_20_1"
    style:display-name="Heading 1"
    style:family="paragraph"
    style:parent-style-name="Heading"
    style:next-style-name="Text_20_body"
    style:default-outline-level="1"
    style:class="text">
+<style:font-face
    style:name="Adobe Caslon Pro Bold"
    svg:font-family="'Adobe Caslon Pro Bold'"
    style:font-family-generic="roman"
    style:font-pitch="variable" />

<style:text-properties fo:font-size="115%"
    fo:font-weight="bold"
    style:font-size-asian="115%"
    style:font-weight-asian="bold"
    style:font-size-complex="115%"
    style:font-weight-complex="bold" />
</style:style>
```

```
-<style:style style:name="Heading_20_2"
    style:display-name="Heading 2"
    style:family="paragraph"
    style:parent-style-name="Heading"
    style:next-style-name="Text_20_body"
    style:default-outline-level="2"
    style:class="text">

<style:font-face style:name="Arial"
    svg:font-family="Arial"
    style:font-family-generic="swiss"
    style:font-pitch="variable" />

<style:text-properties fo:font-size="14pt"
    fo:font-style="italic"
    fo:font-weight="bold"
    style:font-size-asian="14pt"
    style:font-style-asian="italic"
    style:font-weight-asian="bold"
    style:font-size-complex="14pt"
    style:font-style-complex="italic"
    style:font-weight-complex="bold" />
</style:style>
```

The equivalent CSS style is listed below:

```
heading1{
          font-family: Adobe Caslon Pro Bold;
          font-size:14pt;
          font-style: normal;
          font-variant: normal;
          font-weight: bold;
          line-height: 16pt;
          text-align: left;
          color: black;
          background-color: none;
          text-decoration: none;
          text-transform: normal;
        }

heading2{
          font-family: Arial;
          font-size:14pt;
          font-style: italic;
          font-variant: normal;
          font-weight: bold;
          line-height: 16pt;
          text-align: left;
          color: black;
          background-color: none;
          text-decoration: none;
          text-transform: normal;
        }
```

When comparing the above two style coding standards, the Cascading Style Sheet (CSS) is a simple and straightforward formulation without complex namespaces, additional attributes and other details which are not mandatory to form a style.

1.1 Advantages of CSS style patterns

- CSS is simple to author.
- CSS makes it possible for the entire style and layout to be abstracted out of the HTML, so the HTML has only the content.
- Different stylesheets can be used for different media without the user having to explicitly choose one; e.g., printers, desktop monitors and other smaller portable devices.
- Implementing CSS is straightforward for the HTML engines.

1.2 Disadvantages of CSS style patterns

While new additions to CSS3 provide a stronger, more robust feature-set for layout, CSS is still at heart a styling language (for fonts, colours, borders and other decoration), and not a layout language (for blocks with positions, sizes, margins, and so on). These limitations mean that creating fluid layouts generally requires hand-coding of CSS.

2 Introducing the Cascading Style Sheet Markup Language (CSSML)

SASS [7] is a meta-language on top of CSS that is used to describe the style of a document clearly and structurally, with more power than flat CSS. SASS provides a simpler, more elegant syntax for CSS and also implements various features that are useful for creating manageable stylesheets.

SASS is an extension of CSS3 and provides several useful features which can handle nested rules, common variables, etc. However, it is not an XML model and cannot be validated using a DTD/schema.

We will introduce here the Cascading Style Sheet Markup Language (CSSML), an XML version of CSS that can be used as a standard for creating stylesheets and templates across different platforms and pagination systems. It is also an extension of CSS3 to handle nested rules as in SASS, and can be validated using DTD/schema.

We also hope CSSML will eventually evolve to circumvent all the current limitations in CSS and XSL-FO, especially the rules of placement of figures and tables in multi-column layout of text. However, we want to keep the CSSML as a clean data model by not introducing a scripting language on top of it as is the case with SASS.

3 Definition of CSSML

In general, our Cascading Style Sheet Markup Language (CSSML) specifies style format details in a well-structured XML format. The style names used in CSSML are similar to CSS; the only difference is that the style names are defined as XML tag elements. Here is an example to explain the difference between CSS and CSSML style coding:

```
  CSS: font-family: Adobe Caslon Pro Bold;
CSSML: <font-family>Adobe Caslon Pro Bold
       </font-family>
```

The main advantage of the CSSML tag pattern is that we can validate the CSSML document using XML Schema or XML DTD which is not possible in CSS. We can write our own XML DTD/schema to validate the CSSML document as follows:

- Elements and attributes that must/may be included, and are permitted in the structure.
- The structure as specified by a regular expression syntax.
- How character data is to be interpreted, e.g. as a number, a date, etc.

However, creating the CSSML document is not as simple as creating CSS. CSSML needs XML tagging for all the data, and the user needs to wrap all the details with appropriate XML elements. To avoid such difficulties, we have provided a user interface to create CSSML automatically with appropriate XML elements.

4 Namespaces

In this section we compare namespaces in CSSML, CSS, and XML.

4.1 CSSML

```
<styles xmlns="http://www.tnq.co.in/CSSML">
<namespace prefix="html"
           value="http://www.w3.org/1999/xhtml"/>
<namespace prefix="tux"
           value="http://www.tnq.co.in/TUX"/>
  <html:p>
   <style><text-color>yellow</text-color></style>
  </html:p>
  <tux:p>
   <style><text-color>blue</text-color></style>
  </tux:p>
</styles>
```

4.2 CSS

```
@namespace html "http://www.w3.org/1999/xhtml";
@namespace tux  "http://www.tnq.co.in/TUX";
html|p { display: block; color: yellow; }
tux|p  { display: block; color: blue; }
```

4.3 XML

```
<?xml version="1.0"?>
<?xml-stylesheet href="xml.css"?>
<article
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns:tux="http://www.tnq.co.in/TUX">
  <html:p>This is some text</html:p>
  <tux:p>This is another text</tux:p>
</article>
```

4.4 Sample CSSML coding

```
<styles xmlns="http://www.tnq.co.in/CSSML"
        xmlns:tux="http://www.tnq.co.in/TUX">
  <tux:section1>
   <style>
     <font-family>lmmono10-regular</font-family>
     <font-url>http://www.ctan.org/tex-archive/
         fonts/lm/fonts/opentype/public/lm/
         lmmono10-regular
     </font-url>
     <font-size unit="pt">11</font-size>
     <font-style>normal</font-style>
     <font-variant>normal</font-variant>
     <font-weight>Bold</font-weight>
     <font-face>lmmono10.otf</font-face>
     <line_height unit="pt">13.2</line_height>
   </style>
   <tux:section-label id="B12">
    <style>
      <text-indent unit="pt">6</text-indent>
      <rule-color>black</rule-color>
      <text-transform>normal</text-transform>
      <vertical-align>bottom</vertical-align>
    </style>
   </tux:section-label>
   <tux:section-title1 id="B13">
    <style>
      <text-indent unit="in">0</text-indent>
      <text-align>justify</text-align>
      <word-break>hyphenate</word-break>
      <column-span>all</column-span>
      <vertical-align>bottom</vertical-align>
    </style>
   </tux:section-title1>
  </tux:section1>
</styles>
```

5 CSSML to CSS3 conversion

CSSML provides a more elegant syntax for CSS and implements various features that are useful for creating stylesheets and LaTeX templates. CSSML allows us to use formatting, nested rules, inline imports, etc., all with CSS compatibility. XSLT is used to transform the CSSML XML to CSS format or to a LaTeX class file using the `fontspec` package.

5.1 Formatting

```
<tux:paragraph
     xmlns:tux="http://www.tnq.co.in/TUX">
 <style>
   <font-name>Times</font-name>
   <font-size unit="pt">11</font-size>
   <line_height unit="pt">13</line_height>
   <text-indent unit="pt">11</text-indent>
 </style>
</tux:paragraph>
```

```
@namespace tux "http://www.tnq.co.in/TUX";
tux|paragraph { font-family: Times;
                font-size:   11pt;
                line-height: 13pt;
                text-indent: 11pt; }
```

5.2 Nesting

```
<tux:section1>
 <style>
   <font-family>Times</font-family>
   <font-size unit="pt">11</font-size>
   <font-weight>Bold</font-weight>
   <line_height unit="pt">12</line_height>
 </style>
 <tux:section-label id="B12">
   <style>
     <text-indent unit="pt">6</text-indent>
   </style>
 </tux:section-label>
 <tux:section-title id="B13">
  <style><text-align>left</text-align></style>
 </tux:section-title>
</tux:section1>
```

```
@namespace tux "http://www.tnq.co.in/TUX";
tux|section1 { font-family: Times;
               font-size:   11pt;
               line-height: 12pt;
               font-weight: bold; }

tux|section1>tux|section-label
{ text-indent:6pt; }

tux|section1>tux|section-title
{ text-align:left; }
```

5.3 Selecting nodes

XPath is used to select nodes instead of CSS selectors. Here are some examples of CSS to XPath mappings:

CSS selectors	XPath pattern
`h1p`	`h1//p` (matches any `p` element that is a descendant of an `h1` element)
`h1>p`	`h1/p` (matches any `p` element that is a child of an element `h1`)
`p:first-child`	`*[1]/self::p` (matches element `p` when `p` is the first child of its parent)
`h1+h2`	`h1/following-sibling::*[1]/self::h2` (matches any `h1` element immediately preceded by an element `h2`)

The above table just provides a rough idea of how it is done, and by no means provides an exhaustive list of all CSS3 selectors.

6 CSSML to TEX font conversion

We use the `fontspec` package for font definitions. This package allows users of XƎTEX or LuaTEX to load OpenType fonts in a LATEX document. No font installation is necessary, and font features can be selected and used as desired throughout the document.

XƎTEX and LuaTEX also allow fonts to be loaded by file name instead of font name. When you have a very large collection of fonts, you will sometimes not wish to have them all installed in your system's font directories. In this case, it is more convenient to load them from a different location on your disk.

6.1 Font declaration example

In CSSML:

```
<font-group>
 <font-family>Times</font-family>
 <font-style-1>Times CG</font-style-1>
 <font-style-2>Times-Bold</font-style-2>
 <font-style-3>Times-Italic</font-style-3>
 <font-style-4>Times-BoldItalic</font-style-4>
 <font-style-5>Times-BoldSC</font-style-5>
</font-group>
```

Using `fontspec` in LATEX:

```
\fontspec[
  BoldFont = Times-Bold.otf,
  ItalicFont = Times-Italic.otf,
  BoldItalicFont = Times-BoldItalic.otf,
  SmallCaps = Times-BoldSC.otf,
]{Times.otf}
```

6.2 Paragraph style example

In CSSML:

```
<tux:paragraph>
 <style>
  <font-name>Times</font-name>
  <font-size unit="pt">11</font-size>
  <line_height unit="pt">13</line_height>
  <text-indent unit="pt">11</text-indent>
 </style>
</tux:paragraph>
```

Using `fontspec` in LATEX:

```
\def\normalsize{%
 \fontsize{11}{13}%
 \fontspec{Times}%
 \paraindent=11\p@
}
```

6.3 Section heading style example

In CSSML:

```
<tux:section1>
 <style test="section1">
  <font-family>Times</font-family>
  <font-size unit="pt">11</font-size>
  <line-height unit="pt">13</line-height>
 </style>
 <tux:section-label>
  <style test="section1">
    <font-variant>Bold</font-variant>
  </style></tux:section-label>
 <tux:section-title>
  <style test="section1">
    <text-align>center</text-align>
    <margin-top unit="pt">12</margin-top>
    <margin-bottom unit="pt">6</margin-bottom>
  </style>
 </tux:section-title>
</tux:section1>
```

In LATEX:

```
\newcommand\section{\@startsection
  {section}%
  {1}%
  {\z@}%
  {-12\p@ \@plus -2\p@ \@minus -2\p@}%
  {6\p@}%
  {\centering\fontsize{11}{13}%
   \selectfont\bfseries}%
}
```

References

[1] http://www.w3.org/TR/html5. A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, 25 May 2011.

[2] http://www.w3.org/TR/2011/REC-css3-selectors-20110929. Selectors Level 3. W3C Recommendation, 29 September 2011.

[3] http://omega.enstb.org/xlatex. A DTD/schema which is very close to LATEX.

[4] http://getfo.org/texml. An XML syntax for TEX.

[5] http://blogs.adobe.com/indesignsdk/category/idml. IDML for representing InDesign content.

[6] http://xml.openoffice.org/general.html. OpenOffice.org XML file format.

[7] http://sass-lang.com. Syntactically Awesome Stylesheets.

◇ S. Sankar, S. Mahalakshmi and L. Ganesh
TNQ Books and Journals
Chennai, India
sankar (at) tnq dot co dot in

Towards evidence-based typography: Literature review and experiment design

Boris Veytsman and Leyla Akhmadeeva

Abstract

During several centuries of typography many rules have been developed purporting to ensure better legibility and readability of printed copy. However, modern experimental research questions the absolute importance of these rules.

In this paper we provide a short review of the existing literature and discuss an experimental design for the work we are planning to perform.

1 Introduction

Typography is both a science and an art with several hundred years of history — or, if we count its ancestor, calligraphy, with several thousand years of history. A beginning typographer faces a large amount of knowledge and rules (see, e.g. [8]): for example, that serifed fonts improve readability of body texts, while sans serif is good for advertising and posters; that we know the optimal number of words per line and lines per page, etc. Some of these rules are æsthetic ones, while some are purported to reflect the neurophysiology of reading. With respect to the latter, we can ask, how do we know what we know? The fact that sometimes these recommendations are contradictory — even when offered by one great typographer (compare Tschichold in [36] and [37]!) — adds to the confusion.

The situation here may resemble the history of medical science (and art!). Centuries of practical medicine resulted in a vast number of rules and methods of cure (see a fascinating medical book of the 1600–1700s [16]). Some of them we now know to be reasonable, like the use of diuretics for lowering blood pressure. Some, like purging, have much narrower applicability than was assumed in the past. Some rules turned out to be ineffective or even harmful, like the unrestrained use of bloodletting. Modern evidence-based medicine tries to use a more scientific approach to these rules, putting empirical knowledge in a more formal framework [18].

In this talk we discuss the applicability of an evidence-based approach to typography. While it is difficult to measure the beauty of the book page, we can measure the readability and the understandability of the text and their dependence on the fonts, type area dimensions and other typographic parameters. This area has been actively developing in the last decade. The modern studies question the widespread notions of classical typography such as the use of serifed fonts [3,6,32], the mix of minuscule and majuscule letters in body texts [4,33], text layout [15,40], x-height [25] and other factors [14,27,35]. This research was stimulated by the challenges presented by new technologies [6,17,21,24,34], the use of type in messages and signage [12,19,20,38,39] and special situations like texts for low vision readers [2,4,32], drug information leaflets and other medical data [7,13,29].

An overwhelming majority of published studies deals with English texts, while there are some works on Arabic [1], Chinese [22], Japanese [5,21] and Korean [23] typography. We could find no comparable research on Cyrillic scripts and text perception by Russian readers.

Our group works on a large scale study of the neurophysiology of reading for Russian subjects. We plan to collect a database of readability and understandability as dependent on typographic parameters for Cyrillic texts. In this paper we provide the literature review and discuss the setup of the experiments.

2 The ecological hypothesis and its consequences

The easiest things to measure for the psychophysiology of reading are legibility [31] and readability [26]: the abilities to distinguish between the letters and to read words without errors. There are many studies that try to correlate these metrics with the typography of the text. Some of the results might be surprising for practitioners: for example, it seems that uppercase text is more readable than lowercase [4,33] and it is not clear whether serifs improve legibility or not [3,6,30]. One should keep in mind, however, that a font is a collection of features, and when one compares, for example, Times with Arial, one does not compare just a serifed font with a sans serifed one: many other features are different between these fonts, and the comparisons have too many confounding factors. It is interesting that one study [28] suggested the use of METAFONT to have a better control of font features for such comparisons.

The study of the influence of font size on the legibility and readability is more straightforward. In the recent work [25] a methodical study of such comparisons leads to the following result: legibility suffers when the fonts are too small (x-height smaller than about 4 pt) or too large (x-height larger than about 40 pt), but between these limits lies the “fluent reading range” where the ease of reading largely does not depend on the size. After studying fonts in the old and new copy the authors find that the most of it lies in this zone. Thus they formulate the *ecological hypothesis*:

> ...[F]luent reading is restricted to a broad but limited range of print sizes. The essential claim of our ecological hypothesis is that print sizes in most contemporary and historical publications fall within this fluent range [25].

The hypothesis is formulated for font sizes only. It has a quasi-Darwinian origin: a publisher that systematically makes copy outside of the fluent range would probably go out of business. Interestingly, there *are* some texts that are not meant to be read: for example, the (in)famous small print in legal contracts and drug inserts. In many cases this print is obviously outside the fluent range. This fact probably corroborates the ecological hypothesis.

While the authors of [25] do not discuss other typographic features, it seems reasonable to assume that they follow the pattern of font sizes: the typography of the historical and contemporary publications lies in the fluent range for a reader with normal or correctable vision.

Does this mean that typography does not matter at all?

Some experiments show that such a conclusion is unwarranted. Lewis and Walker [27] studied the perception of text as a function of the font it is typeset in. They used the standard (in psychology) technique of measuring the reaction time for signals: for example, a person is asked to press one button when she sees the word "strong" on the screen, and another button when she sees the word "weak". When the word "strong" appears in bold and "weak" in light weight, the reaction was significantly faster than in the opposite case. Experiments by Brumberger [9–11] show that the font influences the impressions about the author formed by the readers. This might show that typography might be important for other metrics of reading, besides readability or legibility. This might be very important because the aim of a book is not just to be read without errors: it should convey some message to the reader.

3 Proposed experimental setup

In the proposed experiment we study the influence of typography on the long term effect of texts on the example of Russian typography. The subjects are students of Bashkir State Medical University with normal or corrected vision, fluent Russian speakers. They are given short texts in Russian about history of neurology typeset with different fonts and layouts. We are going to use TeX for layout and METAFONT to change the font parameters.[1]

[1] There are several high quality free Cyrillic fonts in METAFONT format to enable such study.

The subjects are asked to read the texts and answer questions about them in writing. After this the texts and the answers are collected.

In one to two weeks the students are again asked to answer the questions about the texts. The difference between the number of errors in the first and the second battery of questions can be used as the metrics to study the influence of typography on the rate of long term effect of the texts.

4 Conclusions

The modern research has shown that some typographic rules of the past are evidently not grounded in legibility and readability requirements. It is still not quite clear, however, how and whether more subtle things such as text impression and long term effects depend on a work's typography.

We propose a study of how typography influences the way the text is remembered. Such study might be of interest to the publishers of textbook and study materials, especially for Cyrillic script.

References

[1] I. M. Al-Harkan and M. Z. Ramadan. Effects of pixel shape and color, and matrix pixel density of Arabic digital typeface on characters' legibility. *Int. J. Ind. Ergon.*, 35(7):652–664, July 2005.

[2] A. Arditi. Adjustable typography: An approach to enhancing low vision text accessibility. *Ergonomics*, 47(5):469–482, April 2004.

[3] A. Arditi and J. Cho. Serifs and font legibility. *Vision Res.*, 45(23):2926–2933, 2005.

[4] A. Arditi and J. Cho. Letter case and text legibility in normal and low vision. *Vision Res.*, 47(19):2499–2505, September 2007.

[5] M. Ayama, H. Ujike, W. Iwai, M. Funakawa, and K. Okajima. Effects of contrast and character size upon legibility of Japanese text stimuli presented on visual display terminal. *Opt. Rev.*, 14(1):48–56, January–February 2007.

[6] M. L. Bernard, B. S. Chaparro, M. M. Mills, and C. Halcomb. Comparing the effects of text size and format on the readibility of computer-displayed Times New Roman and Arial text. *Int. J. Hum.-Comput. Stud.*, 59(6):823–835, December 2003.

[7] L. Bix, H. Lockhart, S. Selke, F. Cardoso, and M. Olejnik. Is x-height a better indicator of legibility than type size for drug labels? *Packag. Technol. Sci.*, 16(5):199–207, September–October 2003.

[8] R. Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, Publishers, Vancouver, BC, Canada, 2004.

[9] E. Brumberger. The rhetoric of typography: Effects on reading time, reading comprehension, and perceptions of ethos. *Technical Communication*, 51(1):13–24, 2004.

[10] E. R. Brumberger. The rhetoric of typography: The awareness and impact of typeface appropriateness. *Technical Communication*, 50(2):224–231, 2003.

[11] E. R. Brumberger. The rhetoric of typography: The persona of typeface and text. *Technical Communication*, 50(2):206–223, 2003.

[12] P. J. Carlson and A. Holick. Maximizing legibility of unlit freeway guide signs with Clearview font and combinations of retroreflective sheeting materials. In *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings 2005*, number 1918 in Transportation Research Record, pages 26–34. National Academy Press, 2005.

[13] A. Chubaty, C. A. Sadowski, and A. G. Carrie. Typeface legibility of patient information leaflets intended for community-dwelling seniors. *Age & Ageing*, 38(4):441–447, July 2009.

[14] A. Coronel-Beltran and J. Alvarez-Borrego. Comparative analysis between different font types and letter styles using a nonlinear invariant digital correlation. *J. Modern Optics*, 57(1):58–64, 2010.

[15] M. dos Santos Lonsdale, M. C. Dyson, and L. Reynolds. Reading in examination-type situations: The effects of text layout on performance. *J. Res. Read.*, 29(4):433–453, November 2006.

[16] T. Dover. *The Ancient Physician's Legacy to his Country: Being what he has collected himself, in Fifty-eight Years Practice: Or, an Account of the several Diseases incident to Mankind, Described in so plain a Manner, That any Person may know the Nature of his own Disease, Together with several Remedies for each Distemper, faithfully let down, Designed for the Use of all Private Families.* Henry Kent, London, 1742.

[17] M. C. Dyson. How physical text layout affects reading from screen. *Behav. Inf. Technol.*, 23(6):377–393, November–December 2004.

[18] A. S. Elstein. On the origins and development of evidence-based medicine and medical decision making. *Inflamm. Res.*, 53 (Suppl. 2):S184–S189, August 2004.

[19] J. L. Gabbard, J. E. Swan, and D. Hix. The effects of text drawing styles, background textures, and natural lighting on text legibility in outdoor augmented reality. *Presence*, 15(1):16–32, February 2006.

[20] P. M. Garvey, K. N. Chirwa, D. T. Meeker, M. T. Pietrucha, A. Z. Zineddin, R. S. Ghebrial, and J. Montalbano. New font and arrow for national park service guide signs. In *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings 2004*, number 1862 in Transportation Research Record, pages 1–9. National Academy Press, 2004.

[21] S. Hasegawa, K. Fujikake, M. Omori, and M. Miyao. Readability of characters on mobile phone liquid crystal displays. *Int. J. Occup. Saf. Ergon.*, 14(3):293–304, 2008.

[22] D.-L. Huang, P.-L. P. Rau, and Y. Liu. Effects of font size, display resolution and task type on reading Chinese fonts from mobile devices. *Int. J. Ind. Ergonomics*, 39(1):81–89, January 2009.

[23] Y.-K. Kong, I. Lee, M.-C. Jung, and Y.-W. Song. The effects of age, viewing distance, display type, font type, colour contrast and number of syllables on the legibility of Korean characters. *Ergonomics*, 54(5):453–465, 2011.

[24] D.-S. Lee, K.-K. Shieh, S.-C. Jeng, and I.-H. Shen. Effect of character size and lighting on legibility of electronic papers. *Displays*, 29(1):10–17, January 2008.

[25] G. E. Legge and C. A. Bigelow. Does print size matter for reading? A review of findings from vision science and typography. *J. Vision*, 11(5)(8):1–22, 2011.

[26] G. E. Legge, D. G. Pelli, G. S. Rubin, and M. M. Schleske. Psychophysics of reading — I. Normal vision. *Vision Research*, 25(2):239–252, 1985.

[27] C. Lewis and P. Walker. Typographic influences on reading. *British J. Psychol.*, 80:241–257, 1989.

[28] L. Liu and A. Arditi. Apparent string shortening concomitant with letter crowding. *Vision Research*, 40(9):1059–1067, 2000.

[29] M. A. Mackey and M. Metz. Ease of reading of mandatory information on Canadian food product labels. *Int. J. Consumer Studies*, 33(4):369–381, July 2009.

[30] M. S. McCarthy and D. L. Mothersbaugh. Effects of typographic factors in advertising-based persuasion: A general model and initial empirical tests. *Psychology & Marketing*, 19(7–8):663–691, 2002.

[31] L. Reynolds. Legibility studies — their relevance to present-day documentation methods. *J. Doc.*, 35(4):307–340, 1979.

[32] E. Russell-Minda, J. W. Jutai, J. G. Strong, K. A. Campbell, D. Gold, L. Pretty, and L. Wilmot. The legibility of typefaces for readers with low vision: A research review. *J. Vis. Impair. Blind.*, 101(7):402–415, July 2007.

[33] J. E. Sheedy, M. V. Subbaram, A. B. Zimmerman, and J. R. Hayes. Text legibility and the letter superiority effect. *Hum. Factors*, 47(4):797–815, Winter 2005.

[34] I.-H. Shen, K.-K. Shieh, C.-Y. Chao, and D.-S. Lee. Lighting, font style, and polarity on visual performance and visual fatigue with electronic paper displays. *Displays*, 30(2):53–58, April 2009.

[35] M. V. Subbaram, J. E. Sheedy, and J. R. Hayes. Effects of font type, smoothing, and stroke width on legibility. *Invest. Ophthalmol. Vis. Sci.*, 45(Suppl. 2):4354, April 2004.

[36] J. Tschichold. *The Form of the Book. Essays on the Morality of Good Design.* Hartley & Marks, Point Roberts, Washington, 1991.

[37] J. Tschichold. *The New Typography.* University of California Press, Berkeley and Los Angeles, CA, 1998.

[38] B. R. Ullman, G. L. Ullman, C. L. Dudek, and E. A. Ramirez. Legibility distances of smaller letters in changeable message signs with light-emitting diodes. In *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings 2005*, number 1918 in Transportation Research Record, pages 56–62. National Academy Press, 2005.

[39] J. H. Wang and Y. Cao. A human factors study on message design of variable message sign. *Int. J. Ind. Eng. — Theory Appl. Pract.*, 10(4):339–344, December 2003.

[40] D. Wendt. Improving the legibility of textbooks — effects of wording and typographic design. *Vis. Lang.*, 16(1):88–93, 1982.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
`borisv (at) lk dot net`

◇ Leyla Akhmadeeva
Bashkir State Medical University
3 Lenina Str. Ufa, 450000, Russia
`la (at) ufaneuro dot org`

A comparative study of methods for bibliographies

Jean-Michel Hufflen

Abstract

First, we recall the successive steps of the task performed by a bibliography processor such as BibTeX. Then we sketch a brief history of the successive methods and fashions of processing the bibliographies of (LA)TeX documents. In particular, we show what is new in the LaTeX 2$_\varepsilon$ packages natbib, jurabib, and biblatex. The problems unsolved or with difficult implementations are listed, and we show how other processors like Biber or MlBibTeX can help.

Keywords Bibliographies, bibliography processors, bibliography styles, Tib, BibTeX, MlBibTeX, Biber, natbib package, jurabib package, biblatex package, sorting bibliographies, updating bibliography database files.

Introduction

As mentioned at the beginning of "'Bibliography Generation", the 13th chapter of *The LaTeX Companion*'s Second Edition [35], the items of a printed document's bibliography may be composed manually, but this method is not recommended, since the result may not be reusable within another context. Indeed, bibliography layouts are very diverse: a publisher may require that authors' names are written *in extenso* as far as possible, whereas another prefers for first names to be abbreviated using only initials, etc. So the best way to deal with bibliographies is the use of *bibliography database files*, containing the whole information about bibliographical items. These database files are searched by a *bibliography processor*, which builds 'References' sections for printed or online documents.

As we recall below, BibTeX [38] was unrivalled for a long time as the bibliography processor used in conjunction with the LaTeX word processor. Now the landscape is changing and other comparable programs have come out. So this article aims to focus on the directions taken by BibTeX and its possible successors. In [19], we compared the programming languages used to design *bibliography styles*, controlling bibliographies' layout. The present article's purpose is different: we are interested in the *evolution* of some successive bibliography processors used in conjunction with LaTeX, this evolution still being in progress. First, in Section 1, we delineate the tasks to be performed by such a bibliography processor. We also explain the requirements for how such a program should be updated. Then Section 2 sketches the features of these successive bibliography processors. Finally, a synthesis is given in Section 3. As mentioned above, the present article only covers bibliography processors used in conjunction with LaTeX, it is complemented by [25] about bibliography processors used in conjunction with ConTeXt [11], another format built out of TeX. Reading this article only requires basic knowledge about LaTeX and BibTeX. Of course, the short descriptions we give hereafter do not aim to replace the complete documentation of the corresponding tools. Readers interested in typographical conventions for bibliographies can consult [4, Ch. 10] and [7, Ch. 15 & 16].

1 Tasks of a bibliography processor

In this section, we summarise the tasks to be performed by a bibliography processor such as BibTeX. Along the way, we give the terminology used throughout this article. Then we point out the features that are still unsolved or with difficult implementations. Of course, a bibliography processor works in conjunction with a text processor such as LaTeX or ConTeXt, denoted by 'the word processor' in the following.

End-users can use bibliography database files, containing bibliographical *entries*. The main role of a bibliography processor is to extract the bibliographical *references* of a document from these entries. According to BibTeX's standard use, bibliographical entries (resp. references) are stored in .bib (resp. .bbl) files. Let us notice that in some documents, there is no 'References' section, but rather bibliographical references are given as footnotes wherever they are cited. In other words, bibliographical references exist as *resources* and are intended to be typeset — so the bibliography processor must build them as processable by the word processor — possibly as a section or sparsely. Sometimes there are several 'References' sections, because each chapter of an important book has its own bibliography, or a unique bibliography is divided into several rubrics.

When several bibliographical references are to be grouped into a section, some bibliographies are *unsorted*, that is, the order of items must be the order of first citations of these items throughout the document. In practice, most bibliographies are 'sorted', most often according to first the authors' names,[1] second the dates: in such a case, it is up to the bibliography processor to perform this sort operation. Let us mention that the 'standard' sort given above is not universal: we personally were in charge of

[1] . . . or editors' names, when there is no author, for example, for a conference's complete proceedings.

the publication list of our laboratory — the LIFC[2] — when the activity report was written according to the directives given by the AERES:[3] we had to sort this list first by research teams, second by *categories*,[4] third by years decreasingly, fourth by authors' names increasingly, fifth by months decreasingly.

Each bibliographical entry is supposed to be accessible from a *citation key*, that is, citation keys must be non-ambiguous. Source texts written by end-users only contain citation keys to point to bibliographical resources, whereas results typeset by the word processor deal with *bibliographical keys*. It is up to the bibliography processor to build a mapping between citation and bibliographical keys. These bibliographical keys depend on the *system* chosen; as an example, they are positive natural numbers in the number-only system. Sometimes, bibliographical keys are built from the first letters of authors' names, followed by the year and possibly by a letter; so does BibTeX's alpha bibliography style. In some other systems — e.g., the author-date or author-number system — some parts of an entry can identify it obviously: cf. [4, Ch. 10] or [35, Ch. 12] for a survey about these systems. Let us mention that the alpha bibliography style belongs to the number-only system, rather than the author-date one, because bibliographical keys are univoque — like natural numbers — and *atomic* in the sense that you cannot divide them into an author and year parts. In other words, they actually work like natural numbers, up to an isomorphism.

Given a document, rules governing the layout of bibliographical references and citations, including ordering bibliographical items in a 'sorted' 'References' section, comprise a *bibliography style*.

What we have expressed above could have been put down when BibTeX was designed and put into action, in the 1980s. Since that time, some additional requirements have appeared. First, the character encoding that was most commonly used at that time and for a long period was ASCII,[5] 7-bit based. Later, some 8-bit extensions — such as Latin 1 or Latin 2[6] — allowed some additional characters used in non-English languages to be included. Then a universal character encoding, Unicode [43], was designed, including some *formats*[7] — such as UTF-8 and UTF-16 — that allow the complete set of Unicode characters to be represented by byte or double-byte sequences. A modern bibliography processor should be able to deal with all these different encodings, in particular UTF-8, which is becoming more and more common. Another point related to multilinguism may be viewed as a particular case of *software localisation*. Let us give an example about person names: first names are usually put before last names in most languages written with the Latin alphabet; as a counter-example, that is not the case for the Hungarian language, where last names come first; a style suitable for bibliographies of documents written in Hungarian should take this point into account. Regarding the document's language, some information included in bibliographical entries should be included or discarded. For example, a transliteration of titles of works in Russian written with the Cyrillic alphabet may be of interest for a document in English, but would be useless for a document in Russian.

A modern bibliography processor should generate source texts for word processors that typeset documents to be printed — as mentioned above — but should also be able to build bibliographies for online documents. Besides, let us recall that for several years, the XML[8] metalanguage has become a central formalism for data interchange in general and for production process of documents in particular. In other words, a bibliography processor should be able to deal with languages using XML-like syntax — a good example is XSL-FO[9] — and languages for the Web, such as (X)HTML.[10]

Last but not least, a bibliography processor for (LA)TeX source texts should be able to deal with bibliography database (.bib) files written according to BibTeX's format, because of backward compatibility. As proof of this program's success, there is a *huge* number of such files in end-users' directories. However, that may be also viewed as *legacy*. Anyway, if another format for bibliographical entries was adopted, a converter from .bib files into this new format would be needed.

[2] ***L**aboratoire d'**I**nformatique de l'université de **F**ranche-**C**omté.*

[3] ***A**gence d'**É**valuation de la **R**echerche et de l'**E**nseignement **S**upérieur*, that is, 'agency evaluating research and university courses'.

[4] That is, articles in well-known international journals and in other international journals, articles in journals having 'national' scope, papers in conferences, etc.

[5] **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

[6] More details about these encodings can be found in [35, § 7.5.2].

[7] 'UTF' stands for '**U**nicode **T**ransformation **F**ormat'.

[8] E**X**tensible **M**arkup **L**anguage. Readers interested in an introductory book to this formalism can consult [41].

[9] E**X**tensible **S**tylesheet **L**anguage — **F**ormatting **O**bjects. This XML dialect aims to describe high-quality output prints. See [40] for an introduction to it.

[10] (E**X**tensible) **H**yper**T**ext **M**arkup **L**anguage. XHTML is a reformulation of HTML — the original language of Web pages — using XML conventions. [36] is a good introduction to these languages.

```
%A Mike Newton
%T Resurgence
%I Worldwide Library
%S Don Pendleton's Mack Bolan
%N 141
%D |APR| 2011
%K additional key
```

Figure 1: Example using the Refer format.

2 A little bit of history

2.1 Early attempt

As far as we know, the first bibliography processor was Refer, used with the troff[11] text processor, already present within the first versions of the Unix system [2]. As shown in Fig. 1, the Refer format for bibliographical entries — used within .ref files — is line-oriented. Unlike BibTeX's format, it does not provide explicit information about entry *types*, such as `article`, `book`, ... Such information is to be determined dynamically, when entries are processed. A similar bibliography processor for TeX source files using this Refer format has been developed: Tib [1]; it seems that Tib has been used mainly for *Plain TeX* source texts. Tib was written in C [28]; like Refer, it is a *preprocessor* in the sense that the source text Tib processes may contain *incomplete citations*, surrounded by '`[.`' and '`.]`':

```
... see [.mack bolan resurgence.], ...   (1)
```

and such citations are *replaced* by bibliographical keys within the result built by Tib:

```
... see \Lcitemark Newton\Citebreak
2011\Rcitemark, ...
```

'`.[]`' at the beginning of a line within the source text processed by Tib is replaced within the result by the source text for the 'References' section. Fig. 1's entry would appear inside such a section as shown in Fig. 2. It can be seen that such a reference uses some TeX commands introduced by Tib to memoize values associated with fields. Then the `\Refformat` command formats the complete reference. Tib provides some styles to customise this operation. Likewise, Tib allows citation keys to be numbers or alpha-like keys, and provides additional commands such as `\Lcitemark` and `\Rcitemark` to control citation keys' layout.

[11] Several steps are needed to make clear this name's etymology. One of the first text formatting programs was runoff — for 'I'll run off a document' — written in the mid-1960s. When this program was adapted in 1969, the new name was abridged in 'roff'. Further reimplementations of roff were called 'nroff' — for '**N**ewer **ROFF**' — and 'troff' — for '**T**ypesetter **ROFF**'. There is a modern version of this program, groff, provided by the GNU (**G**nu's **N**ot **U**nix) project. The groff package includes a new version of Refer.

```
{\Resetstrings%
 \def\Loccittest{}\def\Abbtest{}%
 \def\Capssmallcapstest{}\def\Edabbtest{}%
 \def\Edcapsmallcapstest{}\def\Underlinetest{}%
 \def\NoArev{0}\def\NoErev{0}\def\Acnt{1}%
 \def\Ecnt{0}\def\acnt{0}\def\ecnt{0}%
 \def\Ftest{ }\def\Fstr{10}%
 \def\Atest{ }\def\Astr{Mike Newton}%
 \def\Ttest{ }\def\Tstr{Resurgence}%
 \def\Itest{ }\def\Istr{Worldwide Library}%
 \def\Stest{ }%
 \def\Sstr{Don Pendleton's Mack Bolan}%
 \def\Ntest{ }\def\Nstr{141}%
 \def\Dtest{ }\def\Dstr{April 2011}%
 \Refformat}
```

Figure 2: Reference generated by Tib.

When Tib processes the argument of an incomplete citation, it truncates each word to 6 characters, and looks for a unique entry including all these truncated words as prefixes. For example, the entry given in Fig. 1 matches since it includes the three words '`mack`', '`bolan`', '`resurg`', given in (1). You can use keys at the lines labelled by `%K` — values associated with this field are not printed out in references — but in practice, many Tib users put authors' names and significant words of a title in incomplete citations. *Symbols* can be defined: in Fig. 1, we use this feature as a workaround that allows the month information to be put or not, depending on the value associated with the `APR` symbol. Let us notice that Tib can produce sorted bibliographies: when you call Tib, you can specify a first field for a primary sort key, a second field for a second sort key, and so on. However, only lexicographical sorts are possible: for example, `2` comes after `1999`! Of course, this point is not very important in practice because years coming from 'actual' bibliography database files are often close to each other; it is rare to include entries for documents written in the 1st and 20th centuries, but in such a case, the sort operation would fail.[12] In addition, let us recall that values associated with `%D` fields are supposed to be *dates*. From a theoretical point of view, some accurate encoding would allow complete dates — years, months, days — to be sorted but this operation is quite tedious and in practice, only years are used.

2.2 BibTeX's age

The first edition of the LaTeX manual included an introduction to BibTeX; [32, App. B] reads:

[12] This error disappears if '`2`' is replaced by '`0002`' in the values associated with the `%D` field. But that causes '`0002`' to be printed in the generated document processed by (LA)TeX, so it is an imperfect workaround.

```
@BOOK{newton2011,
        AUTHOR = {Mike Newton},
        TITLE = {Resurgence},
        SERIES = {Don Pendleton's Mack Bolan},
        NUMBER = 141,
        PUBLISHER = {Worldwide Library},
        TOTALPAGES = 320,
        MONTH = apr,
        YEAR = 2011}
```

Figure 3: Example using BibTeX's format.

Once you learn to use BibTeX, you will find it easier to let BibTeX make your reference list than to do it yourself. [...]

The BibTeX program was written in the WEB system, used to program TeX's kernel.[13] In fact, BibTeX was initially designed to work in conjunction with the SCRIBE word processor[14] [42]. That is why the markup of .bib files is introduced with an '@' sign: this convention originates from SCRIBE.[15] Fig. 3 is the specification, for BibTeX, of the entry given in Fig. 1 in the Refer format.

For many years, BibTeX has been intensively used and was unrivalled as the bibliography processor associated with LaTeX. In comparison with Refer or Tib, BibTeX is not a preprocessor: users keep the same source text before and after running BibTeX. To cite Fig. 3's entry, just put:

```
\cite{newton2011}
```

inside your source text, as mentioned in any LaTeX manual. In the typeset result, the 'References' section appears where a `\bibliography` command has been put down within the source text.[16] In fact, given a source (.tex) file, BibTeX never reads it, and only parses the corresponding auxiliary (.aux) file generated by LaTeX in order to store information about cross-references[17] [35, § 12.1.3]. BibTeX is a very robust program, very suitable for bibliographical entries concerning works written in English, and whose authors or editors have English or American names. As a proof that BibTeX is widespread, you can find a huge number of .bib files on the Web. In addition, there are many bibliography database management tools based on the .bib format, some graphical, as reported in [35, § 13.4] and [44, § 9.1].

[13] A recent description of the capabilities of this WEB system, related to *literate programming*, is [30].

[14] SCRIBE influenced LaTeX's design by introducing the notion of *document style*, the ancestor of the notion of LaTeX 2_ε's *document class*.

[15] ... and is still followed by Texinfo, the program used to typeset the GNU project's software manuals [6].

[16] ... in most of BibTeX's bibliography styles. A counter-example will be given in § 2.4.

[17] Let *f* be a file name without suffix, '`bibtex f`' is equivalent to '`bibtex f.aux`'.

```
\documentclass{article}
\usepackage{natbib}

\begin{document}
\citep{newton2011} is a thriller. The Albanian
Mafia is powerful, as mentioned by
\citeauthor{newton2011}.

\bibliographystyle{plainnat}
\bibliography{mb}  %  That is, Fig. 3.
\end{document}
```

Figure 4: Example using the natbib package.

```
\documentclass{article}
\usepackage{jurabib}
\jurabibsetup{titleformat=italic}

\begin{document}
\citetitleonly{newton2011} is a thriller. The
Albanian Mafia is powerful, as mentioned by
\cite{newton2011}.

\bibliographystyle{jurabib}
\bibliography{mb}
\end{document}
```

Figure 5: Example using the jurabib package.

As mentioned in [35, § 12.1.2], the number-only system is the default method supported by standard LaTeX, even if bibliographical keys may be identifiers, as in the alpha bibliography style. After some attempts [35, § 12.3.1], the author-date system has been implemented successfully by the natbib package — as well as a simplified version of the author-number system [35, §§ 12.3.2 & 12.4] — used in conjunction with accurate bibliography styles. In particular, this package provides an interface with references, in that some commands — e.g., `\citeauthor`, `\citeyear` — can get access to particular fields. An example is given in Fig. 4, and typesetting it looks like this:

[Newton, 2011] is a thriller. The Albanian Mafia is powerful, as mentioned by Newton.

A more complete interface is provided by the jurabib package [35, § 12.5.1], implementing the author-date and short-title systems. The example given in Fig. 5 results in:[18]

Resurgence is a thriller. The Albanian Mafia is powerful, as mentioned by Newton. (2)

Such an approach is possible since the `\bibitem` command's optional argument — giving a citation

[18] In fact, jurabib's `\citetitleonly` command, used in Fig. 5, uses the contents of the `SHORTTITLE` field if it is available, the contents of the `TITLE` field otherwise [35, pp. 719–720].

```
\bibitem[Newton(2011)]{newton2011} Mike Newton.
\newblock \emph{Resurgence}.
\newblock Number 141 in Don Pendleton's Mack
Bolan. Worldwide Library, April 2011.
\end{document}
```

Figure 6: Reference for the natbib package.

key [35, § 12.1.2] — is structured. This structure remains light for the natbib package (cf. Fig. 6), becomes heavy for the jurabib package (cf. Fig. 7).

As we can see in Fig. 7 about a reference built by a bibliography style suitable for the jurabib package, the text following a `\bibitem` command and its argument is marked up with LaTeX commands. In fact, such a reference inside a bibliography is not directly formatted by the jurabib bibliography style, but this operation is *deferred* to LaTeX, since these commands are defined within the jurabib package. For example, the `\bibtfont` command is used for books' titles.[19] As another example, the `\bibnf` command applies to five arguments representing the components of a person name — *in extenso* and abbreviated — and controls the layout of such a name. Here is the default layout of a reference built by the jurabib bibliography style:

> **Newton, Mike:** Resurgence. Worldwide Library, April 2011, Don Pendleton's Mack Bolan 141

Considering a name of an author, if you want the last name to be typeset using small capitals, followed by the first name surrounded by parentheses when the *von* and *Junior* parts are absent, just redefine the following commands:

```
\renewcommand{\biblnfont}[1]{\textsc{#1}}
\renewcommand{\bibfnfont}[1]{\textrm{#1}}
\renewcommand{\jbNotRevedNoVonNoJr}{%
 \biblnfmt{\jbLast} %
 (\bibfnfmt{\jbCheckedFirst})}
```

BibTeX has some drawbacks, even if they are solved by workarounds. In fact, since BibTeX has been mainly used for texts to be processed by LaTeX, users get used to put LaTeX commands inside values associated with BibTeX fields. That idea is quite good, but the problem is that BibTeX's conventions are not LaTeX's. As a simple example, you can write '`Pierre V\'{e}ry`' within a LaTeX source text, but you must put:

```
AUTHOR = {Pierre V{\'{e}}ry}
```

[19] As shown by Fig. 7, this `\bibtfont` command is used when the reference is typeset. If you want to customise the titles' layout when they appear throughout your text, use the titleformat option of the jurabib package or the `\jurabibsetup` command, as shown in Fig. 5.

```
\bibitem[{Newton\jbdy {2011}}{}%
 {{0}{}{book}{2011}{}{}{}{}%
  {Worldwide Library\bibbdsep {} 2011}}%
 {{Resurgence}{}{}{2}{}{}{}{}{}}%
 ]{newton2011}
\jbbibargs {\bibnf {Newton} {Mike} {M.} {} {}}
{Mike Newton} {au}
{\bibtfont {Resurgence}\bibatsep\ \apyformat
 {Worldwide Library\bibbdsep {} \aprname\ 2011}
 \numberandseries {141}{Don Pendleton's Mack
  Bolan}} {\bibhowcited} \jbdoitem
{{Newton}{Mike}{M.}{}{}} {} {} \bibAnnoteFile
{newton2011}
```

Figure 7: Reference for the jurabib package.

within a .bib file, especially if you would like to use the alpha bibliography style, as explained in [35, pp. 768–769]. Another example, given in [35, p. 767]:

```
AUTHOR =
  {Maria {\MakeUppercase{de} La} Cruz}
```

in order for the group '`De La`' to be recognised as the name's particle — that is, the *von* part, w.r.t. BibTeX's terminology — even though it does not begin with a downcase letter, as in BibTeX's conventions. In fact, this group's initial uppercase letter will be typeset by LaTeX by means of the predefined command `\MakeUppercase`. From a general point of view, inserting LaTeX commands inside such values is not recommended for .bib files shared by several users or put on the Web, especially if these commands belong to particular packages or should be user-defined. Besides, such commands may be misunderstood by other formats or programs related to TeX, e.g., ConTeXt or LuaTeX [12]. Finally, these commands complicate the conversion of .bib files into HTML pages.[20]

Of course, these drawbacks have appeared only recently in relation to the date of BibTeX's first version. However, a modern version of a format for bibliography database files cannot ignore them. As another drawback, BibTeX — like Tib — provides only lexicographical sorts, as reported in [18], though at least end-users can choose their own sort keys. Last but not least, BibTeX's bibliography styles (.bst files) are written using bst [37], an old-fashioned language using postfixed notations and based on manipulating a stack.

2.3 BibTeX's successors

The BibTeX program has remained stable for more than a decade. A new version (1.0) has been announced in [39], but is not available yet. Other

[20] An example of such a converter is BibTeX2HTML [9]. Other comparable tools are listed in [44, § 9.2].

programs, which have come out quite recently, may be viewed as BibTeX's successors, in the sense that they behave like BibTeX: they use .bib files, look into .aux files. Some aim to replace the bst language of BibTeX by another programming language, more modern.

2.3.1 Programs based on BibTeX

In this section, we consider the programs that do not actually replace BibTeX, because their source files are revisions of BibTeX's, or they need BibTeX when they run.

Whereas BibTeX's original version, written by Oren Patashnik, can only deal with ASCII texts, BibTeX8 [35, § 13.1.1] is a revision of BibTeX that allows end-users to store .bib files using 8-bit codes such as Latin 1 or Latin 2. However, you have to use the same encoding for all the .bib files parsed when you order BibTeX8 to run. In other words, you cannot build a 'References' section by using a .bib file encoded in Latin 1 and a second encoded in Latin 2. In addition, BibTeX8's capacity can be enlarged by means of options, whereas the same operation on BibTeX needs source files to be recompiled.

BibTeXu is another revision of BibTeX, compatible with UTF-8 and integrating sort routines coming from the ICU[21] library. BibTeXu is briefly described in [44, § 4.3].

We include Bibulus [46] in this section because Bibulus needs BibTeX whenever it runs. Bibulus includes the bib2xml bibliography style — written using the bst language — which converts the selected entries into an XML-like format. Then such XML files are processed by a program written using Perl.[22] Whereas BibTeX's bibliography styles written using bst are monolithic programs — identical parts are copied *verbatim* from a style onto another — the features of Bibulus' bibliography styles are controlled by arguments of the \bibulus command:

```
\usepackage{bibulus}
\bibulus{
 ignorevon,cite=alpha,punctuation=/,
 authorfont=\sc}
```

to be put in a LaTeX document's preamble.

2.3.2 Complete reimplementations

These programs aim to replace BibTeX, that is, provide the same service. They are 'complete' reimplementations in the sense that they do not use source files of the BibTeX program. Three allow a .bst

[21] **I**nternational **C**omponents for **U**nicode.

[22] **P**ractical **E**xtraction and **R**eport **L**anguage. A good introduction to this language is [45].

```
(book
 (@ (id "newton2011") ...)
 (author (name (personname (first "Mike")
                           (last "Newton"))))
 (title "Resurgence")
 (publisher "Worldwide Library") (year "2011")
 (month (apr)) (number "141")
 (series "Don Pendleton's Mack Bolan")
 (totalpages "320"))
```

Figure 8: SXML format used by MlBibTeX.

bibliography style to be run. BibTeX++ [8] and cl-bibtex [31] compile it to functions written using their implementation language, Java [26] for the former, ANSI[23] Common Lisp [10] for the latter. This is provided as a compatibility mode; users are encouraged to develop new bibliography styles using these implementation languages.

MlBibTeX[24] [14] is written using the Scheme functional programming language [27] and allows a .bst bibliography style to be interpreted [15]. You can write a bibliography style using Scheme, as we did in [23]; another choice is to use nbst,[25] a language close to XSLT,[26] the language of transformations used for XML documents [41, Ch. 6]. In fact, when MlBibTeX parses a .bib file, the result can be viewed as an XML document, formatted using SXML[27] conventions [29], as shown in Fig. 8. MlBibTeX provides syntactical extensions for .bib files, described in [14]. Most are related to MlBibTeX's features about multilinguism, others ease the specification of person names [16], as shown in Fig. 9.

There is a big difference between BibTeX and MlBibTeX: the latter performs a more precise analysis of .bib files. When a field name is not recognised, a warning message is emitted.[28] That may be viewed as an advantage: if you type '`EDITORS = ...`' instead of '`EDITOR␣ = ...`' inside an entry of type `@INPROCEEDINGS`, MlBibTeX will warn you whereas BibTeX will silently ignore that field. This feature may also be viewed as a drawback: if you specify a `MONTH` field, the associated value must be a symbol among `jan`, `feb`, ..., `dec`. Otherwise, MlBibTeX stops with an error message. This convention may appear as too restrictive,[29] but MlBibTeX can sort

[23] **A**merican **N**ational **S**tandard **I**nstitute.

[24] **M**ulti**L**ingual BibTeX.

[25] **N**ew **B**ibliography **ST**yle.

[26] e**X**tensible **S**tylesheet **L**anguage **T**ransformations.

[27] **S**cheme implementation of XML.

[28] ... but this is just a warning message; the corresponding information is not lost.

[29] If some information about the day of the month is relevant for an entry, some manuals — e.g., [35, § 13.2.3] — recommend to include it into the `MONTH` field. From our point of view,

```
@BOOK{cussler2000,
  AUTHOR = {Clive Cussler with Paul Kemprecos},
  TITLE = {Blue Gold},
  SERIES = {Numa},
  PUBLISHER = {Pocket Books},
  YEAR = 2000}

AUTHOR = {first => Maria, von => De La,
          last => Cruz}

AUTHOR = {Henry Rider Haggard, abbr => H. Rider}

AUTHOR = {John L White, abbr => J. L}

AUTHOR = {org => Word Wide Web Consortium,
          sortingkey => W3C}
```

Figure 9: Syntactical extensions for author names provided by MlBibTeX.

w.r.t. month names[30] whereas BibTeX's standard bibliography styles do not. To perform such an operation, month names must be recognised. Likewise, when years are to be sorted, MlBibTeX applies a numerical sort whereas Tib and BibTeX sort years as strings, as mentioned above. So the value associated with a `YEAR` field must be an integer;[31] otherwise, an error message is emitted.[32] Let us end with mentioning that MlBibTeX's library provides powerful functions for language-defined lexicographical sorts [17, 18] and numerical ones. In particular, MlBibTeX allows successive order keys to be *chained* easily.[33]

As mentioned above, MlBibTeX has been successfully used to process the bibliography of our laboratory's activity report. As explained in [21], this bibliography had to conform with very precise requirements that were not implemented in any bibliography style of BibTeX and would be tedious to program using BibTeX's language. MlBibTeX has been also used to populate the official French site for Open Archives, HAL,[34] as reported in [22, 23, 24]. MlBibTeX can handle (LA)TeX commands that produce accented letters, and it can deal with .bib files using Latin 1 encoding. A future version is planned with other encodings such as Latin 2 or UTF-8.

these two values — months and day numbers — are subject to sort. So mixing them seems to us to be bad technique, unless a precise format is defined, as done within the biblatex package's `DATE` field (cf. § 2.4). Another solution could be the introduction of a `DAY` field.

[30] This information is optional, but MlBibTeX addresses that by means of the function `<month-position`; the Scheme expression `(<month-position` *T* *`default-value`*`)` returns the month's rank if *T* is an SXML subtree containing month information, *`default-value`* if this information is not supplied. So this default value — which is an integer, in practice — allows us to sort any entry regarding this function's results.

[31] Negative values, for years BCE, are allowed.

[32] More precisely, the standard fields subject to additional check are `AUTHOR`, `EDITOR`, `YEAR`, `MONTH`, and `PAGES`.

[33] Let us consider a simple example with two person names whose last names are l_0, l_1, and first names are f_0, f_1. The expression:

`(<english?` l_0 l_1 `(lambda () (<english?` f_0 f_1`)))`

yields `#t` (resp. `#f`), the true (resp. false) value, if l_0 comes before (resp. after) l_1 w.r.t. the lexicographic order in English. If l_0 and l_1 are equal, the third argument is called. That is, if last names are equal, the comparison focuses on first names. This third argument of MlBibTeX's order relations is optional and defaults to `(lambda () #f)`, as for a strict order relation, that is, irreflexive. MlBibTeX's functions like `<english?` are case-sensitive by default — uppercase letters take precedence — another optional argument allows users to customise this behaviour.

```
@AUTHOR{mb141a, NAME = {Mike Newton}}

@CONFERENCE{tug,
  SHORTNAME = {TUG},
  LONGNAME = {{\TeX} Users Group Conference},
  [YEAR = 2008] ADDRESS = {Cork}, MONTH = jul,
  [YEAR = 2011]
     ADDRESS = {Trivandrum}, MONTH = oct}
```

Figure 10: Extensions recognised by CrossTeX.

CrossTeX [3] is written in Python [34] and implements a kind of object-oriented paradigm about bibliography database (.xtx) files. Such files look like .bib files, but everything is an object, defined by a key and some fields. More precisely, CrossTeX extends the cross-reference mechanism of BibTeX. For example, we can define an author as shown in Fig. 10 and use it to specify Fig. 3 more concisely:

```
@BOOK{newton2011, AUTHOR = mb141a, ...}
```

The initial entry type library of CrossTeX extends BibTeX's by other object

definitions such as `@AUTHOR`. Some objects are new entry types, such as `@PATENT`. Fig. 10 also gives an example of *conditional fields*, depending on a field's value. CrossTeX can be used as a replacement of BibTeX — in which case only basic BibTeX's bibliography styles have been implemented — and it also provides a converter from .xtx files into .bib ones and into pages using HTML.

2.4 The biblatex package

Let us recall that the .bbl files suitable for the jurabib package contain texts marked up with LaTeX commands (cf. Fig. 7), that is, formatting 'References' sections is deferred to LaTeX. The biblatex package and bibliography style [33] go further with this approach, as shown in Fig. 11. In this framework, BibTeX is used only to sort bibliographical items and generate labels. A variant of our example is shown in Fig. 12 using the biblatex package. The result looks like the same example with the jurabib package:

[34] ***Hyper-Article** en **Ligne***, that is, 'hyper-article on-line'.

```
\entry{newton2011}{book}{}
\name{author}{1}{}{%
  {{}{Newton}{N.}{Mike}{M.}{}{}{}{}}%
 }\list{publisher}{1}{{Worldwide Library}}
 \strng{namehash}{NM1}
 \strng{fullhash}{NM1}
 \field{number}{141}
 \field{series}{Don Pendleton's Mack Bolan}
 \field{title}{Resurgence}
 \field{month}{04}
 \field{year}{2011}
\endentry
```

Figure 11: Reference for the biblatex package.

cf. (2). We can notice commands interfacing the fields of a reference, such as:

`\citeauthor` `\citetitle` `\citetitle*`[35]

Only one bibliography style — biblatex — is used with the biblatex package; thus, the `\bibliographystyle` command is not given. The `\bibliography` command must be included in the document's preamble,[36] and just specifies the .bib files to be searched in order to build the bibliography.[37] Inserting the 'References' section within the document is done by the `\printbibliography` command.

If you would like to make a source text generated by biblatex.bst conform to a particular bibliography style, you can redefine some LaTeX commands introduced by the biblatex package. For example:

```
\renewcommand{\mkbibnamelast}[1]{%
 \textsc{#1}}
\DeclareFieldFormat[book]{number}{\##1}
```

The former allows last names of people to be typeset using small capitals; the latter puts a '#' sign just before the contents of the NUMBER field for a book. You can organise the elements of a particular entry type, e.g., `\DeclareBibliographyDriver{book}{...}`.

Such technique may lead to a great number of redefinitions; so a better method is to customise the layout of a bibliography by means of the biblatex package's *options*. In fact, using this package is

```
\documentclass{article}
\usepackage{biblatex}
\bibliography{mb}

\begin{document}
\citetitle*{newton2011} is a thriller. The
Albanian Mafia is powerful, as mentioned by
\citeauthor{newton2011}.

\printbibliography
\end{document}
```

Figure 12: Example using the biblatex package.

analogous to using Bibulus (cf. § 2.3.1), in the sense that a bibliography style is built by assembling its features as options of this package, according to the syntax '*key*=*value*'.[38] A rich library of styles is provided. For example:

```
\usepackage[style=authoryear,abbreviate=false,%
 uniquename=init,firstinits]{biblatex}
```

puts the author-date system into action, does not abbreviate keywords such as month names, assumes that author and editor names can be determined using last names only, and retains only the initial letters of authors' first names within the 'References' section. In fact, the style option is the union of two 'suboptions', e.g., 'style=authoryear' is equivalent to:

`bibstyle=authoryear,citestyle=authoryear`

bibstyle controls the layout of 'References' sections, whereas citestyle applies to citations throughout the document. The former (resp. latter) refers to a .bbx (resp. .cbx) file. Even when the inputenc package is used, you can handle .bib files encoded differently by specifying the bibencoding option for biblatex. You can also specify keys for the sort operation, by means of mnemonics: 'sorting=nyt' causes bibliographical items to be sorted w.r.t. authors' names, year, title. This option defaults to 'sorting=nty'. Notice that only ASCII code order is used for sorting if .bib files are searched by means of BibTeX. An unsorted bibliography is produced by 'sorting=none'.

Many additional fields are recognised, for example, SUBTITLE, for a work's subtitle, in addition to its title. You can use the standard fields YEAR and MONTH, or replace them by the DATE field, which allows the specification of *date ranges*:

`DATE = {2011-10-19/2011-10-21}`

The specification of fields recognised by biblatex use *types*: for example, AUTHOR is a *name list*, SUBTITLE and TITLE are *literals*. Some types are described by means of regular expressions, e.g., the *date* type.

[35] Within the biblatex package, the `\citetitle` command behaves like jurabib's `\citetitleonly` command (cf. Footnote 18), p. 292), whereas the `\citetitle*` command always puts the TITLE field's value, even if a short title is present.

[36] We used TeX Live 2010 for our examples. The manual of biblatex's next version [33, § 3.5.1] specifies that the `\bibliography` command has been deprecated and replaced by the `\addbibresource` command [33, § 3.5.1]. More generally, notice that biblatex's development is still in progress, so some details may be slightly out of date. The same remark is suitable about the Biber program (cf. § 2.5).

[37] In BibTeX's 'standard' bibliography styles, this command serves two purposes: specifying .bib files, and the place where the bibliography is to be typeset (cf. § 2.2). In some styles for which references are only typeset as footnotes, the `\nobibliography` command may be used instead [35, § 12.5].

[38] If a key *k* is given without a value, this is equivalent to '*k*=true'.

```
@BOOKINBOOK{robeson1983b,
        AUTHOR = {Kenneth Robeson},
        TITLE = {Death in Silver},
        BOOKTITLE = {Doc Savage #26-27},
        PAGES = {1-133},
        PUBLISHER = {Bantam Books},
        YEAR = 1983,
        MONTH = nov}
```

Figure 13: Nonstandard type recognised by biblatex.

Additional *entry types* can be handled, for example, @BOOKINBOOK, for items originally published as a standalone book and reprinted in collected works of an author (cf. Fig. 13). BibTeX's cross-reference mechanism has been extended [33, § 2.4.1].

Last but not least, biblatex's options encompass some features that have been implemented in separate packages. More precisely, the packages bibtopic, bibunits, chapterbib, and multibib [35, § 12.6], allowing multiple bibliographies within the same document are replaced by the environments refsection and refsegment [33, § 3.5] or by using *filters* [33, § 3.10]; similarly, the babelbib package [13], providing support for multilingual bibliographies should be replaced by the babel option [33, § 3.1.2].

2.5 The Biber program

Roughly speaking, if you use the biblatex package in conjunction with BibTeX, you go on using the latter for tasks it does not perform satisfactorily.[39] In particular, that is true about sorting, since BibTeX's sort procedures do not meet present requirements for multilinguism, as mentioned in § 1. So a new bibliography processor, Biber [5], written using Perl, has been developed. It aims to replace BibTeX for biblatex users;[40] it does not replace BibTeX wholly, since it only generates .bbl files for biblatex. The use of Biber is specified with the backend option of biblatex:[41]

```
\usepackage[backend=biber]{biblatex}
```

Such an order causes a .bcf[42] configuration file — using XML-like syntax — to be built. Let *f* be a file name without suffix, the command '`biber` *f*' is equivalent to '`biber` *f*`.bcf`'. An example of such a .bcf file is given in Fig. 15.

The Biber program is able to deal with the full range of the UTF-8 encoding as well as partial encodings such as Latin 1 or Latin 2. However, several encodings for several .bib files cannot be used for the same job, as in BibTeX8. biblatex and Biber are tightly coupled; some biblatex options are only available if Biber is used.

```
\DeclareSortingScheme{aeres}{
 \sort{presort}\sort[final]{sortkey}
 \sort[direction=descending]{
  \field{sortyear}\field{year}}
 \sort{
  \name{sortname}\name{author}\name{editor}}
 \sort[direction=descending]{
  \field{month}\literal{00}}}
```

Figure 14: Specification of an order relation for Biber.

For example, '`sorting=aeres`' allows you to use the successive keys given in Fig. 14 by means of the `\DeclareSortingScheme` command, usable only if the backend is Biber, and to be put in a document's preamble: this sorting scheme partly implements the order relation used to sort the bibliography for the LIFC's activity report, as mentioned in § 1.

In fact, biblatex may use additional fields for sorting: the first pass is controlled by the PRESORT field; some fields only used for sorting — such as SORTNAME, SORTYEAR, SORTTITLE — take precedence over the corresponding fields for 'actual' information — that is, AUTHOR or EDITOR, YEAR, and TITLE. In particular, this feature is useful when these fields are marked up — in which case some alternative information without markup can be used for sorting — or when a prefix of the title has to be dropped.[43]

The construct '`\sortname[final]{...}`' overrides all subsequent `\sort` commands if the corresponding field is available. As shown in Fig. 14, the sort process stops if the SORTKEY field is available. Within the `\sort` command's argument, the three commands `\name`, `\field` and `\literal` — they correspond with the types recognised by the biblatex package (cf. § 2.4) — give the fields to be considered in turn. We also see that the `\literal` command is used as a fallback when a field is not available: here, entries without MONTH information are to be placed after all the comparable entries with this information.

A language-dependent *collation* can be used by Biber for sorting: '`sortlocale=de`'. Like BibTeX, BibTeX8, and BibTeXu, the Biber program does not perform numerical sorts, it only sorts lexicographically. Sorts are case-sensitive by default, but can be case-insensitive. As in BibTeX, additional fields are ignored silently.

3 A point of view

Let us be honest: we cannot be fully objective since

[39] ... although some points are improved with BibTeX8.

[40] The Biber program is tightly coupled with the biblatex package: if you install both, pay attention to take compatible versions. See also Footnote 36, p. 296.

[41] The other values allowed for this option are `bibtex` (by default), `bibtex8`, `bibtexu`.

[42] **B**iblatex **C**ontrol **F**ile.

[43] This *modus operandi* is not specific to Biber, it is implemented within BibTeX's biblatex bibliography style.

```
<?xml version="1.0" encoding="UTF-8"?>
<bcf:controlfile version="0.9" xmlns:bcf="https://sourceforge.net/projects/biblatex">
  <bcf:options component="biber" type="global">     <!--  Biber options    -->
    <bcf:option type="singlevalued">
      <bcf:key>bibencoding</bcf:key><bcf:value>ascii</bcf:value>
    </bcf:option>     ...
    <bcf:option type="singlevalued">
      <bcf:key>inputenc</bcf:key><bcf:value>ascii</bcf:value>
    </bcf:option>     ...
  </bcf:options>
  <bcf:sorting type="global">                       <!--  Sorting spec    -->
    <bcf:sort order="1">
      <bcf:sortitem order="1" substring_side="left" substring_width="2">presort</bcf:sortitem>
      <bcf:sortitem order="2">mm</bcf:sortitem>
    </bcf:sort>
    <bcf:sort order="2"><bcf:sortitem order="1" final="1">sortkey</bcf:sortitem></bcf:sort>
    <bcf:sort order="3">
      <bcf:sortitem order="1">sortname</bcf:sortitem><bcf:sortitem order="2">author</bcf:sortitem>
      <bcf:sortitem order="3">editor</bcf:sortitem><bcf:sortitem order="4">translator</bcf:sortitem>
      <bcf:sortitem order="5">sorttitle</bcf:sortitem><bcf:sortitem order="6">title</bcf:sortitem>
    </bcf:sort>
    <bcf:sort order="4">
      <bcf:sortitem order="1">sorttitle</bcf:sortitem><bcf:sortitem order="2">title</bcf:sortitem>
    </bcf:sort>
    <bcf:sort order="5">
      <bcf:sortitem order="1">sortyear</bcf:sortitem><bcf:sortitem order="2">year</bcf:sortitem>
    </bcf:sort>    ...
  </bcf:sorting>
</bcf:controlfile>
```

Figure 15: Example of a .bcf file.

we are MlBibTEX's author. Nevertheless, we recall that this program has been successfully used to build the publication list of our laboratory's activity report [21], and to export this publication list to an open archive site [22, 23, 24]. This list approximately contained more than 500 items. We were able to detect all the typing mistakes in BibTEX's field names.[44] Moreover, during this work, we noticed that many entries being of type `@ARTICLE` included a `PUBLISHER` field. Of course, scientific journals are often published by a professional publisher, but BibTEX's standard bibliography styles do not deal with this information, which is ignored, pure and simple. Roughly speaking, the people who specified such information in .bib files — the editors of a conference's proceedings or a journal's publisher, these two mistakes have been reported many times — probably did not know that it would never be put down in any standard bibliography style, that is, they probably would never learn that by using 'old' BibTEX. So checking all the fields of an entry may be very useful and MlBibTEX is unrivalled for this task: it has the advantages and drawbacks of a non-permissive program. By the way, adding supplementary checks is easy.[45] Also, the executable file `mlbibtex2xml` allows MlBibTEX to be used as a converter from .bib files into XML-like ones.[46] In particular, that simplifies the production of HTML pages.

Anyway, we do not deny this program's drawbacks: it is currently unable to deal with encodings other than pure ASCII and Latin 1; we plan to improve that in the next version. Besides, the interface is rudimentary: on the one hand, MlBibTEX's kernel is highly customisable,[47] on the other hand, writing some functions using Scheme in order to assemble elementary parts is often needed, apart from standard

[44] For example, '`EDITORS`' instead of '`EDITOR`', as mentioned in § 2.3.2.

[45] For example, when we exported .bib files to HAL [22], a specific format was required for the `ADDRESS` field, and additional check was added easily.

[46] Metadata for HAL are expressed using an XML dialect [22]. To convert a .bib file into something suitable for HAL, first we produce an XML file according to our internal format, then a second step — the transformation into HAL's format — is delegated to an XSLT processor.

[47] Probably more so than Biber regarding the order relations used for sorting, including language-dependent order relations. The same, writing new functions to *label* references in .bbl files is easier.

use, in which case the executable file `mlbibtex` fits well. That can be viewed as an advantage: end-users can get the full power of a programming language. In fact, programming such *drivers* for MlBibTeX is not very difficult, though we admit that this may restrict the number of potential end-users.

An objective point is that BibTeX's successors have implemented many extensions, often incompatible. These numerous extensions are the proof that this activity domain — looking for some 'better BibTeX' — is productive. Anyway, some of these projects may not pursue the same goal: for example, CrossTeX aims to reduce information redundancy as far as possible by an inheritance mechanism, whereas MlBibTeX focuses on multilingual aspects.[48] But the same notion may be implemented differently. For example, some BibTeX bibliography styles use an additional field for the total number of a book's pages. Often this field is named `TOTALPAGES` — e.g., within the jurabib bibliography style — but the tools related to biblatex know this information as `PAGETOTAL`. Likewise, some styles deal with a `DAY` field, in addition to the fields `YEAR` and `MONTH`.[49] A namesake field is internally used by the biblatex package, but the bibliography style does not recognise it as an 'actual' BibTeX field within .bib files; it is not recognised by Biber, either. These examples — the fields `DAY` and `TOTALPAGES`, there are others — show that the .bib format should be refined regarding the modern tools dealing with such files. That could lead to some convergence among such extensions and allow end-users to experiment with more bibliography processors in compatible ways.

There is a big problem with refining the .bib format: is it sufficient to add new fields, or do we have to extend the syntax of values associated with fields? Obviously, the creators of biblatex and Biber, working in tandem, have chosen not to extend values' syntax, so a kind of meta-information has been included in configuration files, possibly redefined by end-users. For example, we give an extract of Biber's default configuration file — biber.conf — in Fig. 16. These three regular expressions[50] express that prefixes — sequences of two lowercase letters before a punctuation sign, as '`al-`' in '`al-Hassan`' — and diacritics — e.g., the '`ʽ`' sign in '`ʽΗσιοδος`' — are not considered when person names are sorted; also, the sequence '`The␣`' at the beginning of a title is to be ignored during the sort operation.

```
<nosort>
  type_name   \A\p{L}{2}\p{Pd}
  type_name   [\x{2bf}\x{2018}]
  type_title  \AThe\s+
</nosort>
```

Figure 16: Configuration of Biber (extract).

These examples are quite convincing since uncapitalised prefixes of person names are generally ruled out before sorting, the '`ʽ`' sign is only used in Greek[51] and does not have any influence on sorting; concerning the word '`The`', we do not know a language other than English where this word is used... but who knows, after all?[52] This information about prefixes to be dropped is handled by Biber in a language-independent way, which might be error-prone from our point of view. Anyway, here is another example: the abbreviation of first names. In French, *digraphs* should not be cut away, as shown in the first two examples:

Philippe ⟶	Ph.	*(French)*
Christian ⟶	Ch.	*(French)*
— ⟶	Chr.	*(German)*
Henry Rider Haggard ⟶	H. Rider Haggard	

The first name 'Christian' also exists in German and is abbreviated differently in this language.[53] The last example recalls that some person names retain the middle name when they are abbreviated. Even if some general information may be stored in general configuration files, it seems to us that we have to enrich .bib files' syntax in order to add such information about abbreviating first names.[54]

Finally, let us remark that MlBibTeX may be used as is with the biblatex bibliography style: an advantage is that .bib files with enriched syntax for person names can be used now with this style. Some adaptation of MlBibTeX could make it very suitable for this style — e.g., static check of the `DATE` field — and some adaptation of the bibliography style could take as much advantage as possible of MlBibTeX's

[48] However, we studied an extension of the cross-reference mechanism in order to specify translations without information redundancy [20]. This feature's implementation, planned for MlBibTeX's next version, is not finished yet; it uses a `TRANSLATOR` field, as in jurabib and biblatex.

[49] For example, the styles 'apa...' used by the American Psychology Association. See also Footnote 29, p. 294.

[50] The syntax of regular expressions used within the Perl language — Biber's implementation language — is briefly described in [45, Ch. 5].

[51] More precisely, this sign — the *rough breathing* — denoted an aspirate in ancient Greek and has disappeared in modern Greek since 1981.

[52] Accented versions of this word exist: '*thé*' for 'tea' in French, and in the Vietnamese name Hàn Thế Thành, as Karl Berry reminds me. The unaccented word might also exist in another language; who could affirm the contrary?

[53] In fact, German friends told us that this point is debatable. However, we personally saw this abbreviation in a German book.

[54] See Fig. 9 for how MlBibTeX addresses such information.

	Advantages	Drawbacks
BibTeX	Very stable; very robust; many .bib files in use.	Old-fashioned language for bibliography styles; lack of support for Unicode and multilinguism.
CrossTeX	Bibliography database files are more concise.	Lack of suitable bibliography styles.
Biber	UTF-8 supported; better backend for biblatex.	Quite slow, only usable with biblatex.
MlBibTeX	Useful check, possibly user-defined; powerful sort procedures; enriched syntax for person names.	Interface should be improved.

Table 1: Bibliography processors for LaTeX: our synthesis.

multilingual features. Likewise, an additional option '`[backend=mlbibtex]`' could take advantage of MlBibTeX's sort procedures.

4 Conclusion

The result of our synthesis is summarised in Table 1. As mentioned above, a bibliography processor usable with LaTeX should be able to deal with a huge number of extant .bib files. That is a kind of inertia: an efficient parser for .bib files is difficult to write because this syntax is old-fashioned. But recent implementations using various programming languages have shown that this difficulty can be overcome. Now the biblatex package seems to raise much interest within the LaTeX community. But this package shows that a bibliography processor more powerful than BibTeX is needed. Maybe biblatex will be the future standard for bibliographies typeset with LaTeX. Nevertheless, we think that the .bib format should be enlarged into a new format accepted by most of BibTeX's possible successors. So the extensions developed by these programs should remain compatible as far as possible. We personally plan to orient MlBibTeX's further development towards this direction.

Acknowledgements

Thanks to Barbara Beeton and Karl Berry for patiently waiting for and then proofreading this article.

◇ Jean-Michel Hufflen
LIFC — University of Franche-Comté
16, route de Gray
25030 Besançon Cedex, France
`jmhufflen (at) lifc dot univ-fcomte dot fr`
`http://lifc.univ-fcomte.fr/home/~jmhufflen`

References

[1] James C. Alexander: *Tib: A TeX Bibliographic Preprocessor*. Version 2.2. `mirror.ctan.org/biblios/tib/tibdoc.tex`. 1989.

[2] Steve R. Bourne: *The* Unix *System*. Addison-Wesley. 1983.

[3] Robert Burgess and Emil Gün Sirer: "CrossTeX: A Modern Bibliography Management Tool". *TUGboat*, Vol. 28, no. 3, pp. 342–349. In Proc. TUG 2007. 2007.

[4] Judith Butcher: *Copy-Editing. The Cambridge Handbook for Editors, Authors, Publishers*. 3rd edition. Cambridge University Press. 1992.

[5] François Charette and Philip Kime: *Biber: A Backend Bibliography Processor for biblatex. Version biber 0.9 (biblatex 1.6)*. August 2011. `http://biblatex-biber.sourceforge.net`.

[6] Robert J. Chassell and Richard M. Stallman: *Texinfo. The GNU Documentation System. Version 4.13*. `http://www.gnu.org/software/texinfo`. September 2008.

[7] *The Chicago Manual of Style*. The University of Chicago Press. The 14th edition of a manual of style revised and expanded. 1993.

[8] Fabien Dagnat, Ronan Keryell, Laura Barrero Sastre, Emmanuel Donin de Rosière and Nicolas Torneri: "BibTeX++: Towards Higher-Order BibTeXing". *TUGboat*, Vol. 24, no. 3, pp. 472–488. EuroTeX 2003, Brest, France. June 2003.

[9] Jean-Christophe Filliâtre and Claude Marché: *The BibTeX2HTML Home Page*. June 2006. `http://www.lri.fr/~filliatr/bibtex2html/`.

[10] Paul Graham: *ANSI Common Lisp*. Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs, New Jersey. 1996.

[11] Hans Hagen: *ConTeXt, the Manual*. November 2001. `http://www.pragma-ade.com/general/manuals/cont-enp.pdf`.

[12] Hans Hagen: "The Luafication of TeX and ConTeXt". In: *Proc. BachoTeX 2008 Conference*, pp. 114–123. April 2008.

[13] Harald Harders: "Multilingual Bibliographies: Using and Extending the babelbib Package". *TUGboat*, Vol. 23, no. 3–4, pp. 344–353. 2002.

[14] Jean-Michel Hufflen: "MlBibTeX's Version 1.3". *TUGboat*, Vol. 24, no. 2, pp. 249–262. July 2003.

[15] Jean-Michel Hufflen: "BibTeX, MlBibTeX and Bibliography Styles". *Biuletyn GUST*, Vol. 23, pp. 76–80. In *BachoTeX 2006 conference*. April 2006.

[16] Jean-Michel Hufflen: "Names in BibTeX and MlBibTeX". *TUGboat*, Vol. 27, no. 2, pp. 243–253.

TUG 2006 proceedings, Marrakesh, Morocco. November 2006.

[17] Jean-Michel Hufflen: "Managing Order Relations in MlBibTeX". *TUGboat*, Vol. 29, no. 1, pp. 101–108. EuroBachoTeX 2007 proceedings. 2007.

[18] Jean-Michel Hufflen: "Revisiting Lexicographic Order Relations on Person Names". In: *Proc. BachoTeX 2008 Conference*, pp. 82–90. April 2008.

[19] Jean-Michel Hufflen: "Languages for Bibliography Styles". *TUGboat*, Vol. 2008, no. 3, pp. 401–412. TUG 2008 proceedings, Cork, Ireland. July 2008.

[20] Jean-Michel Hufflen: "Specifying Translated Works in Bibliographies". *ArsTeXnica*, Vol. 6, pp. 93–97. In GUIT 2008 meeting. October 2008.

[21] Jean-Michel Hufflen : *Classe superreport — Manuel d'utilisation*. Mars 2010. `http://lifc.univ-fcomte.fr/home/~jmhufflen/superreport/superreport-readme.pdf`.

[22] Jean-Michel Hufflen: "Using MlBibTeX to Populate Open Archives". In: Tomasz Przechlewski, Karl Berry, Gaby Gic-Grusza, Ewa Kolsar and Jerzy B. Ludwichowski, eds., *Typographers and Programmers: Mutual Inspirations. Proc. BachoTeX 2010 Conference*, pp. 45–48. April 2010.

[23] Jean-Michel Hufflen : *Utilisation du convertisseur .bib ⟶ HAL*. Octobre 2010. `http://lifc.univ-fcomte.fr/home/~jmhufflen/superreport/`.

[24] Jean-Michel Hufflen: "From Bibliography Files to Open Archives: The Sequel". In: Karl Berry, Jerzy B. Ludwichowski and Tomasz Przechlewski, eds., *Proc. EuroBachoTeX 2011 Conference*, pp. 61–66. Bachotek, Poland. April 2011.

[25] Jean-Michel Hufflen: "Bibliography Tools and ConTeXt/LuaTeX". To appear in Proc. ConTeXt meeting 2011. September 2011.

[26] *Java Technology*. March 2008. `http://java.sun.com`.

[27] Richard Kelsey, William D. Clinger, and Jonathan A. Rees, with Harold Abelson, Norman I. Adams iv, David H. Bartley, Gary Brooks, R. Kent Dybvig, Daniel P. Friedman, Robert Halstead, Chris Hanson, Christopher T. Haynes, Eugene Edmund Kohlbecker, Jr, Donald Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy Lewis Steele, Jr, Gerald Jay Sussman and Mitchell Wand: "Revised^5 Report on the Algorithmic Language Scheme". *HOSC*, Vol. 11, no. 1, pp. 7–105. August 1998.

[28] Brian W. Kernighan and Dennis M. Ritchie: *The C Programming Language*. 2$^{\text{nd}}$ edition. Prentice Hall. 1988.

[29] Oleg E. Kiselyov: *XML and Scheme*. September 2005. `http://okmij.org/ftp/Scheme/xml.html`.

[30] Donald Ervin Knuth and Silvio Levy: *The CWEB System of Structured Documentation*. Addison-Wesley, Reading, Massachusetts. 1993.

[31] Matthias Köppe: *A BibTeX System in Common Lisp*. January 2003. `http://www.nongnu.org/cl-bibtex`.

[32] Leslie Lamport: *LaTeX: A Document Preparation System*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1986.

[33] Philipp Lehmann: *The biblatex Package: Programmable Bibliographies and Citations. Version 1.6*. 29 July 2011. `http://ctan.org/pkg/biblatex`.

[34] Alex Martelli: *Python in a Nutshell*. 2$^{\text{nd}}$ edition. O'Reilly. July 2006.

[35] Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LaTeX Companion*. 2$^{\text{nd}}$ edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[36] Chuck Musciano and Bill Kennedy: *HTML & XHTML: The Definitive Guide*. 6th edition. O'Reilly & Associates, Inc. October 2006.

[37] Oren Patashnik: *Designing BibTeX Styles*. February 1988. Part of the BibTeX distribution.

[38] Oren Patashnik: *BibTeXing*. February 1988. Part of the BibTeX distribution.

[39] Oren Patashnik: "BibTeX 1.0". *TUGboat*, Vol. 15, no. 3, pp. 269–273. September 1994.

[40] Dave Pawson: *XSL-FO*. O'Reilly & Associates, Inc. August 2002.

[41] Erik T. Ray: *Learning XML*. O'Reilly & Associates, Inc. January 2001.

[42] Brian Keith Reid: *SCRIBE Document Production System User Manual*. Technical Report, Unilogic, Ltd. 1984.

[43] The Unicode Consortium: *The Unicode Standard Version 5.0*. Addison-Wesley. November 2006.

[44] Herbert Voss: *Bibliografien mit LaTeX*. Lehmans Media, Berlin. 2011.

[45] Larry Wall, Tom Christiansen and Jon Orwant: *Programming Perl*. 3$^{\text{rd}}$ edition. O'Reilly & Associates, Inc. July 2000.

[46] Thomas Widman: "Bibulus — a Perl/XML Replacement for BibTeX". *TUGboat*, Vol. 24, no. 3, pp. 468–471. EuroTeX 2003, Brest, France. June 2003.

The `hletter` class and style for producing flexible letters and page headings

Brian Housley

Abstract

A package, *hletter*, is presented which permits the user to specify easily, with the aid of self-defined key-words, letters (with a logo and/or private) and headings. The *heading* may include a footer and the letter provides commands to include a scanned signature, two signees and works with the *merge* package. It illustrates using zero width boxes and converting lengths into counts.

1 Introduction

Your first thoughts are probably "Not another LaTeX letter package" but, maybe, this package does offer something extra and useful. The idea came from my secretary who wrote the minutes of various committee meetings, prepared regulations in three languages, wrote letters on behalf of the committees, etc. The objective was, at first, to have one package which would produce headers in the various languages for the departments, committees, etc., and the letter was an easy extension. Of course, since she is a LaTeX fan, she should also have the possibility of writing private letters (for me as well). The main ideas for the package are:

- Permit the user to specify key-words which, together with the default or specified language, invoke various styles of the heading.
- With letters one may define an option to produce a private letter, i.e., one with no logo but a from-address.
- The header is always centred, at the top of A4 paper.
- Ensure the to-address is centred in a C5/C6[1] window envelope.
- Use a style file to produce headings as for letters with a horizontal rule underneath.
- The text for the heading together with the footer is produced by key-words dependent on a user defined option.
- A command `\closingtwo` may be used to produce letters with two signees.
- The *merge* package by Graeme McKinstry [3] works.
- A scanned signature may be used — which is especially useful with *merge* letters.

[1] I would have supported the North American stationery sizes but I have no access to such envelopes, etc.

2 The general design

The files used are shown in figure 1 where the shaded files should be provided by the user. The package loads the packages *graphicx* and *ifthen*.

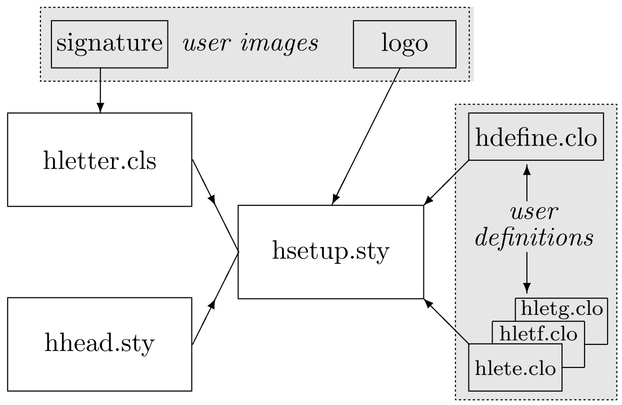


Figure 1: Files used in producing letters and headings

The function of the files are:

`hletter.cls` The class definition file, based upon the standard LaTeX letter class [2]. It redefines various commands and defines new ones (see later).

`hhead.sty` The package to produce the headings at the top of a page. Include `\usepackage{hhead}` in the preamble and the command `\heading` is defined to produce the heading(s).

`hsetup.sty` The file which does most of the work and defines the command to produce the headings and which reads in the files `hdefine.clo` and `hlet`⟨*lng*⟩`.clo` where *lng* is specified in the class or style options (default is English).

`hdefine.clo` The user file which defines key-words for the various headings.

`hlet`⟨*lng*⟩`.clo` The user file which defines the fields for the heading where *lng* is the letter e, f or g for the languages English (actually British), French and German.

`logo` The image file to produce the logo.

`signature` A scanned signature which may be used in the letter(s).

3 Fields used in the header

Figure 2 shows the commands which define the text where the command is shown. Also there is a command `\centrepos{`n`}` where n is a length specifying the offset of the centre text from the middle of the paper. The default is 10 mm and it may be negative.

If a header alone is being produced then it will have a horizontal rule below of a default width of 180 mm. With the command `\barlength` one may

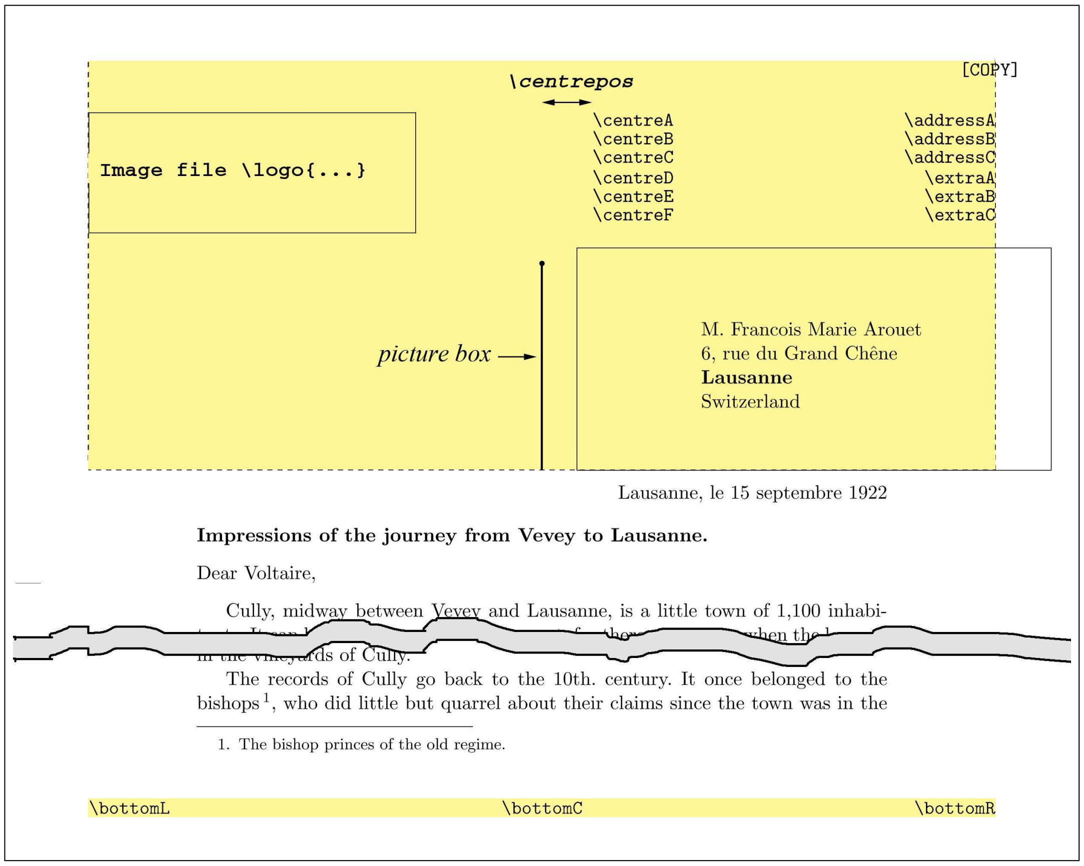

[COPY]
\centrepos
Image file \logo{...}
\centreA
\centreB
\centreC
\centreD
\centreE
\centreF
\addressA
\addressB
\addressC
\extraA
\extraB
\extraC
picture box
M. Francois Marie Arouet
6, rue du Grand Chêne
Lausanne
Switzerland
Lausanne, le 15 septembre 1922
Impressions of the journey from Vevey to Lausanne.
Dear Voltaire,
Cully, midway between Vevey and Lausanne, is a little town of 1,100 inhabi-
in the vineyards of Cully.
The records of Cully go back to the 10th. century. It once belonged to the
bishops 1, who did little but quarrel about their claims since the town was in the
1. The bishop princes of the old regime.
\bottomL
\bottomC
\bottomR

Figure 2: How most of the fields are defined

change this length, even making it 0 mm. If the logo is very high then the header height will be increased accordingly.

4 The layout of the header

Obviously the header for a letter is different from a simple header but both are produced using the *picture* environment and in both cases the origin of the picture has to be the same.

The header must be in the centre of the paper and the offset from the beginning of the text is calculated when the heading is produced. Thus any dimension changes the user may make are taken into account.

4.1 Horizontal positioning

The solution is to space horizontally and then make a LaTeX *picture* of zero width as shown in figure 3.

4.2 Vertical positioning

For letters the header stretches to the bottom of the *to-address* box (for a C5/6 envelope) and is 91 mm from the top of the paper. For the simple header (using the package *hhead*) the bottom of the header

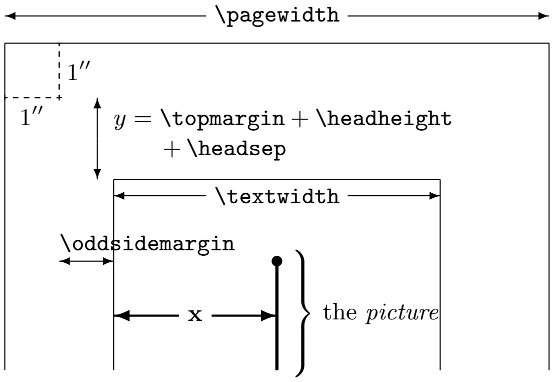


Figure 3: We see that $\mathbf{x} = .5$`\pagewidth` $- 1''$ $-$ `\oddsidemargin`

is 41 mm from the top of the paper but this may be increased if the logo is large.

4.2.1 The letter

As seen in figure 3 we need to calculate $h = 91\text{mm} - 1'' - y$ and if this value is negative then a warning "top margin seems to be too large" is issued. This can only happen if the text area is lower than the *to-address* box.

The variable h is a length variable and is stored as scaled points but for the picture we need a counter which depends on `\unitlength`. Thank goodness, TeX is very accommodating and we set a *counter* to the length h and then divide by `\unitlength`. The value is truncated but I think a header to within 1 mm is sufficiently accurate and one could modify the package to use a *unitlength* of 0.1 mm if one wishes more accuracy.

The command `\begin{picture}(0,h)(0,-41)` is used to produce the picture containing the header.

4.2.2 A simple header

Here the value calculated is $h = 46\text{mm} - 1'' - y$ and again we divide by *unitlength*. If the height of the logo is large then the value of the offset of the rule under the header is increased and the picture must be higher and the lower left of the picture is set to a negative value.

If the document is in *twocolumn* format then the command `\twocolumn` is used to ensure that the header spans the two columns.

5 The user files

`hdefine.clo` Defines the names to select the various type of headings, together with a sequentially increasing integer. An example is:

```
\logo{GCCS}
\newoption{private}{1}
\newoption{signit}{2}
\newoption{bruni}{3}
\newoption{test}{4}
```

As shown, the logo may also be specified in this file to provide a default which may be changed in the *hlet* files. The file `hsetup.sty` simply defines a new option which, if used, sets a global counter:

```
\newcommand*{\newoption}[2]{%
  \DeclareOption{#1}%
  {\global\hltype=#2}%
  \typeout{*** Option #2 has name #1}}
```

and types out the option and value in the log file. Originally the package generated the number automatically but early users wanted to specify the numbers themselves and cut and paste the define file as comments in the next files.

`hlet`⟨*lng*⟩`.clo` For each of the languages English, French and German which are supported (one could add more) the user must provide a file which defines the fields for the options used in `hdefine.clo`. The structure is shown in figure 4.

```
% Letter options for English
\ifcase\hltype
% case = 0 (no user option)
  definitions for default case
\or % case = 1 (private)
  \address{...
    defining an address gives a private letter
    ...}
\or % case = 2 (signit)
  definitions for signit option
\or % case = 3 (bruni)
  definitions for bruni option
\else
  % all other cases (should not be used)
  \addressA{?} \addressB{?} \addressC{?}
  \extraA{Telephone: ?}
  \extraB{Telefax: ?} \extraC{eMail: ?}
\fi
```

Figure 4: Structure of definitions file for English in `hlete.clo`

the logo The command `\logo[ht]{`⟨*file*⟩`}` sets the logo file. If the optional height is not specified, 24 mm is used. This command may be used in the definition file and/or in the *hlet* file(s).

signature file A scanned signature may be inserted; particularly useful for merge letters. Define the file with the command `\sign[ht]{`⟨*file*⟩`}`. If *ht* is not specified, it will be 15 mm high.

6 Creating a letter

Assuming that the define file and the *hlet* files have been created, one makes a letter in the usual LaTeX way but with a few additional commands. The class *hletter* is used with options for the point size, language (default English) and maybe one of the user options defined in `hdefine.clo` to select the required letter type.

6.1 Short summary of the letter commands

`\signature` The single argument is the name under the closing signature. Separate multiple lines with `\\`.

`\address` The *from-address* and, when used, makes a private letter without a logo. Separate multiple lines with `\\`.

`\reference` If used the argument is set centred under the opening for English and above, left justified, otherwise.

`letter` **environment** Starts the letter and the argument is the *to-address*.

`\date` The date to be printed under the header.

`\opening` This command has an optional argument which, when used, is placed in typewriter font at the top right of the letter, e.g., `\opening[{[DRAFT]}]{Dear Voltaire,}`.

`\closing` The argument is the closing text above the signature. Terminate multiple lines with `\\`.

`\closingtwo` Supplies the closing text which is centred above two signatures. The `\signature` command should contain two names, each line separated with an '`&`' as in a tabular (which it is), e.g.:

```
\signature{Dr.~A. Boss & Mr.~B. Bitt
           \\ CEO & CIO}
\closingtwo{Yours Faithfully,}
```

`\encl` A list of enclosures; multiple lines separated with `\\`.

`\cc` A list of persons who are to receive copies of the letter; multiple lines separated with `\\`.

7 Creating simple headings

In the document prologue one loads the package *hhead* with any optional argument such as language and a user option. A header is produced with the command `\heading`, which has an optional argument which if used will be printed at the top right of the page. If *heading* is used more than once in a document then a *cleardoublepage* is issued and the page count is reset.

8 Merge or form letters

The package *merge* from Graeme McKinstry works well with this letter package. It reads a file of {*to-address*, *opening*} pairs which are used to create a letter which is addressed to many recipients. When TeX reads from an external file it honours grouped lines; i.e., to enter the address over many lines in the merge file (new lines terminating with `\\`) enclose the address in `{...}`. The package uses a tabular to set the to-address so these brackets, if present, must be removed. Fortunately *The TeXbook* [1] (as usual) provides the answer and the *to-address* is produced with these, at first look, rather strange commands:

```
\def\dotoaddress#1{%
  \setbox0\hbox{\expandafter\cmda#1}%
  \ifnum\myc=1\settoaddress{#1}\else
  \expandafter\settoaddress#1\relax\fi}
\def\settoaddress#1{\global\setbox\addrbox
  \hbox{\begin{tabular}{@{}l@{}}#1\end{tabular}}}
\newcount\myc
\def\cmda#1{\global\myc=0 \cmdb#1\end}
\def\cmdb#1{\ifx#1\end \let\next=\relax
   \else \global\advance\myc by1 \let\next=\cmdb
   \fi \next}
```

Thus the creation of the address file is very easy and readable.

To make it a little easier, a small modification to `merge.sty` has been made so that *after* the first address pair one can insert a `%` as the first character of a line. The modified version is called `mergeh.sty`.

9 Examples

In these examples, the extent of the contents of the picture are shown together with its origin to illustrate what is happening. The file `hdefine.clo` was as shown in section 5.

Ex. 1 The LaTeX file contained:

```
\documentclass[11pt,english]{hletter}
\begin{document}
\signature{Sir Frederick Treves\\
   Sergeant-Surgeon to His Majesty the King}
\reference{Impressions of the journey from
   Vevey to Lausanne}
\date{Lausanne, le 15 septembre 1922}
\begin{letter}{M. Francois Marie Arouet \\
              6, rue du Grand Ch\^{e}ne \\
              \textbf{Lausanne} \\
              Switzerland}
\opening[{[COPY]}]{Dear Voltaire,}
 ...
\closing{I remain, Sir,\\yours Truly,}
\vfill
\cc{All Smiths in London
    \\ Mademoiselle S. Curchod}
\encl{Tourist guide to Switzerland.
     \\ Plan of Cully.}
\end{letter}
\end{document}
```

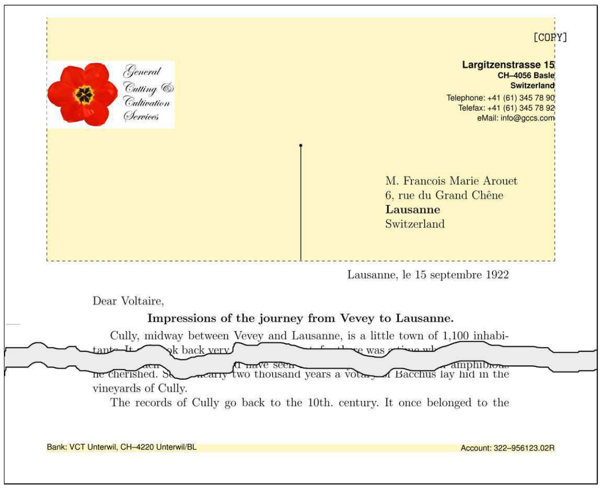
and the default (value=0) in the file `hlete.clo` specified:

```
\addressA{Largitzenstrasse 15}
\addressB{CH--4056 Basle}
\addressC{Switzerland}
\extraA{Telephone: +41 (61) 345 78 90}
\extraB{Telefax: +41 (61) 345 78 92}
\extraC{eMail: info@gccs.com}
\bottomL{Bank: VCT Unterwil, CH--4220
         Unterwil/BL}
\bottomR{Account: 322--956123.02R}
```

The truncated output is shown in figure 5. The example would be improved if the logo was somewhat larger, e.g., `\logo[36mm]{GCCS}`.

Ex. 2 Here the commands used were:

```
\documentclass[11pt,german,bruni]{hletter}
\begin{document}
\signature{Dr.~C. Featherstonehaugh &
   Dr.~A. Beauchamp \\ CEO & CIO}
\reference{Impressions of Lausanne}
\date{Lausanne, le 15 septembre 2008}
\begin{letter}{Sir F. Treves, Bart.,\\
              \textbf{Vevey.}\\
```



[COPY]

Largitzenstrasse 15
CH–4056 Basle
Switzerland
Telephone: +41 (61) 345 78 90
Telefax: +41 (61) 345 78 92
eMail: info@gccs.com

General Cutting & Cultivation Services

M. Francois Marie Arouet
6, rue du Grand Chêne
Lausanne
Switzerland

Lausanne, le 15 septembre 1922

Dear Voltaire,

Impressions of the journey from Vevey to Lausanne.

Cully, midway between Vevey and Lausanne, is a little town of 1,100 inhabitants. It [illegible]ok back very [illegible] re was [illegible]
[illegible] have seen [illegible] amphibious
he cherished. S[illegible] two thousand years a vota[illegible] Bacchus lay hid in the vineyards of Cully.

The records of Cully go back to the 10th. century. It once belonged to the

Bank: VCT Unterwil, CH–4220 Unterwil/BL Account: 322–956123.02R

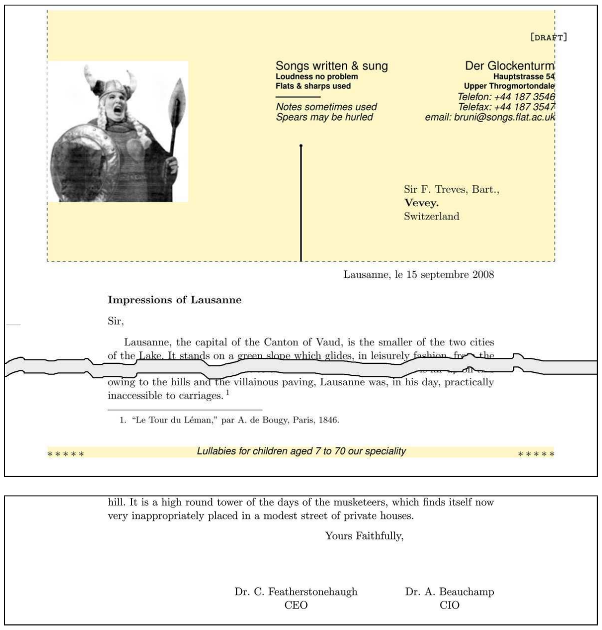
Figure 5: The letter using the defaults (Ex. 1).

```
               Switzerland}
\opening[\textsc{[draft]}]{Sir,}
 ...
\closingtwo{Yours Faithfully,}
\vspace{2cm}
\cc{All Smiths ... S. Curchod}
\encl{Tourist guide ... Cully.}
\vfill
\end{letter}
\end{document}
```

The file `hletg.clo` for the option *bruni*:

```
% case = 3 (bruni)
\addressA{Der Glockenturm}
\addressB{Hauptstrasse 54}
\addressC{Upper Throgmortondale}
\extraA{Telefon: +44 187 3546}
\extraB{Telefax: +44 187 3547}
\extraC{email: bruni@songs.flat.ac.uk}
\centreA{Songs written \& sung}
\centreB{Loudness no problem}
\centreC{Flats \& sharps used}
\centreD{\rule[.5ex]{16mm}{1pt}} % a rule
\centreE{Notes sometimes used}
\centreF{Spears may be hurled}
\centrepos{-10mm}
   % fancy footer:
\bottomL{$\ast\ast\ast\ast\ast$}
\bottomC{Lullabies ... our speciality}
\bottomR{$\ast\ast\ast\ast\ast$}
\sign[10mm]{signat}
\logo[50mm]{Bruennhilde}
\DeclareFixedFont{\newfa}{OT1}
  {phv}{m}{n}{12pt}
\DeclareFixedFont{\newfc}{OT1}
  {phv}{m}{sl}{10pt}\or
```

This contained a larger logo, two signees, a rather special footer and it also changed the default fonts `\newfa` and `\newfc`. The font `\newfa` is used for `\addressA` and `\centreA`; `\newfb` is used for address and centre B and C; all the other fields use `\newfc`.



[DRAFT]

Songs written & sung
Loudness no problem
Flats & sharps used

Notes sometimes used
Spears may be hurled

Der Glockenturm
Hauptstrasse 54
Upper Throgmortondale
Telefon: +44 187 3546
Telefax: +44 187 3547
email: bruni@songs.flat.ac.uk

Sir F. Treves, Bart.,
Vevey.
Switzerland

Lausanne, le 15 septembre 2008

Impressions of Lausanne

Sir,

Lausanne, the capital of the Canton of Vaud, is the smaller of the two cities of the Lake. It stands on a green slope which glides, in leisurely fashion, fr[illegible] the
[illegible]
owing to the hills and the villainous paving, Lausanne was, in his day, practically inaccessible to carriages.[1]

1. "Le Tour du Léman," par A. de Bougy, Paris, 1846.

***** *Lullabies for children aged 7 to 70 our speciality* *****

hill. It is a high round tower of the days of the musketeers, which finds itself now very inappropriately placed in a modest street of private houses.

Yours Faithfully,

Dr. C. Featherstonehaugh
CEO

Dr. A. Beauchamp
CIO

Figure 6: First part of the Bruennhilde letter and the double closing (Ex. 2).

The output is shown in figure 6. The `\sign` command is ignored for two signees.

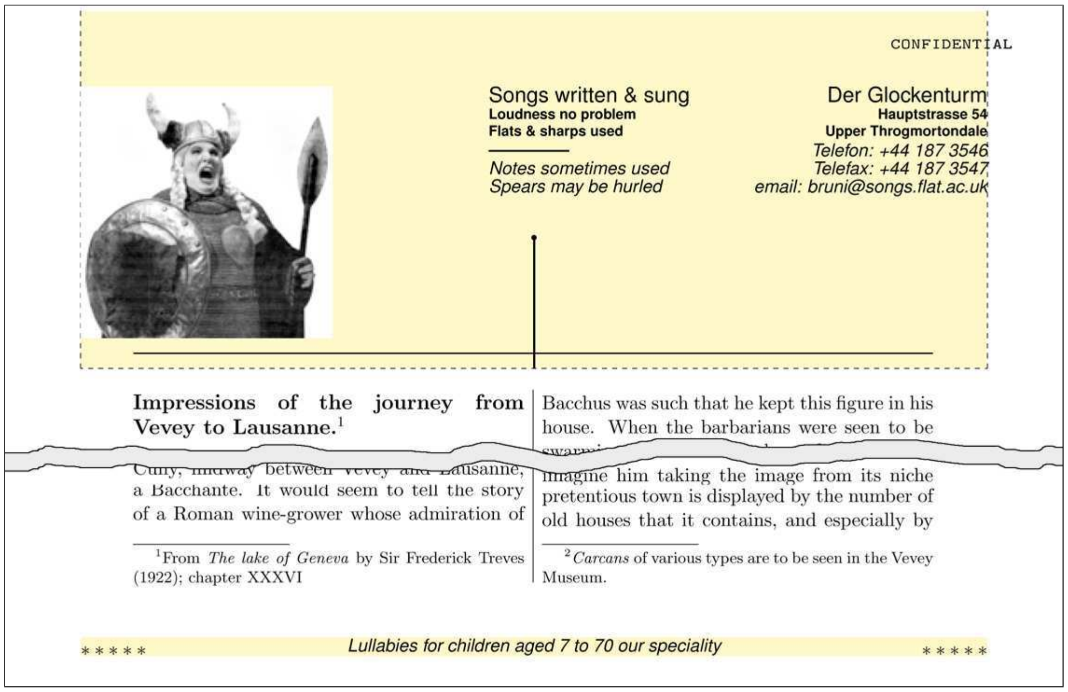
Ex. 3 This example is a simple heading for a two column document. The *bruni* option is used again and the document used the commands:

```
\documentclass[11pt,a4paper,twocolumn]
  {article}
\usepackage[german,bruni]{hhead}
\begin{document}
\setlength{\columnseprule}{.4pt}
\barlength{\textwidth}
\heading[\textsc{confidential}]
```

Note that the commands to specify the header may be placed in the definition file, the *hlet* file or in the document itself. The result is shown in figure 7.

Ex. 4 An example of using the slightly modified *merge* package contains the commands:

```
\documentclass[11pt,english,signit]{hletter}
\usepackage{mergeh}
\signature{A. Nother\\Head of Batology Dept.}
\date{Lausanne, le 15 septembre 2008}
\begin{document}
 \reference{Impressions of the journey from
     Vevey to Lausanne}
 \begin{merge}{testmerge.dat}
 between Vevey and Lausanne
```



CONFIDENTIAL

Songs written & sung
Loudness no problem
Flats & sharps used

Notes sometimes used
Spears may be hurled

Der Glockenturm
Hauptstrasse 54
Upper Throgmortondale
Telefon: +44 187 3546
Telefax: +44 187 3547
email: bruni@songs.flat.ac.uk

Impressions of the journey from Vevey to Lausanne.[1]

Cully, midway between Vevey and Lausanne, a Bacchante. It would seem to tell the story of a Roman wine-grower whose admiration of

Bacchus was such that he kept this figure in his house. When the barbarians were seen to be

imagine him taking the image from its niche pretentious town is displayed by the number of old houses that it contains, and especially by

[1]From *The lake of Geneva* by Sir Frederick Treves (1922); chapter XXXVI

[2]*Carcans* of various types are to be seen in the Vevey Museum.

***** *Lullabies for children aged 7 to 70 our speciality* *****

Figure 7: A heading for Bruennhilde (Ex. 3).

```
 ..
 unfortunately the suggestion is unfounded.
 \closing{Yours Sincerely,}

 \vfill
 \cc{All Smiths ... S. Curchod}
 \encl{Tourist guide ... Cully.}
 \end{merge}
\end{document}
```

and part of the address file `testmerge.dat` is shown below.

```
{Professor Alfred B. Colquhoun\\
   Tittlebat Research Centre\\
   \textbf{Isle of Skye}\\
   Scotland}
Dear Prof.~Colquhoun,
%  old Coony
{Mr.~A. Miller\\
   23a, Council Flats\\
   Park Lane\\
   \textbf{London WC1}}
Dear Archibald,
% first Miller
Dr.~V. M\"{u}ller\\ Langstrasse 15
  \\ \textbf{3012 Bern}
Dear Vee,
%
%{Mr.~A. Nother\\
%   123 High street\\
%   \textbf{Nether Poppleton}\\
%   Nr. York\\ England}
%Hello Alf,
%% Skip alf today
{Viscountess Elizabeth
     Featherstonehaught-Cholmondeley\\
   Cathedral Close\\
   \textbf{Winchester}}
My Dearest Elizabeth,
%
{Sir Archibald Bloggs\\
   Jones Old Yard\\
   Gasworks Lane\\
   \textbf{Throgmortendale}}
Howdy Sir Archie,
% NOTE:
% Comments are allowed between addresses
% but NOT before the first address
% and NO BLANK LINES!
```

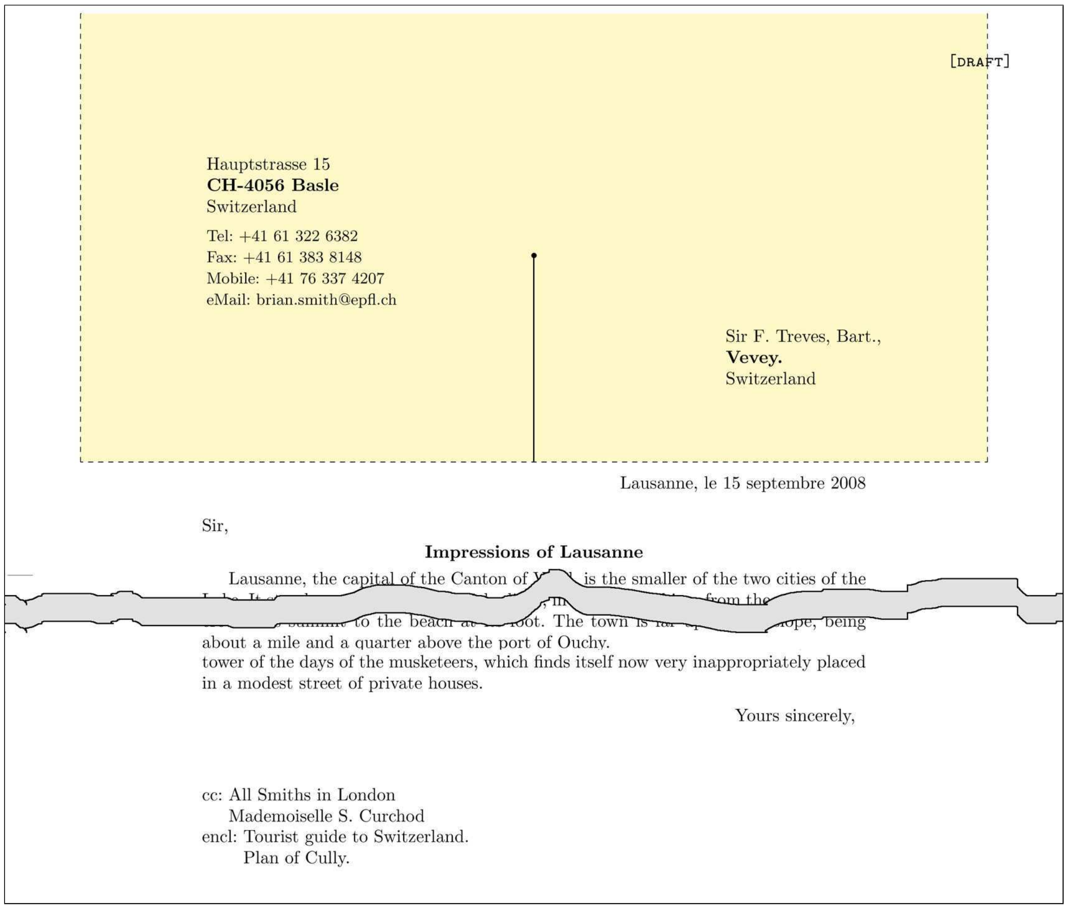
The address of the viscountess gives a class warning, ‘`** Address too wide for window **`’.

Ex. 5 A private letter used the commands:

```
\documentclass[10pt,private,french]{hletter}
\begin{document}
\signature{} % do not used closing name
\reference{Impressions of Lausanne}
\date{Lausanne, le 15 septembre 2008}
\begin{letter}{Sir F. Treves, Bart.,\\
               \textbf{Vevey.}\\
               Switzerland}
```

and here `hletf.clo` contained:

```
% case = 1 (private)
```



[DRAFT]

Hauptstrasse 15
CH-4056 Basle
Switzerland

Tel: +41 61 322 6382
Fax: +41 61 383 8148
Mobile: +41 76 337 4207
eMail: brian.smith@epfl.ch

Sir F. Treves, Bart.,
Vevey.
Switzerland

Lausanne, le 15 septembre 2008

Sir,

Impressions of Lausanne

Lausanne, the capital of the Canton of [illegible] is the smaller of the two cities of the [illegible] from the [illegible] to the beach at [illegible]foot. The town is [illegible]ope, being about a mile and a quarter above the port of Ouchy.
tower of the days of the musketeers, which finds itself now very inappropriately placed in a modest street of private houses.

Yours sincerely,

cc: All Smiths in London
Mademoiselle S. Curchod
encl: Tourist guide to Switzerland.
Plan of Cully.

Figure 8: A private letter (Ex. 5).

```
    \address{Rue principal 15\\
        \textbf{CH-4056 B\^ale}\\
        La Suisse\\[1ex]
        \small Tel: +41 61 322 6382\\
        \small Fax: +41 61 383 8148\\
        \small Mobile: +41 76 337 4207\\
        \small eMail: brian.smith@epfl.ch}
\or
```

and the result is shown in figure 8.

10 Possible future changes

The first version was called *gletter* (for the company GCCS), *h* was the next letter so maybe a future version will be called *iletter*.

One change which has been suggested is to make the dimensions of the headers easier to specify rather than changing values in the package. Also, the positioning of the text and logo should be more flexible. I also wish to sort out the present confusion in the package between the *babel* options *english* and *british*. At the moment specifying *english* invokes *british* which is really not correct. The reason for the mix is that *english* was originally used and then it was requested that I also include *british* — but I was rather lazy!

The support of North American stationery was planned but depends on when and if I acquire samples of the writing materials.

◇ Brian Housley
GCCS GmbH, Switzerland
`brian dot housley (at) gccs dot ch`

References

[1] Donald E. Knuth, *The TEXbook*, 15th ed., Addison-Wesley, 1989, ISBN-10: 0201134489.

[2] Leslie Lamport, *LATEX: User's guide & reference manual*, 2nd ed., Addison-Wesley, 1994, ISBN-10: 0-201-52983-1.

[3] Graeme McKinstry, *Form letters*, TUGboat **8** (1987), no. 1, 60–61, (macros revised 6 September 1988).

Towards LaTeX coding standards

Didier Verna

Abstract

Because LaTeX is only a macro expansion system, the language does not impose any kind of good software engineering practice, program structure or coding style. Maybe because in the LaTeX world, collaboration is not so widespread, the idea of some LaTeX coding standards is not so pressing as with other programming languages. Over the years, however, the permanent flow of personal development experiences contributed to shaping our own taste in terms of coding style. In this paper, we report on all these experiences and describe the programming practices that have helped us improve the quality of our code.

1 Introduction

If the notion of coding style is probably almost as old as computer science itself, the concern for style in general is even older. An interesting starting point is the book "The Elements of Style" [16], first published in 1918 (the fourth edition appeared in 1999). This book is a style guide for writing American English and has been a source of inspiration for similar books in computer science later on. It is interesting to mention the fact that this book has been virulently criticized since its very first publication. Although generally considered as a reference book, the authors were also accused of being condescending in tone and of having only a very partial view on what proper American English should be. This is already a strong indication that talking about style, whether in natural of programming languages, can be quite controversial. Indeed, a style, in large part, is a matter of personal taste before anything else. Consequently, what is considered to be good style by one person can legitimately be viewed as bad style by another person.

The first book on style in programming languages was published in 1974 (a second edition appeared in 1978) and was entitled "The Elements of Programming Style" [8], as an explicit reference to its aforementioned predecessor. Although this book was not dedicated to one programming language in particular, it was still largely influenced by the few of that time. Since then, numerous books on programming style appeared, many of them focusing on one specific language, and being entitled "The Elements of `XXX` Programming Style" to follow the tradition. This includes recent languages such as C#.

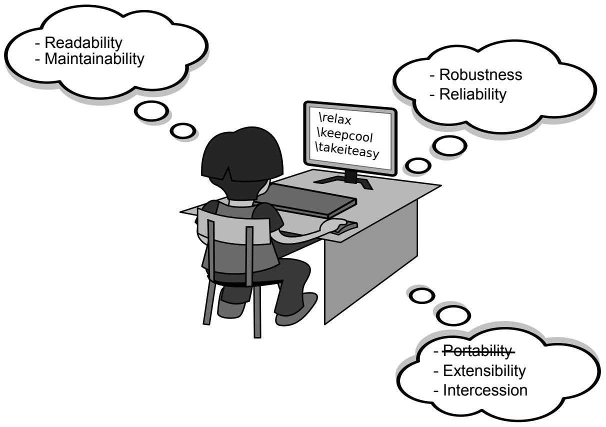


Figure 1: The coding standards many-festos

1.1 The coding standards many-festos

If one looks at the rationale behind most coding styles, the intended purpose is always to

> help programmers <u>read</u> and <u>understand</u> source code, not only their own, but that of <u>others</u>.

An interesting paragraph from the introductory section of the GNU Coding Standards [15] reads as follows:

> Their purpose is to make the GNU system <u>clean</u>, <u>consistent</u>, and easy to install. This document can also be read as a guide to writing portable, <u>robust</u> and <u>reliable</u> programs.

From these widely accepted views on the notion of coding style, we can draw three different points of view on the subject, as depicted in figure 1.

Human From the human point of view, using a proper coding style helps to improve the readability of source code, and as a corollary, its maintainability.

Software From the software point of view, using a proper coding style helps to make the program more robust and reliable. Note that there is a subtle but important difference between robustness and reliability. Reliability means that the program should do what it is expected to do. Robustness means that the program should handle unexpected situations as gracefully as possible.

Man-in-the-middle Third and last, the intermediate point of view is at the interface between humans and programs (note the plural). In this regard, the GNU Coding Standards mention the question of portability. This is essentially due to the fact that the GNU project mostly deals with C code, for which portability is indeed an important problem. This, however, is much less of a concern to us because practically all LaTeX programs are inherently portable (TeX itself being mostly a virtual machine). A much more important question for us is the question of extensibility and more generally, intercession.

By extensibility, we mean to answer the following question: given a package that does *almost* what a specific user wants, is it possible to make this package provide the requested functionality without modifying its internal implementation? If the answer is yes, then the package (or at least one of its functionalities) can be said to be extensible. In this context, one of the purposes of a coding style is to help provide more, and better extensibility.

Unfortunately, full extensibility is only a utopia because ultimately, the specific desires of a user are completely unpredictable. In such situations, a package may need to be internally modified. This is what we call "intercession". The terminology comes from the more general field of so-called "reflexive" languages [10, 14]. Roughly speaking, a reflexive language provides the ability to reason about the program itself (procedural reflection) or even the language itself (behavioral reflection). Reflection is usually decomposed into "introspection" (the ability to look at yourself) and "intercession" (the ability to modify yourself).

While extension is usually a matter of user–package interaction, intercession is usually due to inter-package interactions. In the LaTeX world, we can identify three major sources of intercession.

1. LaTeX core modification: a package needs to modify LaTeX itself in order to provide the required functionality.
2. Package inter-compatibility: a package needs to co-exist with another package, and this requires modifications in either or both of them.
3. Package conflict: two (or more) packages intercede on the same piece of code but in different ways, or one package modifies some code and another package is not made aware of these modifications. In both cases, compilation breaks.

Every LaTeX user faces the "package conflict nightmare" one day or another [19], to the point that this is probably the major gripe against it these days. Consequently, we would hope that a proper coding style addresses this issue, for example by providing design patterns for graceful inter-package compatibility handling.

1.2 Consistency

One final keyword that appears quite a lot in discussions on coding style is "consistency". Given the fact that there is much personal taste in a coding style, consistency means that the exact coding style that you decide to use is actually less important than the fact of sticking to it. A person not familiar with your coding style can probably get used to it, provided that it is used consistently in the whole source code, and that for example, similar situations are identifiable as such because the same idioms are used in all of them.

1.3 30 years and no style?

Since more or less official coding standards seem to exist for many programming languages and communities, one can't help but wonder why, after 30 years of existence, the LaTeX community still doesn't have any. Several reasons come to mind.

1.3.1 Learning by example

LaTeX is not a real programming language. It is not even a macro expansion system. LaTeX is a *library*: a macro layer written on top of TeX. Because of that, the purpose of LaTeX can be anything you might want to do related to typography, which can eventually be expressed in terms of TeX. Consequently, it is impossible to write "The LaTeX programming language" book and in fact, this book doesn't exist. The things that such a book would have to describe are infinite: they depend on every user's goal. Note that the *LaTeX Companion* [12] is *not* a LaTeX programming book. For the most part, it describes some of the core functionalities, plus a lot of package features.

This explains why learning by example is an important process in the LaTeX community. It is quite easy to backtrack from a particular feature to the way it is done: you just need to look at the implementation. As a result, many LaTeX programmers (especially newcomers) start by actually looking at what other people did before us, copy-pasting or imitating functionality until they reach a satisfactory level of understanding. In doing so, they also implicitly (and unconsciously) inherit the coding style (or lack thereof) of the code they are getting inspiration from. This behavior actually encourages legacy (the good *and* the bad) and leads to a very heterogeneous code base.

1.3.2 Lack of help

Because it is only a macro library, LaTeX is not a structured language but a very liberal one. By providing such paradigms as object-oriented, functional, logic, declarative programming, *etc.*, traditional languages provide support for some "elements of style" by construction: the choice of a specific programming paradigm already imposes a particular design on your program. On the other hand, when you program in LaTeX, you are essentially on your own. You don't get any help from the language itself.

For the same reason (lack of structure), getting help from your favorite text editor is even more complicated. Even theoretically simple things such as indentation can be an editor's nightmare. Indenting within braces in a traditional language is relatively simple: it's a matter of syntactic analysis. But suppose that you want to indent the contents of `\if<whatever>` conditionals in (LA)TeX. First, this is not a syntactic construct but a macro call. Next, the closing `\fi` may be difficult to spot: it may be the result of the expansion of another macro for instance. Worse, its precise location may also depend on a dynamic (run-time) context! This shows that in general, it is impossible to get even simple things right without doing a full semantic analysis of the program, which itself may not even suffice. In a powerful development environment such as the combination of (X)Emacs [3] / AUC-TeX [1], you will typically need to indent the first line of a conditional by hand, and the rest will follow by simply hitting the TAB key. If, on the other hand, you let the editor do everything, you end up with a broken indentation layout scheme.

1.3.3 Lack of need

A survey conducted in 2010 [19] shows that LaTeX is mostly a world of dwarfs. In the TeX Live 2009 distribution, the average size of a package is 327 lines of code, the median being 134. Admittedly, very few people would feel the need for a proper coding style, when it comes to maintaining just a few hundred lines of code. In addition to that, it seems that LaTeX suffers from an anti-social development syndrome: most packages are single-authored and maintained, which leads to the same kind of consequences. When no interaction is required between people, establishing a set of coding standards for working on a common ground is a far less pressing issue.

On the other hand, imagine the difference with an industrial language in which millions of lines of code would be maintained by a team of hundreds of developers. The need for coding standards is obvious. Unfortunately, if you consider the LaTeX code base on CTAN — [2], it *is* a huge one, only maintained in a completely independent and uncontrolled fashion.

1.4 30 years and *almost* no style...

Claiming that there is no coding style for LaTeX turns out to be a slight exaggeration. By looking closely enough, we can spot a few places where the question is indeed addressed.

1.4.1 Tools

TeX itself provides some facilities for stylish programming. The equivalence between blank lines and `\par` encourages you to leave more room in your text, therefore improving its readability. TeX also conveniently ignores blanks at the beginning of lines, a crucial behavior when it comes to indenting your code without leaving spurious blanks in the generated output.

A number of packages (*e.g.* `calc` and `ifthen`) provide additional layers of abstraction on top of the LaTeX kernel, therefore providing structure where it was originally lacking. More abstraction means improved readability. Some packages like `record` even go as far as providing data structures or programming paradigms coming from other, more traditional programming languages. Of course, the difficulty here is to be aware of the existence of such packages.

1.4.2 Conventions

A number of coding conventions have existed for a long time now, including in the LaTeX kernel itself. The use of lowercase letters for user-level macros and mixed up/downcase for extension names (*e.g.* `\usepackage` *vs.* `\RequirePackage`) is one of them. The special treatment of the `@` character in macro names effectively allows one to make a clear distinction between internal and external code.

It is worth mentioning that LaTeX itself does not fully adhere to its own conventions (we see here a typical effect of legacy). For example, the macro `\hbox` is not supposed to be used externally, and hence should have been named with an `@` character somewhere. Conversely, a better name for `\m@ne` would have been `\MinusOne`.

1.4.3 Documentation

The *LaTeX Companion* contains some advice on style. Section 2.1 describes how document files should be structured and Section A.4 does the same for package source code (as we will see later, we disagree with some of the advice given there). It also mentions some of the development tools that help making source code more structured and modular (*e.g.* `doc`, `docstrip`, `ltxdoc`).

Some of these packages are described at length, although this does not count as style advice: only mentioning their existence counts, the rest is technical documentation. Modulo this remark, it turns out that the amount of style advice provided in the *Companion* is extremely limited: it amounts to less than 1% of the book.

1.5 The need for coding standards

Even though we can explain the lack of LaTeX coding standards, and even though some people certainly don't feel any need for them, we still think that they would be a valuable addition to the LaTeX world, especially in a much more developed form than what we have described in the previous section. Some important reasons for this are provided below.

Learning by *good* example We have seen earlier how LaTeX encourages "learning by example". Obviously, the existence of coding standards would help filter out poor quality code and have people learn mostly by *good* example only.

Homogeneity We have also seen how a plethora of small packages with no coding style, or author-specific ones only, contributes to make LaTeX a very heterogeneous world. This is the point at which it is important to make a distinction between coding *style* and coding *standards*. A coding style can be seen as a set of personal tastes and habits in terms of programming. A coding standard, by extension, should be defined as a coding style which multiple people agree to conform to.

In spite of the anti-social aspect of LaTeX development, that is, even if independent package developers rarely talk to each other, we know that the very high level of intercession in LaTeX implies that developers are forced to read and understand other people's code. In that context, it becomes apparent that having more-or-less official coding standards would make it easier for people to read and understand others' code. Homogeneity facilitates interaction.

One important problem here is that a consensus never comes without any concession. Coming up with coding standards that would satisfy everyone is highly unlikely, given the importance of personal taste, even if those coding standards leave room for some degree of flexibility. The question that remains open is hence the following: to what extent would people agree to comply with coding standards that diverge from their own habits or preferences, if it is for the greater good of the community?

Intercession There are many other reasons why having coding standards would be a plus. Intercession is another very important one. The way a particular package handles a particular typesetting problem only affects itself: both the problem and the solution are localized. The situation is however very different when it comes to intercession. The way a particular package handles an extension or a conflict (for example by applying dynamic modifications to another package or to the LaTeX kernel) *does* affect the outside world. As a consequence, one would expect coding standards to help clean up the current "intercession mess" by providing a set of rules, perhaps even some design patterns [4, 5, 6, 7, 9, 13] for intercession management. Intercession would become much less of a problem if every package handled it in the same way.

1.6 Coding style levels

Coding standards are supposed to help with writing better code, although we need to clarify what we mean by "better". In our personal development experience, we have identified four abstraction levels at which it is interesting to consider the notion of style. These four levels, which we are going to explore in the next sections, are the following.

1. **Layout** (low). At the layout level, we are interested in code formatting, indentation, macro naming policies, *etc.*
2. **Design** (mid). The design level deals with implementation: how you do the things that you do. This is where we address software engineering concerns such as modularity, encapsulation, the potential use of other programming languages' paradigms, *etc.*
3. **Behavior** (high). The behavior level is concerned with functionality as opposed to implementation: what you do rather than how you do it. At this level, we are interested in user interfaces, extension, intercession (notably conflict management), *etc.*
4. **Social** (meta). Finally, the social level is a meta-level at which we consider human behavior in the LaTeX community. Notions like reactivity and open development are examined.

1.7 Target audience

In the LaTeX world, it is customary to distinguish the *document* author, writing mostly text, from the *package* author (or LaTeX developer) writing mostly macros. Those two kinds of audience slightly overlap, however. By providing automatically generated text (*e.g.* language-dependent), the package author is a bit of a document author. By using packages, fixing conflicts between them and providing personal macros in the preamble, the document author is also a bit of a LaTeX developer. While this paper is mostly targeted at the package developer, many concerns expressed here (most notably at level 1: layout) are also very important for the document author.

2 Level 1: Layout

In this section, we explore the first level of style, dealing with visual presentation of source code and lexico-syntactic concerns such as naming conventions.

2.1 Formatting

One very important concern for readability is the way you visually organize your code. Most of the time, this boils down to a simple question: how and where do you use blanks. This question is more subtle to address in LaTeX than in other, more syntactically structured languages. We have identified four rules which contribute to better formatting.

2.1.1 The rules

Rule #1: Stay WYSIWYG'ly coherent

LaTeX (or TeX, for that matter) has bits of pseudo-WYSIWYG behavior. The fact that a blank line ends a paragraph is one of them. There are also some commands whose effect can be directly simulated in your source code (or document). In such situations, it is probably a good idea to do so.

The two most prominent examples of this are the `\\` and `\par` commands. Since `\\` effectively ends the current line, we find it reasonable to do so in the source file as well. Put it differently: we find it confusing when a `\\` command is immediately followed by text or code that produces text. The same goes for `\par`, with an additional remark: we have seen code in which `\par` is followed by some text, with the explicit intention of *beginning* a new paragraph. Although ending a paragraph can, in some circumstances, be equivalent to beginning the next one, this use of `\par` is extremely confusing because the its semantics are precisely to *end* the current paragraph.

Tabular-like environments are another situation in which it can be nice to mimic the actual output layout. Although it requires a fair amount of work, probably without the help of your favorite text editor, aligning the various tab commands or columns separators across rows helps readability. If, as we do, you prefer to remain within the bounds of 80 columns, a compromise may be necessary between both constraints.

Rule #2: Be "spacey" in math mode

Surprisingly enough, it seems that many people forget that spaces don't count in math mode. This is a good opportunity to take as much room as you want and make your equations easier to read. Consider the following two alternatives. This:

```
$ f(x) = f(x-1) + f(x-2) $
```

is probably better than this:

```
$f(x)=f(x-1)+f(x-2)$
```

Rule #3: One "logical" instruction per line

This rule may be a wee bit fuzzier than the previous ones. By "logical", we roughly mean something (a code sequence) which makes sense as a whole. In traditional programming languages, a logical instruction is generally a function call along with its arguments, or an operator along with its operands. In LaTeX, the situation is more complicated, notably because of the throes of macro expansion.

Perhaps it is simpler to make this point by providing some examples. We assume here that our logical instructions are small enough to fit on one line, the idea being to avoid putting two of them next to each other.

```
\hskip.11em\@plus.33em\@minus.07em
```

This line constitutes only one logical instruction because the sequence of macros and quantities define a single length.

```
{\raggedleft\foo\bar baz\par}
```

Here, the flushing instruction applies until the closing brace (assuming the paragraph ends before the group), so it would be strange, for instance, to go to the next source line after `\bar`. Note however that for longer contents, not fitting on one line only, we would probably go to the next line right after `\raggedleft`, so that the formatting instruction(s) are distinct from the text to which they apply.

In the same vein, it would be unwise to split things like `\expandafter\this\that`, conditional terms such as `\ifx\foo\bar`, and more generally, everything that can be regarded as an argument to what precedes.

Rule #4: Indent all forms of grouping

This rule is probably the most obvious, and at the same time the most important, when it comes to readability. *All* forms of grouping should have their contents indented so that the beginning and end of the groups are clearly visible. It seems that indenting by 2 columns is enough when you use a fixed width font, whereas 4 or 8 columns (or a tab) are necessary otherwise. In general however, using tab characters for indentation is inadvisable (notably in document sources, but sometimes in package sources as well). Tabs can be dangerous, for instance, when you include code excerpts that should be typeset in a special way.

In LaTeX, grouping can occur at the syntactic level with group delimiters (`{}`, `[]`) or math modes (`$` and `\(\)`, `$$` and `\[\]`), and also at the semantic level (`\bgroup` / `\egroup`, `\begingroup` / `\endgroup` or even `\makeatletter` / `\makeatother`). Your favorite text editor will most likely help you indent at the syntactic level, but you will probably need to do some manual work for semantic grouping. In the case of Emacs for example, manually indenting the first line below a call to `\makeatletter` is usually

```
 1  %% Original version:
 2  \def\@docinclude#1 {\clearpage
 3  \if@filesw \immediate\write\@mainaux{\string\@input{#1.aux}}\fi
 4  \@tempswatrue\if@partsw \@tempswafalse\edef\@tempb{#1}\@for
 5  \@tempa:=\@partlist\do{\ifx\@tempa\@tempb\@tempswatrue\fi}\fi
 6  \if@tempswa \let\@auxout\@partaux \if@filesw
 7  \immediate\openout\@partaux #1.aux
 8  \immediate\write\@partaux{\relax}\fi
 9  % ... \fi :-(
10
11  %% Reformatted version:
12  \def\@docinclude#1{%
13    \clearpage
14    \if@filesw\immediate\write\@mainaux{\string\@input{#1.aux}}\fi
15    \@tempswatrue
16    \if@partsw
17      \@tempswafalse
18      \edef\@tempb{#1}
19      \@for\@tempa:=\@partlist\do{\ifx\@tempa\@tempb\@tempswatrue\fi}%
20    \fi
21    \if@tempswa
22      \let\@auxout\@partaux
23      \if@filesw
24        \immediate\openout\@partaux #1.aux
25        \immediate\write\@partaux{\relax}%
26      \fi
27    % ... \fi :-)
```

Figure 2: The virtues of proper formatting

enough to have the subsequent ones follow the same indentation level automatically. But then again, you will also need to manually *unindent* the closing call to `\makeatother`.

As an illustration of both rules #3 and #4, consider the code in figure 2 in both original and reformatted form. In each case, ask yourself: to which conditional does the upcoming `\fi` belong? This point clearly demonstrates the importance of indentation. Line 14 contains an example of what we called a "logical" instruction, although a longer one this time. The contents of the conditional is a single instruction to write something in the auxiliary file immediately. Also, since there is no `\else` part in this conditional and the whole line doesn't exceed 80 columns, we chose to keep it as a one-liner. The same remark can be made for line 19.

2.1.2 Formatting of syntactic groups

In the case of syntactic groups, various policies can be observed regarding the position of the braces (this is also true in other programming languages). The case of an environment definition could be formatted as follows, as is done on several occasions in the LaTeX standard classes:

```
\newenvironment{env}
               {\opening\code
                \opening\code}
               {\closing\code
                \closing\code}
```

We find this kind of formatting somewhat odd and it doesn't seem to be used so frequently anyway. The conspicuous amount of indentation can be disturbing, and it is also a bit difficult to visually distinguish the opening argument from the closing one.

A more frequent way of formatting this would be more or less as in a piece of C code, as follows:

```
\newenvironment{env}
{%
  \opening\code
  \opening\code
}
{%
  \closing\code
  \closing\code
}
```

This kind of formatting is admittedly more readable, although the two nearly empty lines between the opening and the closing arguments may be considered somewhat spurious. Some people hence take the

```
\newcommand\text{%
  \@nextentry
  \noalign\bgroup
    \gdef\@beforespace{...}%
    \@ifstar{\@stext}{\@text}}

\newcommand\@text[1]{%
    \gdef\@nextentry{}%
  \egroup% end of \noalign
  \multicolumn{3}{@{}p ... \\}}

\newcommand\@stext{%
    \gdef\@nextentry{\egroup\\\par}%
  \egroup% end of \noalign
  \multicolumn{3}{@{}p ...} ...}
```

Figure 3: Inter-macro indentation

habit of joining those two lines as follows:

```
\newenvironment{env}
{%
  \opening\code
  \opening\code
}{%
  \closing\code
  \closing\code
}
```

Other people choose a more compact formatting by closing a group, and possibly opening the next one on the same line, as follows:

```
\newenvironment{env}{%
  \opening\code
  \opening\code}{%
  \closing\code
  \closing\code}
```

Again, this leads to quite compact code that makes it difficult to visually distinguish the opening argument from the closing one. In such a case, a possible workaround is to introduce comments, also an opportunity for documenting the macro's prototype (imagine that in a text editor with fontification, you might also have different colors for code and comments):

```
\newenvironment{env}{%
  %% \begin{env}
  \opening\code
  \opening\code}{%
  %% \end{env}
  \closing\code
  \closing\code}
```

2.1.3 Inter-macro indentation

The case of semantic grouping introduces an additional level of complexity because groups may be opened and closed in different macros (worse: the opening and closing instructions may themselves be the result of macro expansion). When possible, it is a good idea to preserve the amount of indentation corresponding to the current group nesting level, even if the group in question is not syntactically apparent.

Consider for example the code in figure 3 taken from the CurVe class [17]. The `\text` command calls `\noalign`, but the argument passed to `\noalign` (enclosed in `\bgroup`/`\egroup`) starts here and ends in either `\@text` or `\@stext`. You can see that this group's indentation level is preserved across all three macros.

2.1.4 Exceptional situations

No rule goes without exception. Sometimes, and for the greater good, one might be tempted to go against the established rules. Here are two examples.

Consider the following call to `\@ifnextchar`:

```
\@ifnextchar[%] syntax screwup!
  {\@dothis}{\@dothat}
```

The left square bracket, which is in fact the first argument of `\@ifnextchar`, confuses Emacs because it thinks it's the opening of a group, and expects this group to be closed somewhere. In order to compensate for this problem, we usually virtually close the fake group by putting a right square bracket within a comment on the same line. This forces us, however, to provide the "then" and "else" arguments to `\@ifnextchar` on the next line, something that we would normally not do.

Another exceptional situation is the case of empty macro arguments, where we prefer to stay on the same line rather than consuming another one just for an empty pair of braces, as illustrated below:

```
\@ifundefined{#1note}{}{%
  \@fxpkgerror{a short explanation}{%
    a longer one}}
```

2.1.5 Corollary

As a corollary to the rules described in this section, it is essential to note that the `%` character is your "worst best friend". A very important problem when writing macros (and even documents) is the risk of spurious blank spaces. When you indent your code properly, many blanks are inserted, which are not supposed to appear in the final document. TeX helps you with that in several ways: spaces are eaten after a control sequence, consecutive blanks are treated as only one (this includes the newline character), and leading / trailing spaces are discarded on every line.

That alone, however, is not sufficient for a liberal indentation scheme. In the previous examples, we have seen many places (notably after braces) where it is required to create an end-of-line comment with the `%` character, so that the final newline character is not taken as a textual one (see for example figure 3 on the previous page).

In that sense, the `%` character is your best friend. It is also your worst friend because determining the exact places at which an end-of-line comment is required is far from trivial. There are even cases where it could be necessary after opening an environment in a final document! In any case, when there are blanks in your source that you *know* you don't want in the output, and you're unsure whether TEX will skip them on its own, you can safely always insert a comment character at the end of the line.

2.2 Naming

The second concern we want to address in this section is that of naming schemes. Naming conventions are obviously important for readability, but also for backward compatibility. Once you get a name, it's for life. Starting with bad naming conventions can become a major headache, both for your clients (using your API) and yourself (maintaining your own code).

2.2.1 The rules

Rule #1: Use prefixes

Because LATEX lacks a proper notion of module, package, or even namespace, the use of a specific prefix for every package that you write should be a rule of thumb. For example, our FiXme [18] package uses `fx` as a prefix, which means that every command (but see rule #3) starts with those two letters.

The choice of the prefix is also important. In theory, the prefix that would guarantee a minimal risk of name clash between packages would be the full package name. In practice however, this can lead to very long macro names, cumbersome to type. Therefore, a trade-off must be made between the prefix length and its uniqueness (a possible idea is to start by removing the vowels). `fx` for example has the defects of its qualities: it is practical because it is very short, but the risk of collision with only two letters is not negligible.

Once you have chosen a prefix, it is also important to stay consistent and stick to it. Recently, we discovered that for some obscure (and forgotten) reason, our ***FiNK*** package uses a prefix of `fink` for its user-level commands, but only `fnk` for its internal ones. This is not only rinadvisable but also unnecessary since LATEX already provides the `@` character convention for making such a distinction (*cf.* rule #3).

One situation where the prefix rule should probably be relaxed is the case of classes (as opposed to styles). Classes, by definition, are mutually-exclusive and perform similar, very general tasks, although in different ways. It would hence be silly to have to name similar things differently (imagine for instance that the sectioning commands were named `\artsection`, `\rprtsection` and `\bksection`!). On the other hand, the risk of collision is still high, precisely because of the broad spectrum of class functionality. This problem has already befallen us in the ***CurVe*** class, which provides a `\text` macro, also implemented (to do something different) by `siunitx` and probably other packages. `\text` is the perfect example of a very poor choice of name because it is far too general and doesn't really mean anything. This demonstrates that choosing a pertinent, unique and concise name for a macro is an important but tricky exercise.

Rule #2: Use postfixes

In a very analogous way, there are situations in which the use of a postfix may be a good idea in order to avoid name clashes, although this time not with other packages, but with yourself. LATEX provides a number of concepts, loosely related to data types or structures, such as counters and saveboxes. Unfortunately, the provided interfaces are rather inconsistent.

In some situations like counters, you are only required to provide a name, and LATEX constructs the underlying, opaque macros with a specific naming scheme. What's more, you are not supposed to use those macros explicitly. Suppose for example that you want to maintain a counter of "items". There is no need to name this counter `myitemscount` because the standard interface makes things perfectly readable without the postfix:

```
\newcounter{myitems}
... \value{myitems} % not a very good name
... \stepcounter{myitems}
```

Besides, the risk of name clash is minimal because under the hood, LATEX has used a specific and hopefully unique naming scheme for naming the counter macro (`\c@myitems`).

Suppose now that you want to save your items in a box. In that case, you are requested to provide a macro name yourself, and choosing `\myitems` is for sure a bad idea because that name is too general (there is no indication that you're talking about the *box* of them, and not the number, list or whatever else of them). What you need to do, therefore, is decide on a specific naming scheme for boxes, just as LATEX does under the hood for counters. Using a `box` postfix appears to be a good solution:

```
\newsavebox\myitemsbox
... \savebox\myitemsbox{...}
... \sbox\myitemsbox{...}
```

Of course, there is some naming redundancy in this code, but that is what you get from an interface that is not as opaque as it should be.

If you are developing a package (as opposed to a document) and want to maintain an *internal* list of items, you may also be tempted to follow LaTeX's own convention for, say, counters, and call your macro `\b@myitems` or something like that. We advise against that, however, because it conflicts with the prefix rule described previously, and also because it would make your code less readable (remember that you need to use the macro explicitly, not just the "name of the thing").

Finally, note that the ultimate solution to this kind of problem would be to develop another, properly abstracted layer on top of the original one, in which the actual macro names are never used explicitly, and standardize on it...

Rule #3: Obey the *Companion*

The *LaTeX Companion* provides some advice on naming in section A.1. Modulo a substantial amount of legacy code, LaTeX itself tries to adhere to the naming conventions described there so it is a good idea to honor them in your packages as well. For starters, you are invited to name your external macros with lowercase letters only, and reserve a mixture of lowercase and uppercase names for extension APIs. FiXme, for example, follows this convention by providing an end-user command named `\fxuselayout`, and at the same time an equivalent command named `\FXRequireLayout` for theme authors.

The other important and well known naming convention adopted by LaTeX is the use of an `@` character in internal macro names. By turning this character into a *letter* (category code 11) only internally and in packages, LaTeX effectively prevents the document author from using such macros directly (one would have to intentionally enclose a call to an `@`-macro within `\makeatletter` / `\makeatother`).

Third-party packages should obviously follow this convention in order to separate internal from external macros. Package authors should however do a better job at naming internal macros than LaTeX itself (again, we see here the effect of a long legacy). The LaTeX kernel seems to enjoy making fun of the `@` character, using it in place of different vowels (*e.g.* `\sixt@@n` or `\@filef@und`) and with no apparent rationale in terms of number and position (*e.g.* `\@input`, `\@@input` but `\@input@`).

Although we underst@nd how this c@n be fun, it is better for readability to keep a more systematic approach to naming internal macros. Typically, we find that using the `@` character is useful in two situations:

```
\DeclareRobustCommand\fxnote{%
  %% ...
  \@ifstar{%
    %% \fxnote*
    \@ifnextchar[%]
      {\@fxsnote{#2}}
      {\@@fxsnote{#2}}}{%
    %% \fxnote
    \@ifnextchar[%]
      {\@fxnote{#2}}
      {\@@fxnote{#2}}}}

\long\def\@fxsnote#1[#2]#3#4{%
  %% ...
  \@@fxsnote{#1}{#3}{#4}}

\long\def\@@fxsnote#1#2#3{%
  \implement\me}
```

Figure 4: Nesting levels

- as a prefix to indicate an internal implementation of an external functionality,
- as a word separator.

For example, the current (language-dependent) value for the "List of FiXme's" section name is stored in a macro named `\@fxlistfixmename` (an acceptable alternative would be `\fx@listfixmename`).

In some situations, the implementation of a particular feature may go through different levels of indirection. In such cases, we like to use multiple `@` characters to give an indication of the current implementation level. Figure 4 illustrates this. The macro `\fxnote` supports an optional `*` postfix as well as a regular optional first argument provided in square brackets. The implementation goes through a first sub-level that detects the presence of a `*` character (`\@fxnote` / `\@fxsnote`), plus another sub-level which handles the presence of an optional argument (`\@@fxnote` / `\@@fxsnote`).

A final example is the case of "polymorphic" macros (see section 3.3 on page 319), that is, macros whose implementations depend on some context. As mentioned earlier, the `@` character can be used to separate words. For instance, FiXme has a macro named `\@@@fxnote@early`. This macro is polymorphic in the sense that its actual implementation varies according to the document's draft or final mode. The two corresponding effective implementations are named `\@@@fxnote@early@draft` and `\@@@fxnote@early@final`.

2.2.2 Exceptional situations

From time to time, the naming rules exhibited in the previous section may be bypassed for the sake of readability. Here are three typical situations where exceptions are in order.

Conforming to *de facto* standards LaTeX itself has some naming conventions that may impact a package or even a document author. Lists are one such case. For example, the behavior for the list of figures depends on two macros named `\listoffigures` and `\listfigurename`. FiXme supports its own list facility, and for the sake of coherence, provides analogous macros named `\listoffixmes` (instead of `\fxlist` or some such) and `\listfixmename` (instead of `\fxlistname`). Following the usual convention makes it much easier for your users to remember your own API.

Another example is that of conditionals. *All* conditionals in (LA)TeX are named `\if`⟨*something*⟩. So here again, given that you need to implement `mycondition`, it is better to name your conditional `\ifmycondition` than `\myifcondition`.

Forced exceptions There are times where LaTeX itself will force you to depart from your own rules, although this is seldom critical. The case of counters is one of them. When creating a counter for `myitems`, LaTeX creates a macro named `\c@myitems` which is not how you would have named this macro. However, this is not such a big deal because in general, you don't need to use this macro directly.

A slightly more intrusive exception is when LaTeX requires that you implement a specific macro, following its own naming scheme. For instance, supporting a list of FiXme's involves implementing a macro named `\l@fixme`. The `l@` prefix is LaTeX's choice, not ours.

Finally, if you implement an environment named `myenv`, LaTeX will eventually turn this into a macro named `\myenv` and another one named `\endmyenv`. Here again, the names are LaTeX's choice, not yours. And by the way, it is unfortunate that the environment opening macro is not named `\beginmyenv` instead of just `\myenv` because it means that you can't have both a command and an environment with the same name. In the FiXme package, we use a nice naming trick for this kind of situation: environments corresponding to macros are prefixed with "`a`" or "`an`". For example, there is a macro named `\fxnote` and the corresponding environment is named `anfxnote`. This contradicts our own naming conventions but it makes the actual environment usage as readable as if it were plain English:

```
\begin{anfxnote}
...
\end{anfxnote}
```

3 Level 2: Design

In this section, we explore the second level of style, dealing with design considerations such as modularity and other programming paradigms. From a more practical point of view, *design* here is concerned with how to implement a particular feature, rather than the feature itself.

3.1 Rule #1: Don't reinvent the wheel

3.1.1 Feature libraries

In many programming languages, so-called "standard libraries" provide additional layers of functionality, typically functions that perform useful and frequently needed treatments. Browsing CTAN [2] clearly demonstrates that LaTeX is no exception to this rule. People have created packages for making slides, curricula vitae, split bibliographies, tables that span across several pages, *etc.*

When you develop a package, it is important, although not trivial, to be aware of what's already existing in order to avoid reinventing the wheel. For instance there are currently at least half a dozen different solutions for implementing key-value interfaces to macros (`keyval`, `xkeyval`, `kvoptions`, `pgfkeys`, *etc.*). This is very bad because each solution has its own strengths and weaknesses, so the choice of the most appropriate one for your personal needs can be very complicated and time-consuming (in fact, there might not even be a *best* choice).

One rule of thumb is that when you feel the need for implementing a new functionality, someone most probably had the same idea before you, so there is a good chance that you will find a package doing something close to what you want. In such a case, it is better to try and interact with the original author rather than to start over something new on your own. Doing this, however, also requires some rules in terms of social behavior (*cf.* section 5 on page 326).

3.1.2 Paradigm libraries

Furthermore, in the LaTeX world the notion of standard library goes beyond common functionality: it goes downwards to the language level. TeX was not originally meant to be a general purpose programming language, but TeX applications today can be so complex that they would benefit from programming paradigms normally found in other languages. Because of this, there are packages that are meant to extend the language capabilities rather than providing a particular (typesetting) functionality. The two most prominent examples of this are `calc` and `ifthen`. These packages don't do anything useful in terms of typesetting, but instead make the programmer's life easier when it comes to arithmetic calculations or conditional branches. Another one, called record, even goes as far as providing data structures for LaTeX.

It is always a good idea to use these packages rather than doing things at a lower level, or re-inventing the same functionality locally. The more abstract your code, the more readable. The *LaTeX Companion* advertises some of them (notably calc and ifthen). Of course, the difficult thing is to become aware of the existence of such packages (CTAN contains literally thousands of packages).

3.2 Rule #2: Duplication/Copy-paste is evil

This rule is well-known to every programmer, although the "evilness" threshold may be a bit subtle to calculate. It is also interesting to provide some insight on the distinction we make between "duplication" and "copy-paste". The two examples below will shed some light on these matters.

3.2.1 Duplication

Consider the case of FiXme which uses the xkeyval package for defining several layout-related package options. The bad way of doing it would be as follows:

```
\define@key[fx]{layout}{morelayout}{...}
\define@cmdkey[fx]{layout}{innerlayout}{...}
\define@key[fx]{envlayout}{envlayout}{...}
```

This is bad because the [fx] optional argument (the prefix in xkeyval terminology) is *duplicated* in every single call to the xkeyval package (and it is rather easy to forget).

This is a typical case where duplication should be abstracted away in order to avoid redundancy. We can improve the code by providing *wrappers* around xkeyval as follows:

```
\newcommand\@fxdefinekey{\define@key[fx]}
\newcommand\@fxdefinecmdkey{\define@cmdkey[fx]}
\@fxdefinekey{layout}{morelayout}{...}
\@fxdefinecmdkey{layout}{innerlayout}{...}
\@fxdefinekey{envlayout}{envlayout}{...}
```

It should be noted that this new version is actually *longer* than the previous one. Yet, it is clearer because more abstract. Using such wrappers is like saying "define a FiXme option". This is more abstract than "define an option which has an fx prefix".

Note also that in this example, two "layout" options are defined. One could hence be tempted to abstract the layout family, for example by providing an \@fxdefinelayoutkey command. We decided not to do this but it could be a legitimate choice. This is an illustration of the flexibility and perhaps also the difficulty there is to decide on the exact "evilness duplication threshold" mentioned earlier.

3.2.2 Copy-paste

Consider again the case of FiXme which defines several Boolean options. For each Boolean option foo, FiXme also provides a corresponding nofoo option, as a shortcut for foo=false. *E.g.* the langtrack / nolangtrack option can be defined as follows:

```
\@fxdefineboolkey{lang}{langtrack}[true]{}
\@fxdefinevoidkey{lang}{nolangtrack}{%
  \@nameuse{fx@lang@langtrack}{false}}
```

Defining the silent / nosilent option can be lazily done by *copy-pasting* the previous code and only modifying the relevant parts (the option and family names):

```
\@fxdefineboolkey{log}{silent}[true]{}
\@fxdefinevoidkey{log}{nosilent}{%
  \@nameuse{fx@log@silent}{false}}
```

This way of doing things obviously screams for abstraction. It is better to make the concept of "extended Boolean" explicit by providing a macro for creating them:

```
\newcommand*\@fxdefinexboolkey[3][]{%
  \@fxdefineboolkey{#2}{#3}[true]{#1}
  \@fxdefinevoidkey{#2}{no#3}{%
    \@nameuse{fx@#2@#3}{false}}}

\@fxdefinexboolkey{lang}{langtrack}
\@fxdefinexboolkey{log}{silent}
```

3.3 Rule #3: Conditionals are evil

This rule may sound surprising at a first glance, but experience proves that too many conditionals can hurt readability. In fact, this is well known in the object-oriented community. After all, object-orientation is essentially about removing explicit conditionals from code.

There are two main reasons why conditionals should be avoided whenever possible.

- First, too many conditionals, especially when they are nested, make the program's logic difficult to read.
- Second, the presence of multiple occurrences of the *same* conditional at different places is a form of duplication, and hence should be avoided.

One particular design pattern that helps a lot in removing explicit conditionals is to centralize the logic and use polymorphic macros. This is explained with the following example.

Figure 5 on the next page implements a macro \doeverything, the behavior of which depends on whether the document is in draft or final mode. This macro is in fact decomposed in three parts: the "do this" part, a middle part (left as a comment) and a final "do that" part. Because the same conditional

```
\newif\ifdraft

\def\doeverything{%
  \ifdraft
    \dothis\this\way
  \else
    \dothis\this\other\way
  \fi
  %% ...
  \ifdraft
    \dothat\that\way
  \else
    \dothat\that\other\way
  \fi}

\DeclareOption{draft}{\ifdrafttrue}
\DeclareOption{final}{\ifdraftfalse}
\ExecuteOptions{final}
\ProcessOptions
```

Figure 5: Conditional duplication

```
\def\dothis@draft{\this\way}
\def\dothis@final{\this\other\way}

\def\dothat@draft{\that\way}
\def\dothat@final{\that\other\way}

\def\doeverything{%
  \dothis
  %% ...
  \dothat}

\DeclareOption{draft}{
  \let\dothis\dothis@draft
  \let\dothat\dothat@draft}
\DeclareOption{final}{
  \let\dothis\dothis@final
  \let\dothat\dothat@final}
\ExecuteOptions{final}
\ProcessOptions
```

Figure 6: Centralized logic

branch clutters the code in two different places, the three-step nature of this macro is not very apparent.

A better and clearer implementation of the same functionality is proposed in figure 6. Here, the two mode-dependent parts of the `\doeverything` macro are explicitly implemented in different macros, with a postfix indicating in which mode they should be used. In the `\doeverything` macro, the three parts are now clearly visible. This new version of the macro is obviously much more concise and readable. The important thing to understand here is that when you read the code of `\doeverything`, you are in fact not concerned with implementation details such as how `\dothis` and `\dothat` vary according to the document's mode. It is more important to clearly distinguish the three steps involved.

Finally, you can also note that the logic involving conditionals is centralized at the end, where the actual `draft` or `final` options are processed. As a side note, the `\ifdraft` conditional is not needed anymore and the total amount of code is smaller in this new version. This time, clarity goes hand in hand with conciseness.

You may still be wondering what we meant by "polymorphic macros". Although slightly abusive, this term was coined because of the resemblance of this design pattern with object-oriented polymorphism, encountered in virtual methods à la C++ or generic functions à la Lisp. The macros `\dothis` and `\dothat` are polymorphic in the sense that they don't have a regular implementation (in other words, they are only *virtual*). Instead, their actual implementation varies according to some context.

3.4 Rule #4: Be modular

Modularity is another final principle which is rather obvious to follow, although it is perhaps even more crucial in LaTeX than in other programming languages. Modularity affects all levels of a document, from the author's text to the packages involved.

At the author's level, it is a good idea to use LaTeX's `\include` command and split your (large) source files into separate chunks. When used in conjunction with `\includeonly`, compilation may be considerably sped up by making LaTeX process only the parts on which you are currently working.

From a package development perspective, modularity is important at different levels. In terms of distribution, the `docstrip` package is an essential component in that it allows you to split your source code into separate files, provides conditional inclusion and (and perhaps most importantly) lets you generate separate distribution files from a centralized source. This is important because splitting a package across different files allows you to subsequently load only the relevant ones. Less code loaded into LaTeX means reduced memory footprint and improved performance. Imagine for instance if Beamer had to load every single theme every time it is run!

At a lower level, the modularity principle dictates that it is better to have 10 macros of 10 lines each rather than one macro of 100 lines. Every programmer knows this but perhaps LaTeX programmers don't realize that this is even more critical for them. There is indeed one LaTeX-specific reason for keeping your macros small. That reason is, again, intercession. Since other package developers may need to tweak *your* code for compatibility reasons, it is better to let them work on small chunks rather than big ones.

To illustrate this, let us mention the case of **CurVe** in which, at some point, we decided to support the `splitbib` package. In order to do so, we needed to override some parts of `splitbib`'s macro `\NMSB@writeentry`. This macro was originally 203 lines long. After dead branch removal, that is, after cutting out pieces of code that we knew would never be executed in the context of **CurVe**, we ended up with 156 lines that needed to be imported into **CurVe**, only to modify 5 of them. Our modifications consequently involve only 3% of the code that needed to be imported. One can easily imagine how bad this is in terms of maintainability. We need to keep track of potential modifications on 203 lines of `splitbib` just to make sure our 5 keep functioning correctly.

4 Level 3: Behavior

In the previous section, we claimed to be more concerned with how to implement particular features, rather than the features themselves. In this section, we focus on features through the lens of behavior. What we are interested in is the impact of your package features on the people that may interact with it.

4.1 Rule #1: Be user-friendly

The first category of people who will interact with your package is its end users. Hopefully, you belong to this category as well. There is, however, one major difference between you and other users: you know the package much better than they do, since you wrote it. Being user-friendly means doing everything possible to make your package easy to use. This can mean many different things, but two important aspects are documentation and backward compatibility.

4.1.1 Documentation

Nowadays, the vast majority of LaTeX packages comes with documentation. The combination of `doc`, `ltxdoc` and `docstrip`, by allowing for literate programming, has greatly helped the community in this respect. Nevertheless, there remains a huge difference between documentation and *good* documentation.

The difficulty in writing good documentation is to put yourself in the position of the casual user — which you are not because you know the package so well already. Thinking from a user perspective is probably the most difficult thing to do, but it can also be a very rewarding experience (we will get back to this later).

One of the major pitfalls to avoid when writing documentation is forgetting that a user manual is not the same thing as a reference manual. Just doing literate programming is not enough. Documenting your macros around their implementation is not enough. The casual user is not interested in the brute list of commands, nor in the internals of your package. The casual user wants an overview of the package, what it is for, what it can do, what it can't, probably a quick start guide describing the entry points and the default behavior, with examples. Only then, when the major concepts are understood, you may delve into complexity and start talking about customization, additional but less important features, and so on.

A good user manual will sacrifice sufficiency to the benefit of gradualness and redundancy. You shouldn't be afraid of lying by omission to the readers. It is for their own good. They don't want to be overwhelmed by information. A typical hint that you are reading a bad manual is when the documentation starts with the full list of package options. There is no point in introducing an option dealing with a concept that the reader does not understand yet (that would be a reference manual). Another hint is when a manual starts referring to another package (that it happens to use internally) and assumes that the reader knows everything about it already. The end user shouldn't have to read two or three other manuals to understand yours, especially if in the end, they will never use those other packages directly.

Why, as we said earlier, can it be rewarding to write a good manual? Because writing documentation is in fact a *feedback loop*. The difficult thing, again, is to put yourself in the position of someone who knows nothing about the things you are going to talk about, and ask yourself: "what do I need to say first?" If you can do that, you will discover that many times, the answers to that question reveal design flaws in your package, its design or its APIs. Things that a casual user would want to do but can't, things that should be simple to do but aren't, default behavior that shouldn't be by default, concepts that are not apparent enough, not distinct enough, names that are not sufficiently self-explanatory. *Etc.* other words, writing or improving the quality of your manual often helps you improve the quality of your code, and *vice-versa*.

4.1.2 Backward compatibility

Documentation is an important feature. Backward compatibility is another. Users can get very frustrated when a package breaks their documents from one version to another, or more generally, when a document doesn't compile anymore after a couple of years. This was in fact a concern that Donald Knuth had in mind at the very beginning of TeX and which had a considerable influence on the design of the LaTeX Project Public License [11], the LPPL.

Maintaining backward compatibility often goes against the "greater good". The natural evolution of a design might require a complete change of API, or at least an important amount of hidden trickery in order to stay compatible with the "old way". That is why it is all the more important to take great care with the design right from the start.

Just as in the case we made for modularity, the very high level of intercession in LaTeX makes backward compatibility an even more important concern. Because other developers will interfere with your code in order to fix compatibility or conflict problems between your package and theirs, the changes you make in your internals *will* affect them as well. So it turns out that backward compatibility is not only a surface concern, but also something to keep in mind even when working on the inner parts of your code. In the LaTeX world, nothing is really private... Of course, you may decide not to care about that, pretending that it's the "other guy's responsibility" to keep up to date with you, as he's the one who messes up with your code. But this is not a productive attitude, especially for the end user of *both* packages. In that regard, the following excerpt from `hyperref`'s README file is particularly eloquent:

> *There are too many problems with varioref. Nobody has time to sort them out. Therefore this package is now unsupported.*

In order to balance this rather pessimistic discourse, let us mention two cases where the burden of backward compatibility can be lightened. The first is the case of packages focused on the development phase of a document. FiXme is one of them. As it is mostly dedicated to handling collaborative annotations to draft documents, the cases where you would want to keep traces of it in a finished document are rare. Under those circumstances, we would not care about backward compatibility in FiXme as much as in other packages. For a document author perspective, it is very unwise to upgrade a LaTeX distribution in the middle of the writing process anyway...

When you decide that backward compatibility is too much of a burden, it is still possible to smooth the edges to some extent. Here is an idea that we are going to use for the next major version of ***CurVe***: change the name of the package, possibly by postfixing the (major) version number. In our case, the current version of ***CurVe*** (the `1.x` series) will be declared deprecated although still available for download, and the next version will be available in a package named `curve2`. This way, former ***CurVe*** documents will still compile in spite of backward incompatible changes to the newest versions.

```
\ExecuteOptionsX[my]<fam1,...>{opt1=def1,...}
\ProcessOptionsX*[my]<fam1,...>

\newcommand*\mysetup[1]{%
  \setkeys[my]{fam1,...}{#1}}

\newcommand\mymacro[2][]{%
  \setkeys[my]{fam1,...}{#1}%
  ...}
```

Figure 7: `xkeyval` programming example

Even if you do so, as a convenience to your users, it might still be a good idea to decorate your manual with a transition guide from one version to the next.

4.1.3 Key-value interfaces

We mentioned already the importance of feature design and the effect it can have on backward compatibility. The case of key-value interfaces is a typical example. Implementing package or macro options in a `key=value` style is a feature that every package should provide nowadays.

Key-value options are user-friendly because they are self-explanatory and allow you to provide a flexible API in a uniform syntax. It is better to have one macro with two options and 5 values for each rather than 25 macros, or 5 macros with 5 possible options.

As usual, the difficulty is in knowing all the existing alternatives for key-value interface, and choosing one. For that, Joseph Wright wrote a very useful paper that might help you get started [20]. Once you get used to it, programming in a key-value style is not so complicated.

Figure 7 demonstrates how easy it is to empower your package with key-value options both at the package level and at the macro level with `xkeyval`. Assuming you have defined a set of options, you can install a default behavior with a single macro call to `\ExecuteOptionsX`, and process `\usepackage` options with a single call to `\ProcessOptionsX`. Several packages provide a "setup" convenience macro that allows you to initialize options outside the call to `\usepackage`, or change the default settings at any time in the document. As you can see, such a macro is a one-liner. Similarly, supporting a key-value interface at the macro level is also a one-liner: a single call to `\setkeys` suffices.

In order to understand how key-value interfaces provide more flexibility and at the same time make backward compatibility less of a burden, consider one of the most frequently newbie-asked questions about LaTeX: how do I make a numbered section which does not appear in the table of contents (TOC)? The general answer is that you can't with the standard interface. You need to reprogram a sectioning macro with explicit manipulation of the section counter, *etc.*

We know that `\section` creates a numbered section which goes in the TOC. We also know that `\section*` creates an unnumbered section that does *not* go in the TOC. Finally, the optional argument to `\section` allows you to provide a TOC-specific (shorter) title. So it turns out that there's no standard way to extend the functionality in a backward-compatible way, without cluttering the syntax (either by creating a third macro, or by providing a new set of options in parentheses for instance). In fact, two macros for one sectioning command is already one too many.

Now imagine that key-value interfaces existed when `\section` was designed. We could have ended up with something like this:

```
% Number and TOC:
\section{Title}

% TOC-specific title:
\section[toctitle={Shorter Title}]{Title}

% Unnumbered and not in the TOC:
\section[numbered=false]{Title}
```

Obviously here, we intentionally reproduce the same design mistake as in the original version: assuming that unnumbered also implicitly means no TOC is suboptimal behavior. But in spite of this deficiency, when somebody wanted a numbered section not going in the TOC, we could have added a new option without breaking anything:

```
\section[toc=false]{Title}
```

What's more, we could also handle the opposite request for free: an unnumbered section still going in the TOC:

```
\section[numbered=false,toc=true]{Title}
```

4.2 Rule #2: Be hacker-friendly

The second category of people who will interact with your package is its "hackers", that is, the people that may need to examine your code or even modify it for intercession purposes. Of course, you are the first person to be in this category. Being hacker-friendly means doing everything possible to make your package easy to read, understand and modify. Note that as the first person in this category, you end up doing yourself a favor in the first place. Being hacker-friendly can mean many different things, including concerns that we have described already, such as modularity. In this section we would like to emphasize some higher level aspects, notably the general problem of code organization. In our experience, we find that organizing code in a bottom-up and feature-oriented way works best.

4.2.1 From bottom to top

Organizing code in a bottom-up fashion means that you build layers on top of layers and you organize those layers sequentially in the source file(s). The advantage in being bottom-up is that when people (including yourself) read the code, they can rely on the fact that what they see only depends on what has been defined above (previously). Learning seems to be essentially an incremental process. Reading is essentially a sequential process. Being bottom-up helps to conform to these cognitive aspects.

The bottom-up approach is sometimes confused with the design of a hierarchical model in which one tries to establish nested layers (or rings) of functionality. These are different things. For example, not all problems can be modeled in a hierarchical way and trying to impose hierarchy leads to a broken design. Sometimes, it is better to be modular than hierarchical. Some concepts are simply orthogonal to each other, without one being on top of the other. The bottom-up approach allows for that. When you have two orthogonal features to implement, you can just write them down one after the other, in no particular order. The only rule is that one feature depends only on the preceding, or more precisely, a subset of the preceding.

As a code organization principle, the bottom-up approach will also inevitably suffer from a few exceptions. Any reasonably complex program provides intermixed functionality that cannot be implemented in a bottom-up fashion. Macro inter-dependency (for instance, mutual recursion) is one such case. Another typical scenario is that of polymorphic macros (*cf.* figure 6 on page 320): you may need to use a polymorphic macro at a time when it hasn't been `\let` to its actual implementation yet. Those exceptions are unavoidable and are not to be feared. A short comment in the code can help the reader navigate through those detours.

4.2.2 Feature-oriented organization

In terms of code organization, the second principle to which we try to conform is arranging the code by feature instead of by implementation. This means that we have a tendency to think in terms of "what it does" rather than "how it does it" when we organize code sections in source files. In our case, this is a relatively recent change of perspective which, again, comes from the idea of putting oneself in the "hacker" position. When people need to look at your code, they are most of the time interested in one particular feature that they want to imitate, extend, modify or adapt for whatever reason. In such a situation, acquiring an understanding of the feature's inner workings is easier when all the code related to that feature is localized at the same place in the source.

To illustrate this, we will intentionally take an example which may be controversial: the case of internationalization. The CurVe class has several features which need to be internationalized: rubrics need a "continued" string in case they extend across several pages, bibliographic sections need a "List of Publications" title, *etc.* In CurVe 1, the code is already organized by feature, except for multi-lingual strings which are all grouped at the end, like this:

```
%% Implement rubrics
%% ...

%% Implement bibliography
%% ...

\DeclareOption{english}{%
  \continuedname{continued}
  \listpubname{List of Publications}}
\DeclareOption{french}{%
  \continuedname{suite}
  \listpubname{Liste des Publications}}
%% ...
```

These days, we find this unsatisfactory because the code for each feature is scattered in several places. For instance, the `\continuedname` macro really belongs to the rubrics section and hence should not appear at the end of the file. This kind of organization is indeed implementation-oriented instead of feature-oriented: we grouped all multi-lingual strings at the end because in terms of implementation, the idea is to define a bunch of `\<whatever>name` macros.

In CurVe 2, we will take a different approach, as illustrated below:

```
%% Implement rubrics
\newcommand*\continuedenglishname{%
  continued}
\newcommand*\continuedfrenchname{%
  suite}
%% ...

%% Implement bibliography
\newcommand*\listpubenglishname{%
  List of Publications}
\newcommand*\listpubfrenchname{%
  Liste des Publications}
%% ...

\DeclareOption{english}{%
  \def\@currlang{english}}
\DeclareOption{french}{%
  \def\@currlang{french}}
%% ...
```

After that, using the appropriate "continued" string is a matter of calling

```
\csname continued\@currlang name\endcsname
```

This new form of code organization has several advantages. First, *all* the code related to one specific feature is now localized in a single place. Next, it conforms better to the bottom-up approach (no forward reference to a multi-lingual string macro is needed). Finally, and perhaps unintentionally, we have improved the flexibility of our package: by implementing a macro such as `\@currlang`, we can provide the user with a means to dynamically change the current language right in the middle of a document, something that was not possible before (language processing was done when the package was loaded).

Earlier, we said that this example was taken intentionally because of its controversial nature. Indeed, one could object here that if someone wants to modify the multi-lingual strings, or say, support a new language, the first version is better because all internationalization macros are localized in a single place. It is true that if you consider internationalization as a *feature*, then our very own principle would dictate to use the first version. This is simply a demonstration that in general, there is no single classification scheme that can work for all purposes. However, we think that this argument is not really pertinent. If you would indeed want to modify all the multi-lingual strings, you would open the source file in Emacs and use the `occur` library to get all lines matching the regular expression

```
^\\newcommand\*\\.+name{
```

From the occurrence buffer, you can then reach every relevant line directly by hitting the `Return` key. This is really not complicated and in fact, could be more good programming advice: *know your tools.*

4.3 Rule #3: Use `filehook` for intercession

The final behavioral rule we would like to propose in this section deals more specifically with the intercession problem. We recently came up with a design pattern that we think helps smooth the implementation of inter-package compatibility.

4.3.1 Standard tools

The first thing we need to realize is that in general, the standard LaTeX tools are too limited.

`\@ifpackageloaded` allows you to detect when a package has been used or required, and possibly take counter-measures. However, this is only a *curative* way of doing things: it only lets you provide post-loading (*a posteriori*) code. What if you need to take precautionary measures instead?

`\AtBeginDocument` allows one to massively defer code execution until the beginning of a document, that is, after every package has been loaded. This is obviously a very gross granularity. For example, it doesn't provide any information on the order in which the packages have been loaded, something that might be needed even for post-preamble code.

Consider for example the following scenario:

- Style S calls `\AtBeginDocument{\things}`
- Class C loads style S

And ask yourself the following question: how does class C intercede on `\things`? There is no simple way to sort this out with the standard LaTeX tools.

4.3.2 `filehook`

Like probably almost every package developer, we have fought against these problems for years with intricate and obfuscated logic to fix inter-package compatibility. We think however that the very recent appearance of Martin Scharrer's `filehook` package is (should be) a crucial component in cleaning up the current intercession mess.

The `filehook` package provides pre- and post-loading hooks to files that you input in every possible way (`\include`'d files, packages, even class files). Thanks to that, one can now handle intercession in a context-free way, which is much better than what was possible before. For example, you can take both *a priori* and *a posteriori* counter-measures against any package, without even knowing for sure if the package is going to be loaded at all. This notably includes the possibility of saving and restoring functionality, much like what OpenGL does with its `PushMatrix` / `PopMatrix` or `PushAttrib` / `PopAttrib` functions (although OpenGL uses real stacks for this).

Eventually, the existence of `filehook` allowed us to come up with a particular design pattern for intercession management that can be summarized as follows. So far, this pattern works (for us) surprisingly well.

- First of all, start by writing your code as if there were no intercession problem. In other words, simply implement the default behavior as usual, assuming that no other package would be loaded.
- Next, handle compatibility problems with packages, one at a time, and only locally: use pre- and post-hooks *exclusively* to do so.
- Remember that hook code is only *potential*: none of it will be executed if the corresponding package is not loaded.
- Also, note that getting information on package loading order is now trivial if you use `\@ifpackageloaded` in a pre-hook.
- Avoid using `\AtBeginDocument` for intercession, unless absolutely necessary; for instance, if you need to take a counter-measure against a package that already uses it. Again, in such a case, calling `\AtBeginDocument` in a post-hook will allow you to plug in the relevant code at exactly the right position in the `\@begindocumenthook` chain.

4.3.3 Bibliography management in CurVe

To provide a concrete example of these ideas, let us demonstrate how recent versions of **CurVe** handle compatibility with various bibliography-related packages. A summarized version is given in figure 8 on the following page. Roughly speaking, what this code does is:

- install the default, **CurVe**-specific behavior first,
- step back if `bibentry` is loaded,
- merge with `multibib`,
- step back before `splitbib` and re-merge afterwards,
- render `hyperref` inoperative.

Knowing where we came from, that is, how this logic was done before `filehook`, it is amazing how readable, clear, concise, and in fact simple, this new implementation is. We cannot be sure how striking this will be for the unacquainted reader, but in case it is not, you are invited, as an exercise, to try an implement this only with the standard LaTeX macros (hint: it is practically impossible).

Earlier, we claimed that `filehook` would allow us to program in a context-free way. Let us explain now what we meant by that. First of all, note that because we use hooks exclusively to plug our intercession code, the five special cases could have been implemented in *any* order in the **CurVe** source file. We can move the five blocks around without modifying the semantics of the program. This is what we mean by being "context-free": the current dynamic state of the program has no effect on the code we put in hooks. Another instance of context freedom is in the specific case of `hyperref`. The important thing to notice here is that we save (and restore) *whatever* state we had just before loading `hyperref`.

```
%% Step 1: implement the default bibliographic behavior
%% ...

%% Backup LaTeX's original macros and replace them by our own:
\let\@curveltx@lbibitem\@lbibitem
\def\@curve@lbibitem[#1]#2{...}
\let\@lbibitem\@curve@lbibitem
%% ... do the same for \@bibitem, \bibitem etc.

%% Step 2: special cases

%% Bibentry. Restore standard definitions because bibentry just inlines
%% bibliographic contents.
\AtBeginOfPackageFile{bibentry}{
  \let\@lbibitem\@curveltx@lbibitem
  ...}

%% Multibbl. Merge its definition of \bibliography with ours.
\AtEndOfPackageFile{multibbl}{
  \def\bibliography##1##2##3{...}}

%% Splitbib.
%% Before: restore standard definitions because ours are only used as part of
%% the \endthebibliography redefinition.
\AtBeginOfPackageFile{splitbib}{
  \let\@lbibitem\@curveltx@lbibitem
  ...}
%% After: Modify \NMSB@writeentry and re-modify \endthebibliography back.
\AtEndOfPackageFile{splitbib}{
  \def\NMSB@writeentry##1##2##3##4##5,{...}%
  \def\endthebibliography{...}}

%% Hyperref. Currently, we don't want hyperref to modify our bibliographic
%% code, so we save and restore whatever bibliographic state we had before
%% hyperref was loaded.
\AtBeginOfPackageFile{hyperref}{
  \let\@curveprevious@lbibitem\@lbibitem
  ...}
\AtEndOfPackageFile{hyperref}{
  \let\@lbibitem\@curveprevious@lbibitem
  ...}
```

Figure 8: Intercession management

In other words, here again we don't need to know our exact context (the specific definition for the macros involved). We just need to save and restore it. And again, it is impossible to do that without the ability to hook code before and after package loading — which `filehook` now provides.

5 Level 4: Social

This section, shorter than the others, addresses a final level (or rather, a meta-level) in coding standards: the social level. Here, we are interested in how the human behavior may affect the LaTeX world and in particular the development of packages. We only provide two simple rules, and a rather unfortunate, but quite illustrative story.

We mentioned in the introduction the anti-social development syndrome that LaTeX seems to suffer from. In our opinion, this behavior is what leads to wheel-reinvention (*cf.* section 3.1 on page 318) and hence redundancy (for instance, the existence of half a dozen packages for key-value interfaces). In an ideal world, the situation could be improved by following the two simple rules described below.

5.1 Rule #1: Be proactive

The first rule of thumb is to permanently try to *trigger* collaboration. Package development frequently comes from the fact that you are missing a particular functionality. However, there is little chance that you are the first person to miss the functionality in question. Therefore, the first thing to do is to look for an existing solution instead of starting your own. By the way, we *know* that it is fun to start one's own solution. We have done that before, but it is nothing to be proud of!

Once you find an already existing solution (and you will, most of the time), it will probably not be an exact match. You will feel the need for implementing a variant or an extension of some kind. Here again, don't take this as an excuse to start your own work, and don't keep your work for yourself either. Try to be proactive and trigger collaboration: contact the original author and see if your ideas or your code could be merged in some way with the upstream branch. This is the first key to avoid redundancy.

5.2 Rule #2: Be reactive

Of course, this can only work if there is a response from the other side. And this is the second rule of thumb: if collaboration is proposed, *accept* it. Maintaining a package should be regarded as a certain responsibility towards its users. People frequently escape from their maintenance responsibility by hiding behind the free software banner (free as in freedom and/or as in beer). This is of course legitimate but also abused to the point of leading to the anti-social syndrome we have been discussing.

Being reactive means reviewing and accepting patches from other people in a reasonable time frame (for some definition of "reasonable"). It also means listening to other people's suggestions and implementing them within the same reasonable time frame. We understand that this is not always possible, but when you feel that there is some kind of pressure on you, there is also an alternative: *trust* people and *open* the development. Use a version control system (VC) of some kind. Put your code on `github` or a similar place and let people hack on it. The advantage to using a VC is that it is always possible to revert to an earlier state in the history of the package.

5.2.1 ***FiNK*** and `currfile`

We realize these considerations may be somewhat idealistic. In order to illustrate why they are important anyways let us tell a short story.

Sometime in 2010, we were contacted by Martin Scharrer, the author of `filehook`, about another package of his named `currfile`. This package maintains the name of the file currently being processed by LaTeX. Martin was inquiring about a potential cross-compatibility with ***FiNK***, one of our own packages, which does exactly the same thing.

We answered politely with the requested information and continued the email conversation for a little while, not without a wee bit of frustration however. Why yet another package for this? Wasn't ***FiNK*** good enough? Couldn't its functionality have been extended rather than duplicated?

Interestingly enough, we were recently sorting out some old mail when we dug up a message from this very same Martin Scharrer, providing a patch against ***FiNK*** in order to ground it onto `filehook`. This message was one year and thirty eight weeks old. Of course, we had completely forgotten all about it. In terms of time frame, one year and thirty eight weeks is way beyond "reasonable". So much for being reactive, lesson learned, the hard way...

6 Conclusion

In this paper, we addressed the notion of LaTeX coding standards. We started by analyzing the reasons why no such thing seems to exist as of yet. In short, the lack of coding standards for LaTeX can be justified by a mostly anti-social development syndrome, a not so pressing need for them in the view of the developers and the lack of help and support from the usual text editors. We however demonstrated that having a set of coding standards would be extremely beneficial to the community. First, they would help make the most of a programming language that is far less structured than the more casual ones. Next, they would also help in terms of code homogeneity and readability, both key components in collaboration. This is especially important in LaTeX because even if *intentional* collaboration is not so widespread, there *is* a very frequent form of *forced* collaboration, which is intercession (inter-package compatibility and conflict management).

We then reported on our own development experience and proposed a set of rules and design patterns that have helped improve our own code over the years. Those rules were organized in four different abstraction levels: layout (formatting and naming policies), design (modularity and other programming paradigms), behavior (interfaces and intercession management) and finally the meta-level (social behavior).

We don't expect that everyone would agree to every one of these rules, as we know that a coding style is above all a matter of personal taste. In fact, a coding style is important, but it is even more important to stick to it, that is, to stay coherent with yourself. Developing a coding style is also a matter of keeping the problem in mind permanently, not unlike a daemonized process running in the background of one's head. Every time you write a line of code, you need to ask yourself, "is this the proper way to do it?" This also means that a coding style may be a moving target, at least partially. It will evolve along with the quality of your code. Finally, one should remember that there can be no rule without exceptions. Knowing when to escape from your style for the greater good is as important as conforming to it.

The ideas, rules and design patterns proposed in this article are those that work best for us, but our hope is that they will also help you too. Many other ideas have not been tackled in this paper, both at the level of the document author and at the level of the package developer. Much more could be said on the matter, and if there is enough interest in the community, maybe it is time for an "Elements of LaTeX Programming Style" book which remains to be written. Perhaps this article could serve as a basis for such a book, and we would definitely be willing to work on such a project.

References

[1] AUC-TeX. `http://www.gnu.org/s/auctex`.

[2] The comprehensive TeX archive network. `http://www.ctan.org`.

[3] The XEmacs text editor. `http://www.xemacs.org`.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture*, volume 1. Wiley, 1996.

[5] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*, volume 4. Wiley, 2007.

[6] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*, volume 5. Wiley, 2007.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[8] B.W. Kernighan and P.J. Plauger. *The Elements of Programming Style.* McGraw-Hill, 1974.

[9] Michael Kircher and Prashant Jain. *Pattern-Oriented Software Architecture: Patterns for Resource Management*, volume 3. Wiley, 2004.

[10] Patty Maes. Concepts and experiments in computational reflection. In *OOPSLA*. ACM, December 1987.

[11] Frank Mittelbach. Reflections on the history of the LaTeX Project Public License (LPPL) — A software license for LaTeX and more. *TUGboat*, 32(1):83–94, 2011. `http://tug.org/TUGboat/tb32-1/tb100mitt.pdf`.

[12] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion,* second edition. Addison Wesley, 2004.

[13] Douglas C. Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, volume 2. Wiley, 2000.

[14] Brian C. Smith. Reflection and semantics in Lisp. In *Symposium on Principles of Programming Languages*, pages 23–35. ACM, 1984.

[15] Richard M. Stallman. The GNU coding standards. `http://www.gnu.org/prep/standards`.

[16] William Strunk Jr. and E.B. White. *The Elements of Style.* W.P. Humphrey, 1918.

[17] Didier Verna. The CurVe class. `http://www.lrde.epita.fr/~didier/software/latex.php#curve`.

[18] Didier Verna. The FiXme style. `http://www.lrde.epita.fr/~didier/software/latex.php#fixme`.

[19] Didier Verna. Classes, styles, conflicts: The biological realm of LaTeX. *TUGboat*, 31(2):162–172, 2010. `http://tug.org/TUGboat/tb31-2/tb98verna.pdf`.

[20] Joseph Wright and Christian Feuersänger. Implementing key–value input: An introduction. *TUGboat*, 30(1):110–122, 2009. `http://tug.org/TUGboat/tb30-1/tb94wright-keyval.pdf`.

◇ Didier Verna
EPITA / LRDE
14-16 rue Voltaire
94276 Le Kremlin-Bicêtre Cedex
France
`didier (at) lrde dot epita dot fr`
`http://www.lrde.epita.fr/~didier`

TUG 2011 abstracts

Editor's note: Many of the conference presentations are available at `http://www.river-valley.tv` in video form, thanks to Kaveh Bazargan and River Valley Technologies.

Kaveh Bazargan
Why TEX is more relevant now than ever
TEX is around 30 years old, and was conceived and written before the advent of laser printers, personal computers, PostScript and of course the Internet. At that time the idea of WYSIWYG document editing was just a futuristic idea. When people jumped on the WYSIWYG bandwagon, it was predicted that old technologies such as TEX which used mark-up for text would disappear in time. The advent of the Internet brought mark-up to the attention of the public. Somehow it was acceptable again. The recent move to the semantic web and HTML5 has brought renewed attention to mark-up and the need for clear structure in text. I suggest that we have gone full circle and now realise that mark-up is everything. And TEX, which has the most readable and minimalist mark-up, might just be the best tool today for structured documentation.

Dave Crossland
Freeing fonts for fun and profit
Google Web Fonts (`http://www.google.com/webfonts`) has published hundreds of libre fonts during the past year, at an accelerating pace. Dave Crossland has been driving this through consultancy for Google, and presents his personal opinion about the past, present and future of libre fonts — showcasing the latest designs, designers and tools. (Please note that this talk is entirely the personal opinion of Dave Crossland, and does not represent the views of Google, Inc., in any way.)

CV Radhakrishnan
TEX4ht — A Swiss army knife for TEX
There are several technologies to translate LATEX sources into other markup formats like HTML, XML and MathML. TEX4ht assumes a premier position among them owing to the fact that it makes use of the TEX compiler for translation, which helps to assimilate any complex author macros used in the document. This talk provides an overview of how to configure TEX4ht to output custom markup needed by users. More online at `http://www.cvr.cc/tex/tex4ht`.

Jean-luc Doumont
Integrating TEX and PDF seamlessly in pdfTEX
In its ability to generate graphical elements, TEX is basically limited to horizontal and vertical black rules. Extended versions such as pdfTEX add color options and, especially, the possibility to draw more freely on the page by inserting raw code (PDF code in the case of pdfTEX). Still, these two coding environments — TEX and PDF — are too often regarded as disjoint. It would be nice to integrate them seamlessly, for example, to use in PDF code a color or a dimension assigned or calculated in TEX. This presentation points out the challenges of such a consistent and transparent TEX–PDF integration, proposes a set of solutions, and illustrates how these solutions help create graphs flexibly or design pages consistently on a grid.

Frank Mittelbach
LATEX3 architecture

This talk discusses the architecture of LATEX3, starting with the initial ideas dating back to the early '90s. Using an example covering the whole production cycle it is shown that several different roles with different requirements are needed to turn some draft initial manuscript into a final product. The purpose of the LATEX3 architecture is to provide adequate support for these different needs and to resolve or at least mediate conflicts between them.

While the basic building blocks of this architecture were identified long ago, an initial implementation in 1992 showed that it was impossible to use them in practice due to limitations in the processing power of the underlying engines at the time. Furthermore, several ideas that were toyed with at the time — though not wrong as such — were immature and not fully thought through. As a result the project gave up on the broader redesign and instead focused on producing a consolidated LATEX version largely based on the architecture of LATEX 2.09. This fairly successful endeavor, labeled LATEX 2ε, is still the current standard LATEX.

So why is it still relevant? Basically because the drivers and goals that led to the new architecture are issues that haven't been successfully resolved by other typesetting systems. The difference from the situation from the '90s is that processing power in the underlying engine has increased so much that it has become feasible to implement this architecture in TEX (or rather one of its successors). The other reason is that since then further work has been undertaken, refining many of the initially immature ideas. The result is a coherent vision for a future typesetting system based on the principles of TEX and LATEX but moving them to the next level.

The talk discusses the separation of concerns as propagated by the architecture: between logical structure, design layer and the coding and implementation support. At the same time it is shown that for high-quality results this separation needs to be accompanied by built-in support for formatting adjustments and how this is supported by the architecture.

For design support the architecture provides two major complementary concepts: templates and context management. The use of design templates offers abstractions from which real designs can be derived through customization of parameters. The second approach is a general concept for managing design variations based on actual element relationships within a document. For each concept, the theory is discussed and a short live demonstration is given.

Ross Moore
Further advances toward Tagged PDF for mathematics

This is the 3rd presentation on on-going efforts to develop the ability to generated Tagged PDF output using pdfTEX, in conjunction with other software tools. In this talk I'll show how recent improvements to Adobe Reader and Adobe Acrobat Pro software have increased the usefulness of Tagged PDF documents, containing a MathML description of TEX-typeset mathematical content.

In particular, by careful specification of the words to be "Read Out Loud", mathematical content can be conveyed quite effectively to the visually impaired. Also, using Adobe's Acrobat Pro as the PDF browser, the ability to export to XML means that a fully marked-up, with MathML for the mathematics, version of the PDF document's contents can be obtained from the same file that displays the high-quality typeset visual appearance.

Examples will be shown of diverse mathematical content, generated automatically from standard LATEX, along with suitably generated MathML descriptions.

Rishi
Creating magical PDF documents with pdfTEX

PDF has a rich specification, but Adobe Distiller does not exploit all these specifications. We'll demonstrate how pdfTEX can create useful PDF files that are difficult or impossible to create using other technologies. Examples: PDFs showing differences in two TEX source files; PDFs with useful pop-up tools; a simple but useful composite PDF for comparing two nearly identical PDF files.

Karel Skoupý
Typesetting fancy multilingual phrase books with LuaTEX

We used TEX for typesetting a series of phrase books with a fancy graphical design. Each book contained the same content for a different language pair. There were several dozen of them semi-automatically generated, and thanks to the way that the language data was organized and thanks to TEX as a typesetting engine this process was very time and cost-effective.

We have developed interesting TEX macro modules and used many advanced features of pdfTEX and LuaTEX to meet the challenges raised by the graphical design and by some non-Latin script languages. We will show the general structure and discuss some interesting problems and their pdfTEX/LuaTEX solutions.

Karel Skoupý
Data structures in ε-TEX

To construct macro packages, TEX is used as a programming language. Unlike general programming languages it lacks complex data structures. We present the experience of providing record and array data structures and the supporting operations using ε-TEX features. They were successfully applied in real projects for parametrization and as a base for a special table module involving complex dimension calculations. We will show how the abstraction level provided by more powerful data structures can simplify and unify TEX low-level code.

Petr Sojka
Why TEX math search is more relevant now than ever

TEX is around 30 years old, and was conceived and written before the advent of MathML, not to mention the Internet. At that time the idea of indexing and searching mathematics was just a futuristic idea. When people jumped on the Google bandwagon, it was predicted that old technologies such as TEX mark-up for math would disappear in time (it is not used for tokenization and indexing properly). The advent of the Internet and W3C brought mark-up and global search to the attention of the public. Somehow it was acceptable again. The recent move to the semantic search and MathML has brought renewed attention to the need of unambiguous canonical math representation in texts.

As part of the project of building the European Digital Mathematics Library (`http://www.eudml.eu`) we have designed and implemented a math search engine, MIaS (`http://nlp.fi.muni.cz/projekty/eudml/mias`). It currently indexes and searches more than 160,000,000 formulae originally written by authors in TEX in their scientific papers. We will present the system and will discuss the ways towards a global math search engine based on the TEX math notation.

Dominik Wujastyk
My father's book: Typesetting and publishing a family memoir

In 2010, I typeset a 650-page book of memoirs, political essays, and biographical sketches written by my 97-year-old father. The book is in the Polish language, and was published by the University of Lublin. For the design and typesetting I made choices that stylistically echoed my father's life-long links with Malta and Poland. Due to financial restrictions at the University of Lublin, I worked out a cost-effective pathway for printing and distribution using an American web-based printing and distribution service. The final result is of a high standard, and has been gratifyingly well received by all parties. Some niggles remain, however, regarding publicity and distribution. In this paper, I shall describe my choices and discoveries in producing my father's book.

Dominik Wujastyk
Typesetting Sanskrit in various alphabets: XƎLATEX, TEC files, hyphenation, and even XML

The XƎTEX extended TEX engine provides a wealth of sophisticated features, and meets many of the longstanding needs of people working with multilingual or multi-script texts. I shall describe the use of XƎLATEX for typesetting Sanskrit, with both Roman- and Devānagarī-script inputs, and Roman- and Devānagarī-script outputs. I shall describe the complexities of getting differently hyphenated Sanskrit in different scripts. Finally, I shall offer an example of a free IBM XML tool that uses a XƎLATEX TEC file to auto-convert Sanskrit between Roman and Devānagarī for screen display via HTML. If all this sounds a bit messy, it is. But the results are sometimes quite amazing, and open up exciting possibilities for the beautiful printing of Indian texts.

LaTeX News

Issue 20, June 2011

Scheduled LaTeX bug-fix release

This issue of LaTeX News marks the first bug-fix release of LaTeX 2ε since shifting to a new build system in 2009. Provided sufficient changes are made each year, we expect to repeat such releases once per year to stay in sync with TeX Live. Due to the excitement of TeX's 2^5-th birthday last year, we missed our window of opportunity to do so for 2010. This situation has been rectified this year!

Continued development

The LaTeX 2ε program is no longer being actively developed, as any non-negligible changes now could have dramatic backwards compatibility issues with old documents. Similarly, new features cannot be added to the kernel since any new documents written now would then be incompatible with legacy versions of LaTeX.

The situation on the package level is quite different though. While most of us have stopped developing packages for LaTeX 2ε there are many contributing developers that continue to enrich LaTeX 2ε by providing or extending add-on packages with new or better functionality.

However, the LaTeX team certainly recognises that there are improvements to be made to the kernel code; over the last few years we have been working on building, expanding, and solidifying the expl3 programming layer for future LaTeX development. We are using expl3 to build new interfaces for package development and tools for document design. Progress here is continuing.

Release notes

In addition to a few small documentation fixes, the following changes have been made to the LaTeX 2ε code; in accordance with the philosophy of minimising forwards and backwards compatibility problems, most of these will not be noticeable to the regular LaTeX user.

Font subsets covered by Latin Modern and TeX Gyre The Latin Modern and TeX Gyre fonts are a modern suite of families based on the well-known Computer Modern and 'PostScript 16' families with many additional characters for high-quality multilingual typesetting.[1]

Information about their symbol coverage in the `TS1` encoding is now included in `textcomp`'s default font definitions.

Redefinition of `\enddocument` Inside the definition of `\end{document}` the `.aux` file is read back in to resolve cross-references and build the table of contents etc. From 2.09 days this was done using `\input` without any surrounding braces which could lead to some issues in boundary cases, especially if `\input` was redefined by some package. It was therefore changed to use LaTeX 2ε's internal name for this function. As a result, packages that modify `\enddocument` other than through the officially provided hooks may need to get updated.

Small improvement with split footnotes in `ftnright` If in the first column there is more than a full column worth of footnote material the material will be split resulting in footnotes out of order. This issue is now at least detected and generates an error but the algorithm used by the package is unable to gracefully handle it in an automated fashion (some alternatives for resolving the problem if it happens are given in the package documentation).

Improvement in `xspace` and font-switching The `xspace` package provides the command `\xspace` which attempts to be clever about inserting spaces automatically after user-defined control sequences. An important bug fix has been made to this command to correct its behaviour when used in conjunction with font-switching commands such as `\emph` and `\textbf`. Previously, writing

```
\newcommand\foo{foo\xspace}
... \emph{\foo}  bar baz
... \emph{\foo}, bar baz
```

would result in an extraneous space being inserted after 'foo' in both cases; this has now been corrected.

RTL in `multicol` The 1.7 release of `multicol` adds support for languages that are typeset right-to-left. For those languages the order of the columns on the page also needs to be reversed—something that wasn't possible in earlier releases.

The new feature is supported through the commands `\RLmulticolcolumns` (switching to right-to-left

[1]See their respective TUGboat articles for more information:
`http://www.tug.org/TUGboat/tb24-1/jackowski.pdf`
`http://www.tug.org/TUGboat/tb27-2/tb87hagen-gyre.pdf`

typesetting) and `\LRmulticolcolumns` (switching to left-to-right typesetting) the latter being the default.

Improve French `babel` interaction with `varioref`

Extracting and saving the page number turned out to be a source of subtle bugs. Initially it was done through an `\edef` with a bunch of `\expandafter` commands inside. This posed a problem if the page number itself contained code which needed protection (e.g., pr/4080) so this got changed in the last release to use `\protected@edef`. However, that in turn failed with Babel (bug report/4093) if the label contained active characters, e.g., a ":" in French. So now we use (after one failed attempt pr/4159) even more `\expandafter` commands and `\romannumeral` trickery to avoid any expansion other than what is absolutely required—making the code in that space absolutely unreadable.

```
\expandafter\def\expandafter#1\expandafter{%
\romannumeral
  \expandafter\expandafter\expandafter
\z@
\expandafter \@cdr
\romannumeral
  \expandafter\expandafter\expandafter
\z@
\csname r@#2\endcsname\@nil}%
```

Code like this nicely demonstrates the limitations in the programming layer of LaTeX 2_ε and the advantages that expl3 will offer on this level.

The meetingmins LaTeX class: Hierarchically organized meeting agendas and minutes

Brian D. Beitzel

Abstract

Many professionals (including faculty in higher education) must at least occasionally document the happenings of group meetings. Although a few different LaTeX classes are available for this purpose, the meetingmins class is simple and straightforward and most importantly, allows for a hierarchical organization of minutes using standard LaTeX `\section` commands. An agenda function is also available.

1 Introduction

Faculty in higher education and other professionals are often expected to compose a written record of group meetings. In addition, the agenda for these meetings is sometimes expected to be circulated in advance. A few LaTeX classes are available for formatting meeting minutes. Some are simple; others (I'm looking at you, minutes) are extraordinarily powerful but rather complicated. To the best of my knowledge, none integrates an agenda function.

The meetingmins class (`http://ctan.org/pkg/meetingmins`) takes a middle-of-the-road approach, providing a flexible document structure yet including all of the basics needed to chronicle the typical meeting. It is based on Jim Hefferon's mins class (`http://tug.org/pracjourn/2005-4/hefferon/`), which has a one-level (non-hierarchical) document structure. In departmental meetings at academic institutions, faculty report back from departmental committees as well as various institution-wide committees. Thus a hierarchical document structure (with each committee report being subordinate to either the department or the institution) is required to adequately represent the structure of the meeting.

2 Basic features

The nuts and bolts are all here, via commands in the document preamble: the group's name, meeting date, members present, members absent, and guests. The absentee and guest lines are not printed if they are not needed. There is also a `\nextmeeting` command that can be included at the end of the document to display the next meeting date. See the meetingmins documentation for details.

There are no pre-established sections within the body of the document; simply call the `\section` command in the standard way to create sections such as Announcements, Old Business, etc., titled and sequenced as you desire. Numbered items are available within any section by using an environment named `items`.

3 Distinctive features

3.1 Using `\section` commands to establish hierarchy

To transparently represent the hierarchical structure of the document, the standard LaTeX `\section` commands are used (down to `\subsubsection`). The document structure is then visually conveyed through the use of indentation and other formatting. More detail and examples are provided in the meetingmins documentation.

3.2 Agenda

Even the powerful minutes class does not support the creation of meeting agendas, so the lowly meetingmins steps in to fill the gap. To create an agenda, specify the `agenda` option when the class is loaded. The printed document will contain a skeleton agenda, titled "Agenda for ⟨*date*⟩" underneath the committee/department name. Numbered items of business will also be printed if they have not been suppressed (see next section).

3.3 Hidden items

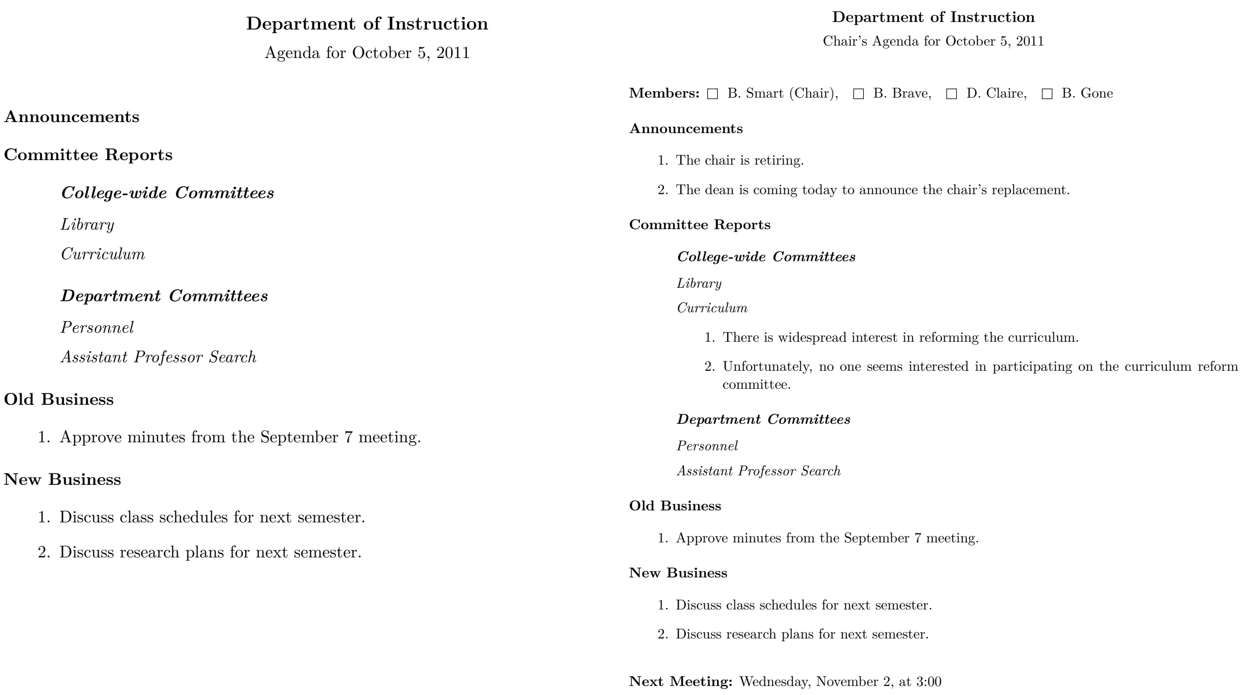
Agenda items can be suppressed from being printed by using the `hiddenitems` environment (in place of the `items` environment). No need to give away the surprise announcement before the meeting! The `hiddenitems` environment can be used in any section of the document. When the `agenda` option is removed from the `\documentclass` line to produce the minutes of the meeting, all items in `hiddenitems` environments will be printed; there is no need to alter environment names.

3.4 Chair's agenda

How many meetings have you attended (or led) in which you asked, "Who is missing?" With the chair's agenda, that question is moot. Specify the `chair` option (instead of `agenda` — don't use both) and a handy list of members will be printed at the top of the agenda, complete with checkboxes beside each name to facilitate taking attendance. And there are no surprises on the chair's agenda; `hiddenitems` environments are printed for easy reference by the chair throughout the meeting.

4 Sample documents

The meetingmins documentation includes complete samples for (a) an agenda containing some hidden items; (b) a chair's agenda; and (c) the meeting



Department of Instruction

Agenda for October 5, 2011

Announcements

Committee Reports

College-wide Committees

Library

Curriculum

Department Committees

Personnel

Assistant Professor Search

Old Business

1. Approve minutes from the September 7 meeting.

New Business

1. Discuss class schedules for next semester.
2. Discuss research plans for next semester.

Department of Instruction

Chair's Agenda for October 5, 2011

Members: ☐ B. Smart (Chair), ☐ B. Brave, ☐ D. Claire, ☐ B. Gone

Announcements

1. The chair is retiring.
2. The dean is coming today to announce the chair's replacement.

Committee Reports

College-wide Committees

Library

Curriculum

1. There is widespread interest in reforming the curriculum.
2. Unfortunately, no one seems interested in participating on the curriculum reform committee.

Department Committees

Personnel

Assistant Professor Search

Old Business

1. Approve minutes from the September 7 meeting.

New Business

1. Discuss class schedules for next semester.
2. Discuss research plans for next semester.

Next Meeting: Wednesday, November 2, at 3:00

Figure 1: Example participant and chair's agenda.

minutes. Users are encouraged to consult and modify these samples for their own use.

4.1 Example source

Here is the source for the output shown in Figure 1. The chair's agenda is created by replacing `agenda` with `chair` in the first line; no other changes.

```
\documentclass[11pt,agenda]{meetingmins}

\setcommittee{Department of Instruction}

\setmembers{
  \chair{B.~Smart},
  B.~Brave,
  D.~Claire,
  B.~Gone
}
\setdate{October 5, 2011}

\begin{document}
\maketitle

\section{Announcements}
\begin{hiddenitems}
\item
The chair is retiring.

\item
The dean is coming today to announce
the chair's replacement.
\end{hiddenitems}

\section{Committee Reports}

\subsection{College-wide Committees}
\subsubsection{Library}

\subsubsection{Curriculum}
\begin{hiddensubitems}
\item
There is widespread interest
in reforming the curriculum.

\item
Unfortunately, no one seems interested in
participating on the curriculum reform committee.
\end{hiddensubitems}

...

\nextmeeting{Wednesday, November 2, at 3:00}

\end{document}
```

◇ Brian D. Beitzel
129 Fitzelle
SUNY Oneonta
Oneonta, NY 13825 USA
brian (at) edpsych dot net
http://www.edpsych.net/brian/

Collaborative LaTeX writing with Google Docs

Igor Ruiz-Agundez

Abstract

Working with LaTeX documents is not an easy task and doing it collaboratively is even harder. The writing of an article by several authors at the same time implies extra coordination tasks to avoid unsynchronised versions, text overlapping or even loss of information. Collaborative writing platforms (e.g., Google Docs) are trying to solve this issue by enabling synchronous online writing for regular documents. Nevertheless, to our knowledge, there is no easy way to use this platform with LaTeX papers. Here we tailor a template and set of functions to enable collaborative work in LaTeX using Google Docs.

1 Introduction

Collaborative working technologies are very efficient tools. They encourage and facilitate team work. In recent years collaborative working platforms have become popular and their efficiency is well-proven [1]. Moreover, there are some works on collaborative writing of LaTeX documents [2, 6] but none of them provide a working template that enables the use of Google Docs as a writing platform.

Against this background, we introduce a template that enables the writing of collaborative LaTeX documents. Our basic approach will be to use Google Docs for editing, with a Makefile to update local files and run TeX.

The remainder of this paper is structured as follows. Section 2 introduces the requirements to use this template. Section 3 describes the files that are included in the template. Section 4 presents the functions included in the Makefile. Section 5 gives a full example of a collaborative work detailing all the required steps. Section 6 concludes and describes avenues for future work.

The template and functions described in this paper are available at:
`http://paginaspersonales.deusto.es/igor.ira/private$/collaborative-latex`.

2 Requirements

The use of this collaborative template requires some background in LaTeX [4], Google Docs [3], GNU Make [5] and GNU/Linux [7]. Nevertheless, this document aims to be self-contained and provide enough information to start creating collaborative documents using Google Docs. The reader will also need a user account on Google Docs.

3 Provided files

A description of the files provided in this collaborative LaTeX writing template:

- `template.pdf`: This paper itself.
- `CHANGELOG`: Tracking of the different versions, detailing the changes between them.
- `TODO`: Ideas for future improvements.
- `/template/`: A folder that contains an executable example of this template.
- `/template/Makefile`: A Makefile that contains all the executable commands to enable collaborative LaTeX writing. This file content will be detailed in Section 4.
- `/template/time-machine/`: A folder that contains a daily backup of the work.
- `/template/figures/`: A folder that contains the figures of this paper.
- `/template/src/`: A folder that contains the source code of this paper.
- `/template/template.tex`: The LaTeX source for this document.
- `/template/template.bib`: The BibTeX source for the references of this document.

4 Makefile description

Description of the Makefile configuration parameters:

- `FILE_TEX`: Name of the main TeX file.
- `DATESTAMP`: Syntax of date stamp for backups.
- `ACCOUNTTYPE`: Account type used to authenticate in Google Docs.
- `EMAIL`: Email account that identifies the author on Google Docs. It must have access to the shared document.
- `PASSWD`: The password associated to the previous user's email. If the user has the 2-step verification system enabled, an authorized application password is required.
- `SERVICE`: The type of service to use in Google Docs. It must be set to `writely`.
- `SOURCE`: Source domain of the request.
- `TEX_GOOGLE_DOCS`: TeX file resource identifier in Google Docs. In order to find the value of this resource, open the collaborative working document in Google Docs, copy and paste the document URL from your browser, and extract the resource id from the URL as in the following example:

  Sample document URL: `https://docs.google.com/document/d/123XX123XX/edit?hl=en_GB#`

  Resource id for this document: `123XX123XX`.

- `BIB_GOOGLE_DOCS`: BibTeX file resource identifier in Google Docs. This resource id is found in the same way as with `TEX_GOOGLE_DOCS`.

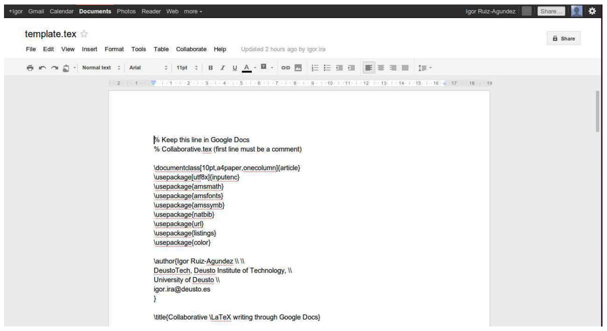
Next, we describe the Makefile functions (targets):

- `all`: Default execution function for the Makefile. Set to *latex*.
- `latex`: Compiles the document using *latex*, *bibtex* and *dvipdfm*. Performs a daily backup of the work. If working with indexes, a *makeindex* line can be uncommented.
- `pdflatex`: Compiles the document using *pdflatex* and *bibtex*. Otherwise like `latex`.
- `rtf`: Compiles the document using *latex*, *bibtex* and *latex2rtf*. Otherwise like `latex`.
- `view`: Opens the generated PDF file with the *evince* document viewer.
- `clean`: Cleans all the temporary working files generated in a compilation. It is used before each compilation in order to avoid possible errors from previous failed compilations.
- `update`: Update collaborative working documents, both TeX and BibTeX files, from the Google Docs version. This function overwrites your local files with the ones from Google Docs! Make sure you upload all your changes to the online version before executing it.

5 Collaborative working example

We are going to enumerate the steps required to perform a collaborative writing piece using this template:

1. Open a Google Docs document with extension `.tex`; we'll use `template.tex` for our example. Get the associated document resource as described in Section 4. Set the Makefile parameter `TEX_GOOGLE_DOCS` to this value. See Figure 1 for an example of editing in Google Docs.


+Igor Gmail Calendar Documents Photos Reader Web more
Igor Ruiz-Agundez
Share...
template.tex
Share
File Edit View Insert Format Tools Table Collaborate Help
Updated 2 hours ago by igor.ira
Normal text
Arial
11pt

```
% Keep this line in Google Docs
% Collaborative.tex (first line must be a comment)

\documentclass[10pt,a4paper,onecolumn]{article}
\usepackage[utf8x]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{natbib}
\usepackage{url}
\usepackage{listings}
\usepackage{color}

\author{Igor Ruiz-Agundez \\ \\
DeustoTech, Deusto Institute of Technology, \\
University of Deusto \\
igor.ira@deusto.es
}

\title{Collaborative \LaTeX writing through Google Docs}
```

Figure 1: TeX file editing in Google Docs

2. Similarly, open a Google Docs document with extension `.bib`; we'll use `template.bib`. Get the associated document resource id as described in Section 4. Set the Makefile parameter `BIB_GOOGLE_DOCS` to this value.
3. Set the other Makefile parameters as detailed in Section 4 with your personal configuration.
4. Give some initial content to `template.tex` in Google Docs. It is worth mentioning that the document will require an extra line at the beginning of the text. This extra line aims to avoid character encoding problems that may occur when importing the document with the Makefile. This first line will be automatically cleaned. We recommend setting this line to '`%Keep this line in Google Docs`', to remind authors that they must not delete it.
5. Give some initial content to `template.bib` in Google Docs. As in the case of the TeX file, this document will require an extra line in the beginning on Google Docs for the same reason. We recommend the same convention.
6. From your shell, run *make update* in the template folder to get the TeX and BibTeX documents from Google Docs.
7. Run *make* to compile the document, or *make latex*, *make pdflatex*, or *make rtf* to perform different compilations and obtain the corresponding output file formats.
8. Run *make view* to open the generated *pdf* document with the *evince* document viewer.

It is important to edit the files as stored on Google Docs and not the local copies. Otherwise, there would not be any collaboration between the authors and you could lose your contributions to the documents when updating your local files from your colleagues' work.

If at some point you cannot access Google Docs (e.g., you do not have Internet access), however, we can only recommend working locally and committing your changes online as soon as possible.

6 Conclusions

This article aims to provide a collaborative LaTeX writing context in Google Docs. It describes the requirements to start creating collaborative documents using this template, the provided files, the parameters and functions, and gives a full execution example.

Future work will focus on supporting various TeX files seamlessly; that is, with no need to edit the Makefile. In this way, documents that are split among different Google Docs files would update and compile automatically. If other collaborative writing platforms emerge, their support could also be included in this collaborative LaTeX template.

◇ Igor Ruiz-Agundez
DeustoTech, Deusto Institute of Technology, University of Deusto
Unibertsitate etorbidea 24
Bilbao, 48007
Basque Country
igor dot ira (at) deusto dot es
http://http://paginaspersonales.deusto.es/igor.ira/

References

[1] C. Clegg, P. Waterson, and N. Carey. Computer supported collaborative working: Lessons from elsewhere. *Journal of Information Technology*, 9(2):85–98, 1994.

[2] S. Dekeyser and R. Watson. Extending Google Docs to collaborate on research papers. *University of Southern Queensland, Australia*, 23:2008, 2006. Retrieved March 2011, Citeseer.

[3] Google. Google Docs — Online documents, spreadsheets, presentations, surveys, file storage and more. http://docs.google.com.

[4] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. The Not So Short Introduction to LaTeX 2ε. 2011. http://mirrors.ctan.org/info/lshort.

[5] Richard M. Stallman, Roland McGrath, and Paul D. Smith. GNU Make. http://www.gnu.org/software/make/manual.

[6] Wikibooks. LaTeX/Collaborative Writing of LaTeX Documents. http://en.wikibooks.org/wiki/LaTeX/Collaborative_Writing_of_LaTeX_Documents.

[7] Wikipedia. GNU/Linux naming controversy. http://en.wikipedia.org/wiki/GNU/Linux_naming_controversy.

A Appendix: Makefile template

```
# Makefile
# Author: Igor Ruiz-Agundez
# Affiliation: DeustoTech, Deusto Institute of Technology, University of Deusto
# Version: v.1.0

###
# TEX configuration
###
# Name of the main TEX file to work with
FILE_TEX=template

###
# Backup configuration
###
# Syntax of the date stamp for the backups
DATESTAMP=`date +'%Y-%m-%d'`

###
# Authentication parameters
###
# Account type that is used to authenticate in Google Docs
ACCOUNTTYPE=GOOGLE

# Email account that identifies the author on Google Docs. Must have access to the collaborative document.
# EMAIL=your-email-with-google-account
EMAIL=your-email-with-google-account

# The password associated to the previous user's email. Note that if the 2-step verification system is enabled an authorized application password is required.
# PASSWD=your-password
PASSWD=your-password

# The type of service to use in Google Docs. It must be set to writely:
SERVICE=writely

# Source domain of the request
SOURCE=deusto.es

###
# Google Docs resource ids
###
# To get the resource ids:
# Open the document with Google Docs
# Copy and paste the document URL from your browser
# Example:
# https://docs.google.com/document/d/123XXX123XXX/edit?hl=en_GB#
# In this example, the resource id is:
# 123XXX123XXX

# .tex file resource id
TEX_GOOGLE_DOCS=123XXX123XXX

# .bib file resource id
BIB_GOOGLE_DOCS=123XXX123XXX


###
# make execution functions
###

all: latex

latex: clean
  latex ${FILE_TEX}.tex
  # Uncomment makeindex runs if needed:
  # makeindex ${FILE_TEX}.nlo -s nomencl.ist -o ${FILE_TEX}.nls
  # makeindex ${FILE_TEX}
  bibtex ${FILE_TEX}
  latex ${FILE_TEX}.tex
  latex ${FILE_TEX}.tex
  dvipdfm ${FILE_TEX}.dvi
  # Backup tex, bib and generated pdf files
  # There is one backup per day
  mkdir -p time-machine/${DATESTAMP}
  cp ${FILE_TEX}.tex time-machine/${DATESTAMP}/${
       FILE_TEX}.tex
  cp ${FILE_TEX}.bib time-machine/${DATESTAMP}/${
       FILE_TEX}.bib
  cp ${FILE_TEX}.pdf time-machine/${DATESTAMP}/${
       FILE_TEX}.pdf

pdflatex: clean
  pdflatex ${FILE_TEX}.tex
  # Uncomment makeindex runs if needed:
  # makeindex ${FILE_TEX}.nlo -s nomencl.ist -o ${
       FILE_TEX}.nls
  # makeindex ${FILE_TEX}
  bibtex ${FILE_TEX}
  pdflatex ${FILE_TEX}.tex
  pdflatex ${FILE_TEX}.tex
  pdflatex ${FILE_TEX}.tex
  # Backup tex, bib and generated pdf files
  # There is one backup per day
  mkdir -p time-machine/${DATESTAMP}
  cp ${FILE_TEX}.tex time-machine/${DATESTAMP}/${
       FILE_TEX}.tex
  cp ${FILE_TEX}.bib time-machine/${DATESTAMP}/${
       FILE_TEX}.bib
  cp ${FILE_TEX}.pdf time-machine/${DATESTAMP}/${
       FILE_TEX}.pdf

rtf: clean
  latex ${FILE_TEX}.tex
  # Uncomment makeindex runs if needed:
  # makeindex ${FILE_TEX}.nlo -s nomencl.ist -o ${
       FILE_TEX}.nls
  # makeindex ${FILE_TEX}
  bibtex ${FILE_TEX}
  latex ${FILE_TEX}.tex
  latex ${FILE_TEX}.tex
  latex2rtf ${FILE_TEX}.tex
  # Backup tex, bib and generated rtf files
  # There is one backup per day
  mkdir -p time-machine/${DATESTAMP}
  cp ${FILE_TEX}.tex time-machine/${DATESTAMP}/${
       FILE_TEX}.tex
  cp ${FILE_TEX}.bib time-machine/${DATESTAMP}/${
       FILE_TEX}.bib
  cp ${FILE_TEX}.pdf time-machine/${DATESTAMP}/${
       FILE_TEX}.rtf

view:
  # Open the pdf document with evince
  evince ${FILE_TEX}.pdf &

clean:
  # Cleaning ${FILE_TEX} related files...
  ls ${FILE_TEX}.* | grep -v \.tex$ | grep -v \.bib$
       | grep -v \.ltx$ | xargs rm -fv
  # Cleaning other tex related files if applicable...
  rm -fv *log *aux *dvi *lof *lot *bit *idx *glo *bbl
        *ilg *toc *ind *blg *out *nlo *brf *nls *pdf
  # Cleaning in subdirectories *.aux files...
  find . -regex '.*.aux' -print0 | xargs -0 rm -rfv
  # Cleaning in subdirectories *.log files...
  find . -regex '.*.log' -print0 | xargs -0 rm -rfv
  # Cleaning in subdirectories *.bbl files...
  find . -regex '.*.bbl' -print0 | xargs -0 rm -rfv
  # Cleaning in subdirectories *.blg files...
  find . -regex '.*.blg' -print0 | xargs -0 rm -rfv
  # If there are other generated files, add the
       previous command again with the proper
       extension


update:
  # Create temporary file with the POST request
       configuration
  # Uses the authentication parameters of this
       Makefile
  echo "POST /accounts/ClientLogin HTTP/1.0\nContent-
       type: application/x-www-form-urlencoded\n\
       naccountType=${ACCOUNTTYPE}&Email=${EMAIL}&
       Passwd=${PASSWD}&service=${SERVICE}&source=${
       SOURCE}" > credentials.txt

  # Perform the authentication
  # Credentials are defined in Makefile
  # and temporally store in updater/credentials.txt
  wget -O clientLogin.txt --no-check-certificate --
       post-file=credentials.txt "https://www.google.
       com/accounts/ClientLogin" >/dev/null 2>&1

  # Remove client login information (for security
       reasons)
  rm credentials.txt

  ##
  # Get the TEX document
  ##

  # Get the document indicated by the first parameter
  wget --header "Authorization: GoogleLogin auth=`cat
        clientLogin.txt | grep Auth | sed "s#Auth=##"
        | xargs echo -n`" "https://docs.google.com/
       feeds/download/documents/Export?docID=${
       TEX_GOOGLE_DOCS}&exportFormat=txt" -O temp.txt

  # The first line of the downloaded line contains
       not valid characters
  # Remove first line of the downloaded document
  sed 1d temp.txt > ${FILE_TEX}.tex
  # Remove the temp file
  rm temp.txt

  ##
  # Get the BIB document
  ##

  # Get the document indicated by the first parameter
  wget --header "Authorization: GoogleLogin auth=`cat
        clientLogin.txt | grep Auth | sed "s#Auth=##"
        | xargs echo -n`" "https://docs.google.com/
       feeds/download/documents/Export?docID=${
       BIB_GOOGLE_DOCS}&exportFormat=txt" -O temp.txt

  # The first line of the downloaded line contains
       not valid characters
  # Remove first line of the downloaded document
  sed 1d temp.txt > ${FILE_TEX}.bib
  # Remove the temp file
  rm temp.txt

  # Remove client login information (for security
       reasons)
  rm clientLogin.txt
```

Glisterings

Peter Wilson

. . . Cloath'd all in glistering coats, which made a shew . . .

Poems and Fancies, Margaret Cavendish

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Corrections, suggestions, and contributions will always be welcome.

Sir, I have found you an argument, but I am not obliged to find you an understanding.

Samuel Johnson

1 Verbatim arguments

I have been reminded recently that one problem with verbatim material is that it cannot be used in an argument to a regular command (or environment). For example to typeset something in a framed `minipage` the obvious way is to use the `minipage` as the argument to the `\fbox` macro:

```
\fbox{\begin{minipage}{0.97\columnwidth}
      Contents of framed minipage
      \end{minipage}}
```

This works well until the contents includes some verbatim material and then you get nasty messages, even though it appears to be wrapped inside the `minipage`.

However, we can put material into a box, declared by `\newsavebox`, and output the typeset contents later on via `\usebox`. This is how the framed text below was processed.

This is the definition of the `framedminipage` environment which lets you put verbatim text into a frame. All this is set within a `framedminipage` to show that it does work.

```
\newsavebox{\minibox}
\newenvironment{framedminipage}[2][c]{%
  \begin{lrbox}{\minibox}
    \begin{minipage}[#1]{#2}}%
    {\end{minipage}\end{lrbox}
    \fbox{\usebox{\minibox}}}
```

I used `0.97\columnwidth` as the width of the environment like this:

```
\begin{framedminipage}%
      {0.97\columnwidth}
...
```

An `lrbox` is an environment form of a `\savebox` (or `\sbox`) and we can use it to solve the framed minipage problem. The code displayed above, after getting a new save box (`\minibox`) defines a `framedminipage` environment which is used just like a regular `minipage`, including the optional positioning argument. It starts by opening an `lrbox` environment, then a `minipage` environment. At the end it closes the `minipage` and `lrbox` environments and then typesets an `\fbox` whose argument is the saved box *the contents of which have already been typeset, verbatims and all.*

In *The TEXbook*, page 363, there is code for a `\footnote` macro that can take verbatim material in its argument. Knuth says that it is subtle and requires trickery, and I don't understand it, but here is the essence, in the form of a one argument macro I've called `\verbtext`. I'm not sure, though, about the location of the `\color@...` macros as there was nothing comparable in Knuth's original code

```
\makeatletter
\long\def\verbtext{\vtintro\futurelet\next\vte@t}
\def\vte@t{\ifcat\bgroup\noexpand\next
              \let\next\vt@@t
          \else \let\next\vt@t\fi \next}
\def\vt@@t{\bgroup\aftergroup\vtend\let\next}
\def\vt@t#1{%
  \color@begingroup
  #1\vtmid
  \color@endgroup}
\let\vtintro\relax
\let\vtmid\relax
\let\vtend\relax
\makeatother
```

The macros `\vtintro` and `\vtend` are called before and after the argument is read and you can try and define them to do something you think is useful. Defining `\vtmid` may, on occasion, be helpful.

So, here is an example of the `\verbtext` command, which can take verbatim text as part of its argument.

```
\verbtext{`The argument to \verb?\verbtext?
          can include \verb?\verb? text.'}
```

'The argument to `\verbtext` can include `\verb` text.'

The following code is a simple example of using `\vtintro` and `\vtend` to specify a small caps font.

```
\makeatletter
\newcommand*{\fred}[1][\@empty]{Frederick%
             \ifx\@empty #1\else\ #1\fi}
\makeatother
\def\vtintro{\begingroup\scshape}
\def\vtend{\endgroup}
\verbtext{The macro \verb?\fred[III]?
          produces \fred[III], while
          \verb?\fred? results in \fred.}
```

THE MACRO \fred[III] PRODUCES FREDERICK III, WHILE \fred RESULTS IN FREDERICK.

Actually this could have been done as easily as:

```
{\scshape\verbtext{...}}
```

without bothering to redefine \vtintro and \vtend, but perhaps you may come across occasions when they can help in solving a particular problem.

> Wickedness is always easier than virtue; for it takes a short cut to everything.
>
> SAMUEL JOHNSON

2 Cut off in its prime

Changing the subject, there was a question posed on comp.text.tex asking if there was any way of cutting a long text short, such as after two or three lines.

Donald Arseneau's truncate package [1] is available for truncating text to a specified width. By default ... (\ldots) is typeset at the end of the truncated text to indicate that something is missing. For instance

```
\truncate{0.9\columnwidth}{The
 \texttt{truncate} package provides a macro
  for cutting off text so that it does not
  exceed a given length.}
```

will result in:

The truncate package provides a macro for ...

However, in response to the query Donald came up with a vertical equivalent to \truncate which he called \vtruncate [2], as follows:

```
\newsavebox\descbox
\newsavebox\partialbox
\newcommand{\vtruncate}[2]{%
  \setbox\descbox\vbox{{#2\par}}%
  \setbox\partialbox\vsplit\descbox to #1\relax
  \vtop{\unvbox\partialbox}%
% or use
%   \par\unvbox\partialbox
}
```

The first argument is the vertical space and the second is the text.

Will Robertson also responded, but with an environment, cutlines, that would truncate its contents if it exceeded a certain height [3]. His definition was:

```
\makeatletter
\newbox\cut@desc
\newenvironment{cutlines}[1][2]{%
  \@tempcnta=#1\relax
  \setbox\cut@desc\vbox\bgroup
  \parskip=0pt}{%
  \egroup
  \vsplit\cut@desc to \@tempcnta\baselineskip}
\makeatother
```

The argument is the number of lines (default 2).

I tried both of these, and found potential problems with each:

1. The text argument to \vtruncate could not include any verbatim material (but this might not be of any concern).
2. If the number of lines specified for the cutlines environment was more than the lines in the original text, then the text was padded out with blank lines to make up the specified number.
3. In both cases the final truncated text was not always the specified height, but it was always to within plus or minus a line. However cutlines seemed to be more precise than \vtruncate.
4. The truncated text ends up in a box that cannot be split across a page boundary.

After some fiddling around[1] I came up with code for a truncate environment that was a mixture of Donald's and Will's code that seemed to avoid the first two of the four problems, and possibly the third as well. The fourth potential problem is inherent in all the proposals.

```
\newsavebox\descbox
\newsavebox\partialbox
\newlength{\vcutl}% for the limit height
\newlength{\Vcutl}% height of full text
\newenvironment{vcutlines}[1][2\baselineskip]{%
  \setlength{\vcutl}{#1}%
  \setbox\descbox\vbox\bgroup
  \parskip=0pt\relax
  }{%
  \egroup
  \Vcutl=\ht\descbox
  \advance\Vcutl \dp\descbox
  \setbox\partialbox\vsplit\descbox to
         \vcutl\relax
  \vtop{\unvbox\partialbox}
  \ifdim \vcutl<\Vcutl \vtruncont \fi}
\newcommand*{\vtruncont}{\noindent\strut\ldots}
```

In the following examples, the test text is:

```
{\itshape
Donald Arseneau created the \verb?\vtruncate?
command and Will Robertson the
\verb?cutlines? environment to truncate text
if it requires more than a specified height.
This is an example, though, of the new
\verb?vcutlines? environment --- a merge
of Donald's and Will's work.}
```

which does include a little verbatim material.

Let's give vcutlines a whirl with a limit of 20 lines (i.e., [20\baselineskip]).

[1] Quite a lot in fact.

Donald Arseneau created the `\vtruncate` *command and Will Robertson the* `cutlines` *environment to truncate text if it requires more than a specified height. This is an example, though, of the new* `vcutlines` *environment — a merge of Donald's and Will's work.*

And now the same text but with a limit of 3 lines (i.e., `[3\baselineskip]`).

Donald Arseneau created the `\vtruncate` *command and Will Robertson the* `cutlines` *environment to truncate text if it requires more than a spec-*

...

If the text is truncated, as in this example, then the environment finishes by calling the `\vtruncont` macro which by default outputs a final line consisting simply of ... (i.e., `\ldots`) to indicate that the original text continued. A comparison of the height of the original text with the specified height is used to decide if there was truncation.

You can change `\vtruncont` to typeset a different marker, or simply

```
\renewcommand*{\vtruncont}{}
```

to not do anything.

Here's a repeat of the last example:

Donald Arseneau created the `\vtruncate` *command and Will Robertson the* `cutlines` *environment to truncate text if it requires more than a specified height. This is an example, though, of the new*

However eliminating the marker this way seems to lead to a slight problem with the spacing after the end of the environment. Defining instead

```
\renewcommand*{\vtruncont}{\noindent}
```

Donald Arseneau created the `\vtruncate` *command and Will Robertson the* `cutlines` *environment to truncate text if it requires more than a specified height. This is an example, though, of the new*

Gives better spacing after the environment, as shown between this and the example immediately above.

References

[1] Donald Arseneau. `truncate.sty` truncate text to a specified width, 2001. `mirror.ctan.org/macros/latex/contrib/truncate`.

[2] Donald Arseneau. Re: How to limit/cut off text after a number of lines? Post to `comp.text.tex` newsgroup, 16 July 2008.

[3] Will Robertson. Re: How to limit/cut off text after a number of lines? Post to `comp.text.tex` newsgroup, 16 July 2008.

◇ Peter Wilson
12 Sovereign Close
Kenilworth CV8 1SQ, UK
`herries dot press (at) earthlink dot net`

Some LaTeX 2$_\varepsilon$ tricks and tips (IV)

Luca Merciadri

1 Introduction

As usual, in this article we shall give some LaTeX hints:

1. How to box an equation in an `align` (or its `align*` brother),
2. How to write a standard but elegant title page,
3. How to write text above and below an image,
4. How to modify spacing between lines,
5. How to write a left brace in a `subequations` environment.

2 Boxing an equation in an `align`-like environment

In [3], I wondered how I could *box an equation in an `align`-like environment*. Mr. Lars Madsen gave me the solution, also in [3], so thanks to him.

2.1 Example

Here is the seductive example of what you might want to achieve:

$$\begin{aligned} \boxed{A = B} \\ A &= B \\ &= C \end{aligned}$$

2.2 Code

The solution is to use the `calc`, and (evidently) the `amsmath` package too. Then, one can define `\Aboxed` like this:

```
\makeatletter
\newcommand\Aboxed[1]{%
  % syntax: \Aboxed{ left & right }
  \@Aboxed#1\ENDDNE}
 \def\@Aboxed#1&#2\ENDDNE{%
  % idea: get the left and right part
  % typeset them in a \boxed AFTER an ‘&’
  % and pull it backwards
  % but in order to get the horizontal
  % placement to work we need to set
  % some appropriate space to the left
  % of the ‘&’
  \settowidth\@tempdima{$\displaystyle#1{}$}%
  \setlength\@tempdima
    {\@tempdima+\fboxsep+\fboxrule}
  % \global does not always mix well
  % with \setlength
  \global\@tempdima=\@tempdima
  \kern\@tempdima
  &
  \kern-\@tempdima
  \boxed{#1#2}%
}
\makeatother
```

This can then be used in a document, like this:

```
\begin{align*}
   \Aboxed{A&=B}\\
   A&=B\\
   &=C
\end{align*}
```

If one wants to box an entire equation complex, the `empheq` package is a good choice.

3 A standard title page

The *title page* is the first page that will be seen on your document, so it has a strong influence on the (potentially future) reader. Doing it with care is a good but difficult thing.

3.1 Example

One can define a standard, but elegant, title page like the one which follows. (It has been scaled to *TUGboat*'s column width. Vertical spaces will adjust appropriately.)

YOUR UNIVERSITY

The name of your book
– in 1 pages, with 2 tables –

FirstName[1] LastName

City, Country

August 31, 2011

[1] FirstName.LastName@provider.domain

This is purely homemade, so I'm open to any suggestion(s) or remarks. This is a title page that I use for many booklets. For example, a slightly modified version of this can be found as the title page of [4]. I invite you to take a closer look at this title page.

Peter Wilson has developed a collection of title pages which can be found at [6]. This collection is worth reading.

3.2 Code

Here is the code which produced the expected title page.

```
\begin{titlepage}
\begin{center}

% Upper part of the page
\textsc{\LARGE YOUR UNIVERSITY}\\[1.5cm]

\includegraphics[width=0.50\textwidth]{img/%
your_university_logo.eps}\\[1cm]

% Title
%\HRule \\[0.4cm]
\rule{\textwidth}{1pt}\par
\vspace{0.50cm}
{\huge \bfseries The name of your book\\ %
\Large -- in \ref{TotPages} pages,
with \AbsTables ~tables  --}\\[0.4cm]
\rule{\textwidth}{1pt}\par
%\HRule \\[1.5cm]
\vfill

% Author
\Large{\textsc{FirstName\footnote{%
\href{mailto:FirstName.LastName@
provider.domain}{FirstName.LastName%
@provider.domain}}} LastName}
\vfill

City, Country
\vfill

% Bottom of the page
{\large \today}

\end{center}
\end{titlepage}
```

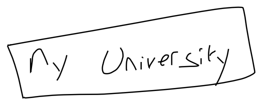
This can be put in your document, assuming the `hyperref`, `totpages`, and `graphicx` packages have been loaded before, having also defined `\AbsTables`. If `\AbsTables` is 1, you can use the `ifthen` package to modify "tables" to "table" automatically. (In the `memoir` class, the `totpages` package is not necessary, as `lastpage` and `lastsheet` are already defined.)

4 Text below an image

4.1 Example

One might appreciate being able to write this:

What could I say about it?
. . .



. . . Well, I won't say anything about it!

Figure 1: A caption

4.2 Code

This is achieved thanks to the following code, coming from [1].

```
\begin{figure}[!ht]
\centering
\parbox{0.25\linewidth}{%
Any Text that you want above \ldots\\%
[\smallskipamount]
\includegraphics[width=0.2\textwidth]%
{myuniv.eps}\\[\smallskipamount]
\ldots or below the image.
}
\caption{A caption}\label{fig:label1}
\end{figure}
```

5 Modifying spacing between lines

Many universities require double spacing to provide examiners with room for annotations. This can be achieved easily thanks to the `setspace` package [5].

5.1 Example

With normal spacing, we get

Maecenas dui. Aliquam volutpat auctor lorem. Cras placerat est vitae lectus. Curabitur massa lectus, rutrum euismod, dignissim ut, dapibus a, odio. Ut eros erat, vulputate ut, interdum non, porta eu, erat. Cras fermentum, felis in porta congue, velit leo facilisis odio, vitae consectetuer lorem quam vitae orci. Sed ultrices, pede eu placerat auctor, ante ligula rutrum tellus, vel posuere nibh lacus nec nibh. Maecenas laoreet dolor at enim. Donec molestie dolor nec metus. Vestibulum libero. Sed quis erat. Sed tristique. Duis pede leo, fermentum quis, consectetuer eget, vulputate sit amet, erat.

With `\doublespacing`, we get

Maecenas dui. Aliquam volutpat auctor lorem. Cras placerat est vitae lectus. Curabitur massa lectus, rutrum euismod, dignissim ut, dapibus a, odio. Ut eros erat, vulputate ut, interdum non, porta eu, erat. Cras fermentum, felis in porta congue, velit leo facilisis odio, vitae consectetuer lorem quam vitae orci. Sed ultrices, pede eu placerat auctor, ante ligula rutrum tellus, vel posuere nibh lacus nec nibh. Maecenas laoreet dolor at enim. Donec molestie dolor nec metus. Vestibulum libero. Sed quis erat. Sed tristique. Duis pede leo, fermentum quis, consectetuer eget, vulputate sit amet, erat.

5.2 Code

You need only include the `setspace` package and then select `\singlespacing`, `\onehalfspacing` or `\doublespacing`.

6 Writing a left brace in a `subequations` environment

6.1 Example

One may want a result like this:

$$\left\{\begin{aligned} a_1(x) &= b_1 \qquad (1a)\\ a_2(x) &= b_2 \qquad (1b)\\ a_3(x) &= b_3 \qquad (1c)\end{aligned}\right.$$

6.2 Code

This is achieved easily thanks to the inclusion of the `empheq` package, with the following code:

```
\begin{subequations}
\begin{empheq}[left=\empheqlbrace]{align}
     a_1(x) &= b_1\\
     a_2(x) &= b_2\\
     a_3(x) &= b_3
\end{empheq}
\end{subequations}
```

Thanks to Mr. Heller for this trick [2].

◇ Luca Merciadri
University of Liège
Luca.Merciadri (at) student dot ulg dot ac dot be
http://www.student.montefiore.ulg.ac.be/~merciadri/

References

[1] Donig, Thorsten. Giving source URL in Figure environment. LaTeX Community (Forum), 2009. http://www.latex-community.org/forum/viewtopic.php?f=44&t=5587.

[2] Heller, Martin and Maciel, Rui. Left brace on a subequations environment? comp.text.tex discussion), 2010.

[3] Madsen, Lars, Merciadri, Luca. How can I use `\boxed{}` in align environment? comp.text.tex discussion, 2010.

[4] Merciadri, Luca. Can a passive house be the solution to our energy problems, and particularly with solar energy?, 2008. Travail de fin d'études (secondaire); http://hdl.handle.net/2268/19645.

[5] Talbot, Nicola. Writing a Thesis in LaTeX: hints, tips and advice, 2010. http://uk.tug.org/wp-content/uploads/2009/01/talbot_slides.pdf.

[6] Wilson, Peter. Some Examples of Title Pages, 2010. http://ctan.org/pkg/titlepages.

TeX as you like it: The Interpreter package

Paul Isambert

Introduction

This article presents the Interpreter package for LuaTeX, designed to preprocess input files on the fly so that the user can map any syntax to proper TeX and type documents with the language s/he finds more convenient.

This is not a comprehensive description of Interpreter, but only highlights of its functionality; the documentation accompanying the package on CTAN remains the ultimate reference, and contains a complete explanation of an interpretation file.

Motivations

Despite loving TeX, I've always hated typing backslashes and braces for some reason (for one, they're rather badly placed on a French keyboard). At least, the latter can often be avoided thanks to delimited arguments, but unless one is willing to define a new character with catcode 0 (something I find almost counterintuitive) or to venture into the dangerous world of active characters, backslashes cannot be avoided.

Also, I've always found TeX source files (mine and others's) quite unreadable. The likes of `\macro` and `\com{mand}` disturb the normal flow of reading. This became more striking still when I started using the Vim editor. Unlike most text editors, Vim's documentation is made of plain text files meant to be read in the editor itself (this is also true of Emacs); thus one can remain in the same working environment and above all browse the help files as one usually browses some code. If only one could read TeX source files so easily!

All in all, what I wanted was to type TeX source in a syntax unrelated to TeX — a lightweight markup language like Markdown or the syntax used for wikis. Without LuaTeX, the only solution (as far as I know) is to use some script to convert a file into proper TeX (thus creating another file), something I've never tried. With LuaTeX, things change: if you want to preprocess a file on the fly before feeding it to TeX, you can do it, just hook into the `open_read_file` callback!*

* This is the reason why Interpreter doesn't work with ConTeXt, in which the callback is frozen. ConTeXt does have modules to process some non-TeX languages, but I'm not aware of a general solution for any language the user might want to define.

Working principles

The basic mechanism behind Interpreter is quite simple; you have a master file in which you input the file(s) to be preprocessed with:

`\interpretfile{`⟨*lang*⟩`}{`⟨*file*⟩`}`

where ⟨*lang*⟩ points to an external file containing the interpretation (explained in the following section). Then Interpreter uses the `open_read_file` callback to control how the lines of ⟨*file*⟩ are to be fed to TeX. This callback is passed a string representing the file to read and should return a table with two entries: `reader`, a function called whenever TeX wants a line, and (optionally) `close`, a function executed when the end of the input file is reached. The simplest implementation of `reader` is to read a line of the input file and return it to TeX; before that, however, one can also modify that line or read others (and perhaps modify them too), which is exactly how Interpreter works. Some practical examples: one can ask Interpreter to change '`some *bold* text`' into '`some \bold{bold} text`' or

```
=========================
=== A section heading ===
=========================
```

into

```
\section{A section heading}
```

or to surround with verbatim macros any material indented with ten spaces, and stop interpreting it at once (so the material is really left verbatim).

Defining simple patterns

As already mentioned, `\interpretfile` will look for an external file matching its first argument; more precisely, if that argument is e.g. `lang`, then the file should be called `i-lang.lua`. It contains all the replacements that will take place to convert the input file; as the extension indicates, the language is Lua. The main function is `interpreter.add_pattern()`, which takes a table defining a pattern to be searched for and replaced with something else. Not surprisingly, one of the entries is `pattern`; another is `replace`; and Interpreter will try to find all material matching the former and replace them with the latter.

For instance, the following will replace `/text/` with `\italic{text}`:†

† Since the function's single argument is a table, Lua allows the parentheses to be omitted; e.g. `myfn({mytab})` and `myfn{mytab}` are equivalent in such cases. The same is true for strings: `myfn"mystr"` works in the same circumstances.

```
interpreter.add_pattern{
  pattern = "/(.-)/",
  replace = "\\italic{%1}"
}
```

The reader will notice that Lua's magic characters are used, and `(.-)` thus means 'capture the shortest possible sequence made of any number of matching characters', and not a dot followed by a minus sign between parentheses. To denote a magic character itself, one should prefix it with `%`; thus if one wanted to use stars instead of slashes, the pattern should be `%*(.-)%*`, because the star is a magic character (see the Lua reference manual for the list of magic characters). Alternatively, Interpreter has a function `interpreter.nomagic()` which reverses the magic: no character is magic unless prefixed with `%`, except that `...` means the magic `(.-)`. For example, `interpreter.nomagic("*...*")` is equivalent to `"%*(.-)%*"`.

I've mentioned captures, and indeed `replace` makes use of them: `%1` refers to the first (and in this case, only) capture of `pattern`. This follows the behavior of Lua's `string.gsub()`, since ultimately Interpreter uses that function to make the replacement. Accordingly, `replace` can be a string, as is the case here, but also a table (and the entry returned is the one with the first capture, or the entire match if there is no capture, as its key) or a function (to which the captures, or the entire match, are passed as arguments).

Now Interpreter will search all lines for the specified pattern and use the replacement if a match occurs; a limitation (and security) is that matches must be contained in a single line. For instance, the following material will be left untouched:

```
This will /not be
put/ in italics.
```

To span several lines, two solutions are possible. First, one can redefine the pattern to match a single slash, which is converted to `\italics{` or `}` depending on a conditional. To do this, one can use a function in `replace`:

```
local italic
local function makeitalic ()
  if italic then
    italic = false
    return "}"
  else
    italic = true
    return "\\italic{"
  end
end
```

```
interpreter.add_pattern{
  pattern = "/",
  replace = makeitalic
}
```

The second solution, sounder and more general, will be explained in the next section.

Before turning to more advanced topics, a word of caution: Interpreter does *not* define TeX macros as `\italic` or `\bold` or `\section`. They are used here because their meaning is clear, but one should obviously use macros defined elsewhere. In other words, Interpreter simply manipulates strings and has nothing to do with typesetting.

Handling paragraphs

Simple patterns are fine as far as they go, but sometimes manipulating input line by line doesn't suffice. For instance, suppose you want to turn

```
1. First item.
2. Second item.
3. Third item.
```

into something like

```
\list
\item First item.
\item Second item.
\item Third item.
\endlist
```

Converting a string of digits followed by a dot at the beginning of a line into `\item` is easy enough. However, how should `\list` and `\endlist` be added to the material?

Such a situation is the reason why Interpreter manipulates paragraphs instead of lines. Instead of fetching a line, converting it according to the defined patterns, and returning it to TeX, Interpreter collects an entire paragraph, does all the conversions, and only then passes it line by line to TeX. In the meantime new lines might have been added.

For Interpreter a paragraph is anything up to and including the first line matching completely the pattern stored in `interpreter.paragraph`, where 'matching completely' means that if the material matching the pattern is removed from the line, the line is empty. By default, `interpreter.paragraph` is defined as `%s*`, i.e. a paragraph is marked by a line containing at most spaces.

To manipulate paragraphs, one should define a pattern with a `call` entry. This should be a function, and it will be executed as follows:

```
function (paragraph, line, index, pattern)
```

The first argument is the entire paragraph where the match occurred. It is represented as a table with numerical indices; `line` is the index of the line where the match occurred, so that `paragraph[line]` returns a string representing that line; `index` is the position in that string where the match was found; finally, `pattern` is the entire table which has been defined with `interpreter.add_pattern`.

Our situation with lists could be solved like this:

```
local item = "^%s*%d+%.%s*"
local function makelist (paragraph)
  for n, l in ipairs(paragraph) do
    paragraph[n] = string.gsub(l, item,
                                   "\\item ",1)
  end
  table.insert(paragraph, 1, "\\list")
  table.insert(paragraph, "\\endlist")
end
add_pattern{
  pattern = item,
  call    = makelist
}
```

The following will happen: when Interpreter spots a string of one or more digits followed by a dot at the beginning of a line (spaces notwithstanding), it calls the `makelist` function. This functions searches for the same pattern in all the lines of the paragraph and replaces it with `\item`; also, it inserts new lines with `\list` and `\endlist` at the beginning and the end of the paragraph.

This example used only the first argument of the `call` function. As a more complicated case using all four arguments, let's solve the question of defining `/.../` as a marker for italics possibly spanning several lines. Basically, the solution is identical to the one shown in the previous section: the first slash should be turned into `\italic{` and the second into `}`. But, as already mentioned this solution will be sounder, because the conversion will be done if and only if a pair of slashes is found (so that a slash on its own isn't modified), and also more general, because the same function will be used for all similar patterns.

```
local match,gsub = string.match,string.gsub
local function markup (par, line, index,
                                    pattern)
 local patt = pattern.pattern
 local rep = "\\" .. pattern.replace .. "{"
 if match(par[line], patt, index+1) then
  par[line] = gsub(par[line], patt, rep, 1)
  par[line] = gsub(par[line], patt, "}", 1)
 else
  local n = line+1
  while par[n] do
   if match(par[n], patt) then
    par[line] = gsub(par[line], patt,
                     rep, 1)
    par[n] = gsub(par[n], patt, "}", 1)
    return
   else
    n = n+1
   end
  end
  return index+1
 end
end
interpreter.add_pattern{pattern = "/",
      call = markup, replace = "italic"}
interpreter.add_pattern{pattern = "%*",
      call = markup, replace = "bold"}
interpreter.add_pattern{pattern = '"',
      call = markup, replace = "quote"}
```

Given a pattern, the `markup` function looks for another occurrence of this pattern in the same line or in the following lines of the paragraph. Only if the search succeeds does the replacement happen. Then we specify patterns so `/text/` will be replaced with `\italic{text}`, `*text*` with `\bold{text}`, and finally `"text"` with `\quote{text}`.

Two things should be remarked upon in the code above. First, the line `return index+1` at the end of the function instructs Interpreter to resume its search for patterns at the next position in the current line; without it, the search would start again at the same position where the pattern was found. This `return` statement occurs if no matching character was found, i.e. if the pattern was launched on a lonely slash (or star or double quote). Thus that character was *not* converted, and if the search were to start again at the same position, Interpreter would find the same character, and enter a loop.

Second, the patterns store the macro to be used in the `replace` field. That is totally arbitrary: the table making up the pattern can contain any field. Here the `replace` entry can be used because if a pattern has both `call` and `replace`, the latter is ignored (i.e. the mechanism described in the previous section doesn't apply).

Bells and whistles

As said in the introduction, this paper is not a complete manual for Interpreter. Here I'll mention a few other bits of functionality.

First and foremost, the search for patterns is done according to an order. Each pattern belongs to a class, as specified by the `class` entry in the pattern table (this entry defaults to the number recorded in `interpreter.default_class`), and classes are applied one after the other in ascending order; patterns belonging to the same class are ordered by length and are applied from longer to shorter.

One of the reasons why classes are important is so input can be protected, i.e. prevent Interpreter from converting some lines or an entire paragraph. For instance, consider a pattern denoting verbatim material. It will launch a `call` function to add something like `\verbatim` and `\endverbatim` as the first and last lines. In addition, it should call the function `interpreter.protect()`, to stop Interpreter from manipulating the current paragraph, i.e. other patterns won't be searched for and replaced, as expected for verbatim material. Such a pattern should belong to the very first class, so that it is executed before all the others; otherwise, protection would be only partial.

Another way to protect input, this time locally, is to record a pair of strings as left and right markers such that the enclosed material shouldn't be touched. The function `interpreter.protector()` does this; e.g. after `interpreter.protector('"')`, material between double quotes will be left intact (if the function is called with only one argument, it is used for both the left and right markers).

Interpreter allows you to mix different syntaxes, or rather, it has no notion of well-formedness for the language you define. Thus usual TEX commands can be used in the middle of an interpreted file. One convenient trick is to define an easy syntax to add new patterns to the file being read. For instance, with

```
interpreter.add_pattern{
  class   = 1,
  pattern = "^DEF%s*(.-)%s*=%s*(.+)",
  replace = function (pat, rep)
      interpreter.add_pattern{
        pattern = pat,
        replace = rep}
        return ""
      end
}
```

simple new patterns can be created as follows:

```
DEF pattern = replacement text
```

I let the reader check that it works properly.

Input files that look like main files

One of Interpreter's limitations is that it works only on input files: it can't work with a main file directly fed to TEX. The reason for this is that it uses the `open_read_file` callback which, as its name implies, concerns `\input` and `\read` (`\interpretfile` ultimately boils down to `\input`).

The main file could be manipulated with the `process_input_buffer` callback, but this isn't as flexible as `open_read_file`, and most importantly it doesn't attach to a specific file. Yet one can have the impression to work on the main file by invoking LuaTEX as follows (this works for plain TEX; LATEX users should adapt accordingly):

```
luatex -jobname=⟨file⟩
       \input interpreter
       \interpretfile{⟨language⟩}{⟨file⟩}
       \bye
```

The important point is to set `-jobname` to the input filename, so the relevant output files (the PDF, log, etc.) are created with the proper name. It might be wise to set `-output-directory` to the file's directory too, but that is not necessary.

Of course, one can also input other files besides Interpreter. It might also be interesting to use a Lua initialization script, an alternative I won't investigate here.

Conclusion

LuaTEX keeps changing my TEX world every day: new horizons, new solutions, a new language. With Interpreter, even my usual TEX source doesn't look the same! I'm even working on a document where unmarked macros represent themselves, i.e. `\macro` is turned to `\string\macro`.

One thing I do not know is whether Interpreter would be convenient to process something like XML; not using that language, I haven't tried to create an interpretation file for it, and I wonder whether Interpreter would be up to the task or would rather get in the way. If the reader finds a solution, just let me know!

◇ Paul Isambert
zappathustra (at) free dot fr

Postscript: Just before this article went to press, Interpreter was rewritten with the Gates package. None of what is said here needs updating, since Interpreter hasn't changed on the surface, but its implementation now allows deep hacking, because it is made of small logical steps that can be externally controlled. Details can be found in the new version's documentation and general principles in the documentation for Gates.

PARCAT — Applying TEX in industry

Wiktor Dziubiński, Marcin Woliński and Grzegorz Murzynowski

Abstract

PARCAT is highly sophisticated software for managing a company's product database and producing a printed catalogue alongside an online shop web site in an automated way. To produce the printed catalogues PARCAT employs XƎLATEX.

In this article we present PARCAT in general and its TEX back-end, describing some of its features in more detail:

- fitting tables to the column width,
- catalogue layouts,
- modularisation of the code,
- system of the layout parameters.

1 Introduction

The PARCAT system is a complex database application whose primary purpose is to comprehensively manage descriptions of products. Descriptions can be given in any language or in several languages. The main distinctive feature of the program is the ability to typeset a catalogue fully automatically. The generated catalogue can be passed to the printing houses in PDF format without any additional treatment. In addition to generating PDF files, PARCAT also prepares files ready for use on web sites. Due to this feature the system can be used as the base tool for managing product descriptions.

Another distinctive and, we dare say, revolutionary feature of PARCAT is its ability to generate several language versions of a catalogue into a single PDF with each language version on a separate layer, which in the printing house can correspond to one black plate, with no need to prepare separate full CMYK plates for each language.

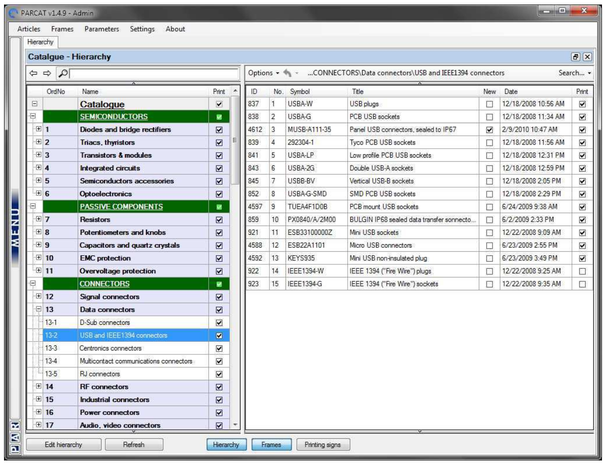
2 History

PARCAT was developed for a large trade company operating for 20 years on the market of electronic components. For over a dozen years Transfer Multisort Elektronik (TME), the company in question, has been publishing the catalogue of its products, in 2011 reaching eight language versions containing about 1800 pages each. The catalogue was made by using CorelDraw software which took approximately five weeks for one language version, excluding the preparation of images. On its web site, the company also presented the same data from the catalogue now in the form of HTML files prepared without any automation. Thus it became obvious that the dynamic development of the company would be limited without tools to manage product descriptions.

This article is a combination of articles from the BachoTEX 2010 and the EuroTEX 2011 proceedings.

PARCAT has been developed continuously since early 2008. After its successful deployment in TME it was decided to continue the development of the software and to try to prepare a universal version which could be useful for other companies as well.

2.1 The structure of a catalogue



	OrdNo	Name	Print
⊟		Catalogue	✓
⊟		SEMICONDUCTORS	✓
⊞	1	Diodes and bridge rectifiers	✓
⊞	2	Triacs, thyristors	✓
⊞	3	Transistors & modules	✓
⊞	4	Integrated circuits	✓
⊞	5	Semiconductors accessories	✓
⊞	6	Optoelectronics	✓
⊟		PASSIVE COMPONENTS	✓
⊞	7	Resistors	✓
⊞	8	Potentiometers and knobs	✓
⊞	9	Capacitors and quartz crystals	✓
⊞	10	EMC protection	✓
⊞	11	Overvoltage protection	✓
⊟		CONNECTORS	✓
⊞	12	Signal connectors	✓
⊟	13	Data connectors	✓
	13-1	D-Sub connectors	✓
	13-2	USB and IEEE1394 connectors	✓
	13-3	Centronics connectors	✓
	13-4	Multicontact communications connectors	✓
	13-5	RJ connectors	✓
⊞	14	RF connectors	✓
⊞	15	Industrial connectors	✓
⊞	16	Power connectors	✓
⊞	17	Audio, video connectors	✓

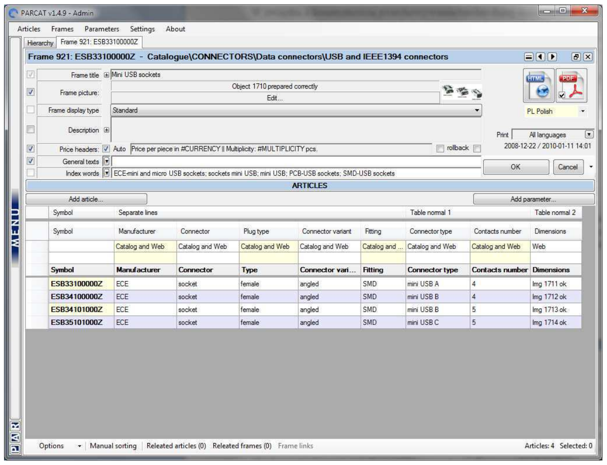
ID	No.	Symbol	Title	New	Date	Print
837	1	USBA-W	USB plugs	☐	12/18/2008 10:56 AM	✓
838	2	USBA-G	PCB USB sockets	☐	12/18/2008 11:34 AM	✓
4612	3	MUSB-A111-35	Panel USB connectors, sealed to IP67	✓	2/9/2010 10:47 AM	✓
839	4	292304-1	Tyco PCB USB sockets	☐	12/18/2008 11:56 AM	✓
841	5	USBA-LP	Low profile PCB USB sockets	☐	12/18/2008 12:31 PM	✓
843	6	USBA-2G	Double USB-A sockets	☐	12/18/2008 12:59 PM	✓
845	7	USBB-BV	Vertical USB-B sockets	☐	12/18/2008 2:05 PM	✓
852	8	USBA-G-SMD	SMD PCB USB sockets	☐	12/18/2008 2:29 PM	✓
4597	9	TUEA4F1D0B	PCB mount USB sockets	☐	6/24/2009 9:38 AM	✓
859	10	PX0840/A/2M00	BULGIN IP68 sealed data transfer sonnecto...	☐	6/2/2009 2:33 PM	✓
921	11	ESB33100000Z	Mini USB sockets	☐	12/22/2008 9:09 AM	✓
4588	12	ESB22A1101	Micro USB connectors	☐	6/23/2009 2:55 PM	✓
4592	13	KEYS935	Mini USB non-insulated plug	☐	6/23/2009 3:49 PM	✓
922	14	IEEE1394-W	IEEE 1394 ("Fire Wire") plugs	☐	12/22/2008 9:25 AM	☐
923	15	IEEE1394-G	IEEE 1394 ("Fire Wire") sockets	☐	12/22/2008 9:35 AM	☐

Figure 1: Main structure window in PARCAT

In connection with the need to store very large amounts of information, the idea of creating descriptions of individual products was taken from the previous manual composition. It is based on a catalogue divided into parts (e.g., semiconductors, tools, electrical and installation equipment, etc.), which are divided into chapters, and chapters are divided into sections. Frames, assigned to sections, are the smallest (and indivisible) part of the catalogue. A list of products featuring such information as manufacturer, summary description and prices is imported from the sales system. There is no need to enter symbols manually, which could pose a serious problem for such large quantities of products (tens of thousands and more).

2.2 Frames

The concept of a *frame* is the foundation of the system. It systematises the management of large amounts of products, facilitates the editing of data, making of amendments and also simplifies the data management for a large number of users. A frame is assigned to a section and can describe any number of products (in particular, one product). It is good if the products are a homogeneous group which



Frame 921: ESB33100000Z – Catalogue\CONNECTORS\Data connectors\USB and IEEE1394 connectors

Frame title	Mini USB sockets
Frame picture:	Object 1710 prepared correctly
Frame display type	Standard
Description	
Price headers:	Auto Price per piece in #CURRENCY ‖ Multiplicity: #MULTIPLICITY pcs. rollback
General texts	
Index words	ECE-mini and micro USB sockets; sockets mini USB; mini USB; PCB-USB sockets; SMD-USB sockets

PL Polish
Print All languages
2008-12-22 / 2010-01-11 14:01

ARTICLES

Symbol	Separate lines					Table normal 1		Table normal 2
Symbol	Manufacturer	Connector	Plug type	Connector variant	Fitting	Connector type	Contacts number	Dimensions
	Catalog and Web	Catalog and Web	Catalog and Web	Catalog and Web	Catalog and ...	Catalog and Web	Catalog and Web	Web
Symbol	**Manufacturer**	**Connector**	**Type**	**Connector vari...**	**Fitting**	**Connector type**	**Contacts number**	**Dimensions**
ESB33100000Z	ECE	socket	female	angled	SMD	mini USB A	4	Img 1711 ok
ESB34100000Z	ECE	socket	female	angled	SMD	mini USB B	4	Img 1712 ok
ESB34101000Z	ECE	socket	female	angled	SMD	mini USB B	5	Img 1713 ok
ESB35101000Z	ECE	socket	female	angled	SMD	mini USB C	5	Img 1714 ok

Figure 2: Frame editor in PARCAT

can be described by means of common characteristics. Thus, for example, it is possible to assign to one frame a whole series of resistors with a power of 0.25W in the SMD0608 housing (housing, power, manufacturer, maximum voltage are common features for all these resistors, and resistance — different for each resistor — is the distinguishing feature), or three universal hammers made by one manufacturer which vary in weight and the length of the shaft, but other characteristics (e.g., material, properties, applications) are the same. In addition, a frame features several permanent elements, such as frame title, overall picture, text description.

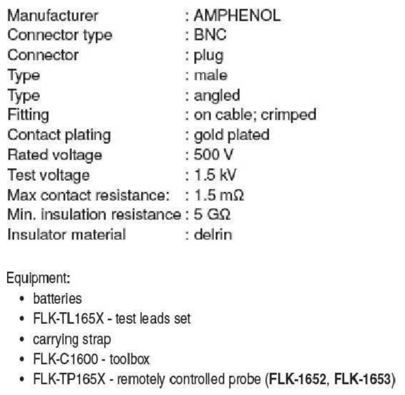
2.3 Parameters

The description of products is mainly based on user-defined parameters. This solution primarily ensures the consistency of generated descriptions and significantly reduces the costs of translation since the fields associated with a given parameter are translated only once. There are various types of parameters: text, single or multiple choice, several numeric options, as well as those allowing assignment of graphical objects.

2.4 Construction of tables

Parameters in a frame are automatically divided into two groups. The first group contains those parameters whose values are the same for all products in the frame, called common parameters. The parameters which have different values for individual products belong to the second group, called distinguishing parameters.

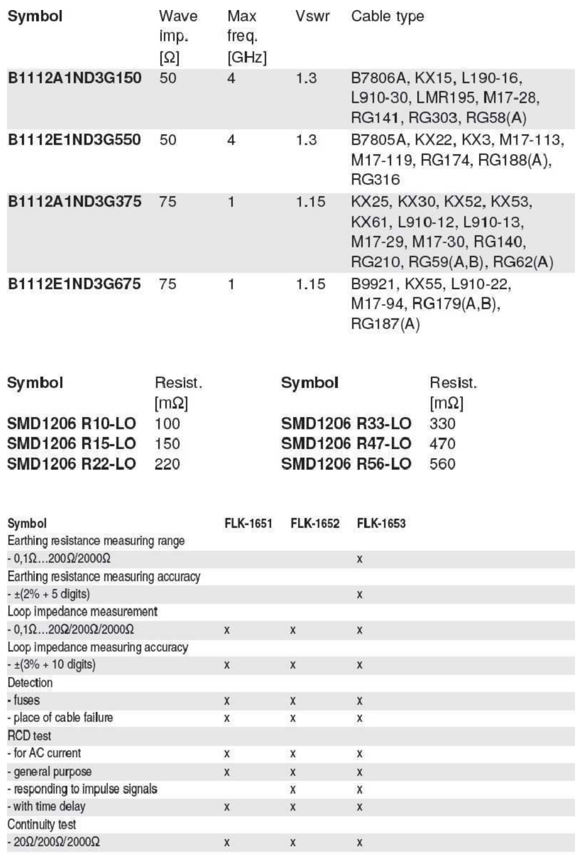
Clearly the method of data presentation for both groups should be different. The common parameters (Fig. 3) can be associated with their values and



Manufacturer	: AMPHENOL
Connector type	: BNC
Connector	: plug
Type	: male
Type	: angled
Fitting	: on cable; crimped
Contact plating	: gold plated
Rated voltage	: 500 V
Test voltage	: 1.5 kV
Max contact resistance:	: 1.5 mΩ
Min. insulation resistance	: 5 GΩ
Insulator material	: delrin

Equipment:
- batteries
- FLK-TL165X - test leads set
- carrying strap
- FLK-C1600 - toolbox
- FLK-TP165X - remotely controlled probe (**FLK-1652**, **FLK-1653**)

Figure 3: Different presentations of common product parameters



Symbol	Wave imp. [Ω]	Max freq. [GHz]	Vswr	Cable type
B1112A1ND3G150	50	4	1.3	B7806A, KX15, L190-16, L910-30, LMR195, M17-28, RG141, RG303, RG58(A)
B1112E1ND3G550	50	4	1.3	B7805A, KX22, KX3, M17-113, M17-119, RG174, RG188(A), RG316
B1112A1ND3G375	75	1	1.15	KX25, KX30, KX52, KX53, KX61, L910-12, L910-13, M17-29, M17-30, RG140, RG210, RG59(A,B), RG62(A)
B1112E1ND3G675	75	1	1.15	B9921, KX55, L910-22, M17-94, RG179(A,B), RG187(A)

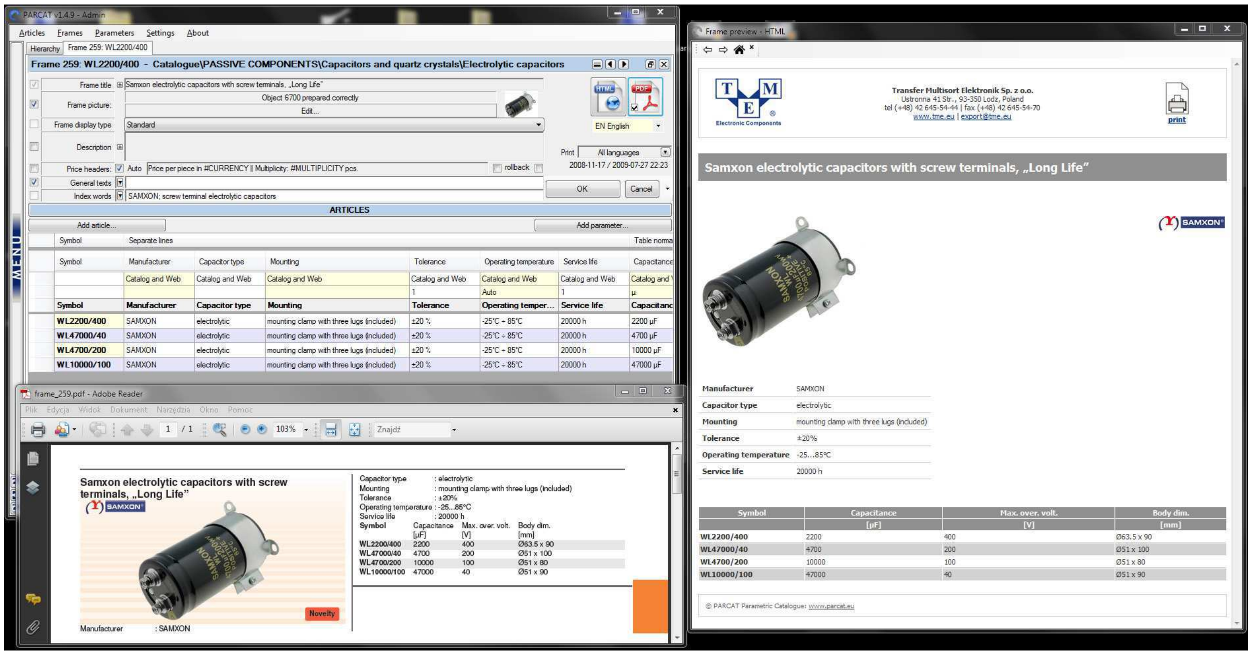
Symbol	Resist. [mΩ]	**Symbol**	Resist. [mΩ]
SMD1206 R10-LO	100	**SMD1206 R33-LO**	330
SMD1206 R15-LO	150	**SMD1206 R47-LO**	470
SMD1206 R22-LO	220	**SMD1206 R56-LO**	560

Symbol	**FLK-1651**	**FLK-1652**	**FLK-1653**
Earthing resistance measuring range			
- 0,1Ω…200Ω/2000Ω			x
Earthing resistance measuring accuracy			
- ±(2% + 5 digits)			x
Loop impedance measurement			
- 0,1Ω…20Ω/200Ω/2000Ω	x	x	x
Loop impedance measuring accuracy			
- ±(3% + 10 digits)	x	x	x
Detection			
- fuses	x	x	x
- place of cable failure	x	x	x
RCD test			
- for AC current	x	x	x
- general purpose	x	x	x
- responding to impulse signals		x	x
- with time delay	x	x	x
Continuity test			
- 20Ω/200Ω/2000Ω	x	x	x

Figure 4: Different presentations of distinguishing product parametrs

listed one under another. They can also be presented as a list with bullets. In general, since the common parameters relate to all products in the frame by definition, a list of respective products does not need to be specified.

The distinguishing parameters (Fig. 4) must be correlated with a list of products, thus showing precisely the products to which they apply.



Frame 259: WL2200/400 - Catalogue\PASSIVE COMPONENTS\Capacitors and quartz crystals\Electrolytic capacitors

Frame title	Samxon electrolytic capacitors with screw terminals, „Long Life"
Frame picture:	Object 6700 prepared correctly
Frame display type	Standard
Description	
Price headers:	Auto: Price per piece in #CURRENCY ‖ Multiplicity: #MULTIPLICITY pcs. rollback
General texts	
Index words	SAMXON; screw terminal electrolytic capacitors

2008-11-17 / 2009-07-27 22:23

Symbol	Separate lines						Table norma
Symbol	Manufacturer	Capacitor type	Mounting	Tolerance	Operating temperature	Service life	Capacitance
	Catalog and Web	Catalog and Web	Catalog and Web	Catalog and Web	Catalog and Web	Catalog and Web	Catalog and
				1	Auto	1	µ
Symbol	**Manufacturer**	**Capacitor type**	**Mounting**	**Tolerance**	**Operating temper...**	**Service life**	**Capacitanc**
WL2200/400	SAMXON	electrolytic	mounting clamp with three lugs (included)	±20 %	-25°C ÷ 85°C	20000 h	2200 µF
WL47000/40	SAMXON	electrolytic	mounting clamp with three lugs (included)	±20 %	-25°C ÷ 85°C	20000 h	4700 µF
WL4700/200	SAMXON	electrolytic	mounting clamp with three lugs (included)	±20 %	-25°C ÷ 85°C	20000 h	10000 µF
WL10000/100	SAMXON	electrolytic	mounting clamp with three lugs (included)	±20 %	-25°C ÷ 85°C	20000 h	47000 µF

Samxon electrolytic capacitors with screw terminals, „Long Life"

Capacitor type	: electrolytic
Mounting	: mounting clamp with three lugs (included)
Tolerance	: ±20%
Operating temperature	: -25...85°C
Service life	: 20000 h

Symbol	Capacitance [µF]	Max. over. volt. [V]	Body dim. [mm]
WL2200/400	2200	400	Ø63.5 x 90
WL47000/40	4700	200	Ø51 x 100
WL4700/200	10000	100	Ø51 x 80
WL10000/100	47000	40	Ø51 x 90

Manufacturer : SAMXON

Transfer Multisort Elektronik Sp. z o.o.
Ustronna 41 Str., 93-350 Lodz, Poland
tel (+48) 42 645-54-44 | fax (+48) 42 645-54-70
www.tme.eu | export@tme.eu

Samxon electrolytic capacitors with screw terminals, „Long Life"

Manufacturer	SAMXON
Capacitor type	electrolytic
Mounting	mounting clamp with three lugs (included)
Tolerance	±20%
Operating temperature	-25...85°C
Service life	20000 h

Symbol	Capacitance	Max. over. volt.	Body dim.
	[µF]	[V]	[mm]
WL2200/400	2200	400	Ø63.5 x 90
WL47000/40	4700	200	Ø51 x 100
WL4700/200	10000	100	Ø51 x 80
WL10000/100	47000	40	Ø51 x 90

© PARCAT Parametric Catalogue: www.parcat.eu

Figure 5: HTML and PDF previews in PARCAT

By default, PARCAT shows a simple text form for the common parameters and a table for the distinguishing parameters. However, the user has a number of tools to change the look of these tables, their order, the order of parameters inside the table, the font size in the table, the position of pictures around the table and many other aspects of the created description. By modifying the construction of tables, the user can adjust the appearance of the frames to their needs and present the data in the most readable way.

2.5 Translation

The list of languages used for product descriptions is not fixed and can be dynamically changed during work on a particular catalogue. All texts present in the system will be automatically marked for translation into a newly added language.

PARCAT features a complex panel for translation management. It indicates the exact number of phrases which require translation into a given language, and each phrase will be translated only once. It allows you to select those sections of a catalogue to be translated. Generated orders are sent to translators for whom a special application is prepared to facilitate their work. It is very important that the translators always see the full context of translation, even if they need to translate only one phrase. This allows for translation quality at the highest level. After the translator finishes work, their order is imported into the database, after verification steps to eliminate mistakes and improve the quality of translation further.

2.6 Graphics

All graphical elements in the program are stored as a pair of PDF files (ready for printing, CMYK with a cut out background) as well as JPG, GIF, etc. (for Internet use). The system features a mechanism allowing users to order the preparation or editing of graphic element. In this way, users without the ability to use graphics programs can still manage graphics. Orders along with the comments are sent to other users with permission to edit graphics (graphic designers). After the image processing is done, the user who sent the order is able to accept changes or request new ones.

2.7 Preview

The possibility of previewing a frame at any stage of its creation (in particular, with the graphics still not accepted!), even without saving changes, is a very important feature of the system (Fig. 5). What is more, the preview in a freely chosen language or languages is created immediately and shows the frame exactly as it will look in print (with accurate pagination) or when connected to a web site, along the lines of WYSIWYG.

A true and fast preview is vital in extensive publications of over 1000-page catalogues. It makes it possible for all users of the program to assist in the preparation of material, eliminating the complicated, time consuming and discouraging intermediate stage which is usually "the preparation of the preview" in traditional composition systems. Furthermore, the instantaneously generated preview allows users to quickly find both factual and graphical errors.

In addition to the frame preview it is possible to prepare a preview of a section, chapter or part of a catalogue. This allows you to check portions of a catalogue without the need to process all the material. It also enables easy monitoring of the number of pages in the forthcoming catalogue, which is important for ordering at the printing houses. The composition of a single chapter is no longer instantaneous, but it takes only a few minutes to compose tens of generated pages. Therefore these previews can also be generated as necessary.

All these features translate into a significant, if not revolutionary, way to shorten the time users must spend to prepare the catalogue material.

2.8 Typesetting of a catalogue

After preparing and checking all the material you can proceed to a catalogue composition. A generated PDF file features cropmarks and trimboxes, which makes it suitable for submission to the printing houses without additional modifications.

Multi-lingual typesetting with replacement of only the black colour plate is a strategic function of PARCAT. The system provides the ability to compose a catalogue where the data in all languages chosen in the process of composition can be found in one output file. This means that the texts in each of the languages are inserted into the same spaces maintaining the common position of illustrations and the same page breaks. Every language is placed on a separate layer, which allows you to have a preview or trial print for the given language. In addition, each language layer is composed by means of its own additional colour (the spot colour), which facilitates the work of the CTP studio before the preparation of printing plates. This method of printing, with replacement of the black plate, allows us to achieve enormous financial savings. The more language versions of a catalogue and the more catalogue pages, the more savings.

3 Under the hood: TEX in PARCAT

The typesetting in PARCAT is done using LATEX with the XƎTEX engine and a highly customised document class. Since we have to cope with text in several languages written in Latin and Cyrillic scripts, we eagerly switched to using Unicode. In this context XƎTEX with its UTF-8 input and its ability to use multilingual OpenType fonts provided a comfortable working environment.

TEX, being a batch processor, plays its role as a typesetting back-end for PARCAT very well. It is fast enough to provide almost instantaneous preview of selected frames. The user can also be practically certain that the result shown for a separate frame will be identical in a complete chapter. We think that in some aspects we have reached the edge of TEX's abilities, e.g., with respect to rearranging the language variants. Some of these manipulations would perhaps be easier in LuaTEX. We will probably investigate this possibility in the future.

Typesetting product catalogues is a rather atypical use of LATEX, so we had to solve quite a few TEXnical problems, the most important being the handling of language variants. This comprises combining several streams of text in one source file; overlaying language variants in such a way that page-breaking and picture positions are the same in all variants, and finally outputting all the variants of the text to the same PDF file using the "optional content groups" feature of the PDF format, with a separate spot colour for each language.

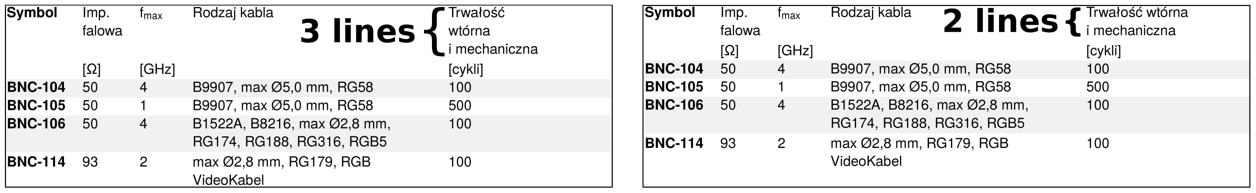
3.1 Tables

One major issue was handling the tables presenting parameters of the products being offered, an important part of the catalogues. Some of these tables are long, spanning up to about 20 pages. They contain headers repeating at the top of each column and should be typeset in a two-column arrangement. Unfortunately, standard LATEX packages (multicols and longtable, supertabular, etc.) do not handle such a combination. We developed a specialised solution in which page breaking is done by the multicols package. The package was slightly modified to carry the headers for the table in TEX marks (we used the ε-TEX ability to create new mark classes).

The column widths are automatically set to fit a table to the column width. A typical table in question consists of a body containing some parameter values, usually decimal numbers, and a header naming those parameters, sometimes with very long contents.

The original class hacks the `tabular` environment so that it measures its columns and performs a trial setting of a table. There are at most three trials:

- without line breaking,
- with line breaking, at every allowed blank (not at ~'s),


3 lines
2 lines

Symbol	Imp. falowa	f_{max}	Rodzaj kabla	Trwałość wtórna i mechaniczna
	[Ω]	[GHz]		[cykli]
BNC-104	50	4	B9907, max Ø5,0 mm, RG58	100
BNC-105	50	1	B9907, max Ø5,0 mm, RG58	500
BNC-106	50	4	B1522A, B8216, max Ø2,8 mm, RG174, RG188, RG316, RGB5	100
BNC-114	93	2	max Ø2,8 mm, RG179, RGB VideoKabel	100

Symbol	Imp. falowa	f_{max}	Rodzaj kabla	Trwałość wtórna i mechaniczna
	[Ω]	[GHz]		[cykli]
BNC-104	50	4	B9907, max Ø5,0 mm, RG58	100
BNC-105	50	1	B9907, max Ø5,0 mm, RG58	500
BNC-106	50	4	B1522A, B8216, max Ø2,8 mm, RG174, RG188, RG316, RGB5	100
BNC-114	93	2	max Ø2,8 mm, RG179, RGB VideoKabel	100

Figure 6: A table justified with the original algorithm (left) and the "more subtle" version (right)

- with line breaking, at any allowed point (including word hyphenations).

In some not-so-extreme cases the result appears as in Fig. 6, left. The table consists of narrow columns and large "column-glue" at the right filling the width to `\columnwidth`.

Let's underline that making this work *fully automatically* is quite an achievement, as anyone who knows something of TeX would agree. But it doesn't look too good compared with other tables, especially if you know nothing of TeX, does it?

So, the next step taken is to make such a table look as in Fig. 6, right, still fully automatically and without changes to the source of that table (so that no changes are necessary to the front-end software generating it).

The desired effect is achieved by repeating trial settings with an increasing `\looseness` in a sort of `\raggedright` scope (turning respective cells into `p{`⟨*dimen*⟩`}`), until a minimum value of `\looseness` is found (or a limit of iterations reached). Then widths of table columns are measured and applied to the final leading.

3.2 New layouts

As the project developed and the PARCAT system is offered to different clients, the need for new page layouts, or rather, graphical concepts, is natural. Samples of the layouts designed so far are shown in Fig. 7.

They are intended not only to present different shapes of graphical elements or placement of headings, but also to illustrate the fact that all those samples are typeset from the same product data and, moreover, the same "intermediate" TeX code.

To be more specific, PARCAT's front-end software (non-TeX) produces TeX code such as

```
{\sizevii
 \begin{wykaz}{@{}lll}
  \wyknaglowek{\textbf{\war{PL}{%
Symbol}\war{EN}{Symbol}\war{CZ}{Symbol}%
\war{DE}{Symbol}\war{HU}{Jelölés}} &
\najweziej{\war{PL}{Klasa wykonania}%
\war{EN}{Manufacture class}\war{CZ}{Třída
provedení}\war{DE}{Ausführungsklasse}%
\war{HU}{Kiviteli osztály}} & \najweziej{%
\war{PL}{Pokrycie styku}\war{EN}{Contact
plating}\war{CZ}{Povrch kontaktu}\war{%
DE}{Kontaktbeschichtung}\war{HU}{Érintkező
bevonata}} \\ }
   ...
 \end{wykaz}
} % wielkosc czcionki
```

in a multitude of files named `frame_`⟨*id*⟩`.tex`. Each file corresponds to a catalogue frame. PARCAT also produces code like this:

```
\begin{multicols}{2}
...
\KeysForNextFrame {2x3=0:1/0}
\NamedInput{frame_24_1/frame_24_1.tex}

\KeysForNextFrame {2x3=1:2-1}
\NamedInput{frame_3_1/frame_3_1.tex}
...
\end{multicols}
```

in so-called *intermediate* files. `\NamedInput` is an input wrapped with stacking the file name so that it can be referred to in messages, which we'll discuss later (section 3.4). So, as you see, the intermediate file inputs the frame files.

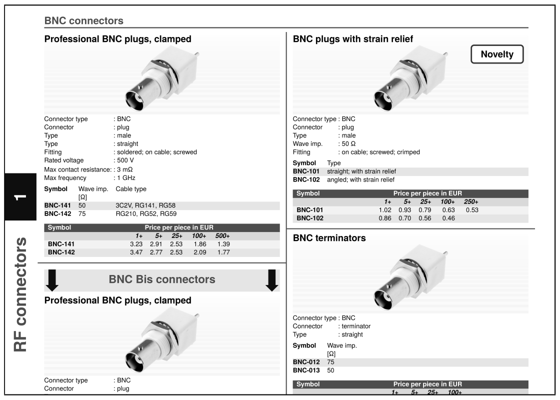
This file in turn is input by the main LaTeX document file, alongside files containing settings and configuration data generated by the front-end of the system.

You get all the different outputs (and more) depending on which main file you use — *on the same intermediate and frame files*!

Our intention is to keep all the templates compatible with one another. For example, notice the unconditional invocation of the `multicols` environment in the intermediate file (the code sample above) while only one of the examples shown in Fig. 7 is actually two-column.

Turning the `multicols` environment off was relatively easy. (Relatively, since it's off only at the main level, where `\currentgrouplevel` = 0.)

A bit more difficult was to reach a reasonably simple solution for the layout introduced in the IL template (lower left corner of Fig. 7). As you see, the pictures are typeset on the right side of text (tables) and the table(s) break in pages. The complication



BNC connectors

RF connectors

1

Professional BNC plugs, clamped

Connector type	: BNC
Connector	: plug
Type	: male
Type	: straight
Fitting	: soldered; on cable; screwed
Rated voltage	: 500 V
Max contact resistance:	: 3 mΩ
Max frequency	: 1 GHz

Symbol	Wave imp. [Ω]	Cable type
BNC-141	50	3C2V, RG141, RG58
BNC-142	75	RG210, RG52, RG59

Symbol	Price per piece in EUR				
	1+	5+	25+	100+	500+
BNC-141	3.23	2.91	2.53	1.86	1.39
BNC-142	3.47	2.77	2.53	2.09	1.77

BNC Bis connectors

Professional BNC plugs, clamped

Connector type : BNC
Connector : plug

BNC plugs with strain relief

Novelty

Connector type	: BNC
Connector	: plug
Type	: male
Wave imp.	: 50 Ω
Fitting	: on cable; screwed; crimped

Symbol	Type
BNC-101	straight; with strain relief
BNC-102	angled; with strain relief

Symbol	Price per piece in EUR				
	1+	5+	25+	100+	250+
BNC-101	1.02	0.93	0.79	0.63	0.53
BNC-102	0.86	0.70	0.56	0.46	

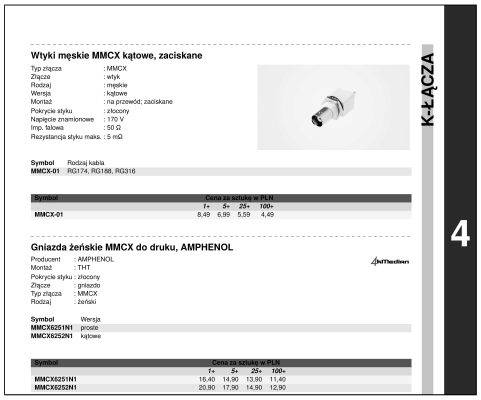
BNC terminators

Connector type	: BNC
Connector	: terminator
Type	: straight

Symbol	Wave imp. [Ω]
BNC-012	75
BNC-013	50

Symbol	Price per piece in EUR			
	1+	5+	25+	100+

The “classic” STE template



K-ŁĄCZA

4

Wtyki męskie MMCX kątowe, zaciskane

Typ złącza	: MMCX
Złącze	: wtyk
Rodzaj	: męskie
Wersja	: kątowe
Montaż	: na przewód; zaciskane
Pokrycie styku	: złocony
Napięcie znamionowe	: 170 V
Imp. falowa	: 50 Ω
Rezystancja styku maks.	: 5 mΩ

Symbol	Rodzaj kabla
MMCX-01	RG174, RG188, RG316

Symbol	Cena za sztukę w PLN			
	1+	5+	25+	100+
MMCX-01	8,49	6,99	5,59	4,49

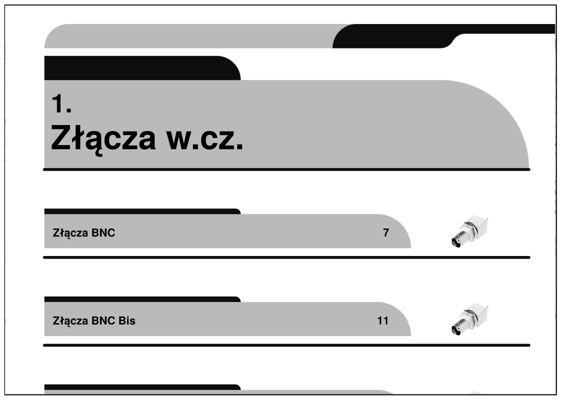
Gniazda żeńskie MMCX do druku, AMPHENOL

Producent	: AMPHENOL
Montaż	: THT
Pokrycie styku	: złocony
Złącze	: gniazdo
Typ złącza	: MMCX
Rodzaj	: żeński

Symbol	Wersja
MMCX6251N1	proste
MMCX6252N1	kątowe

Symbol	Cena za sztukę w PLN			
	1+	5+	25+	100+
MMCX6251N1	16,40	14,90	13,90	11,40
MMCX6252N1	20,90	17,90	14,90	12,90

The WZ template

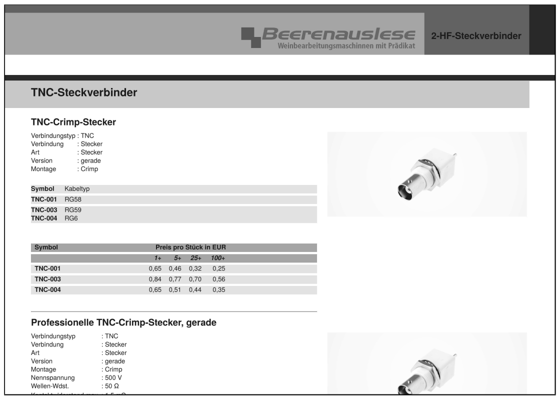


1.
Złącza w.cz.

Złącza BNC 7

Złącza BNC Bis 11

The IL template, mini-toc on left page



Beerenauslese
Weinbearbeitungsmaschinnen mit Prädikat

2-HF-Steckverbinder

TNC-Steckverbinder

TNC-Crimp-Stecker

Verbindungstyp	: TNC
Verbindung	: Stecker
Art	: Stecker
Version	: gerade
Montage	: Crimp

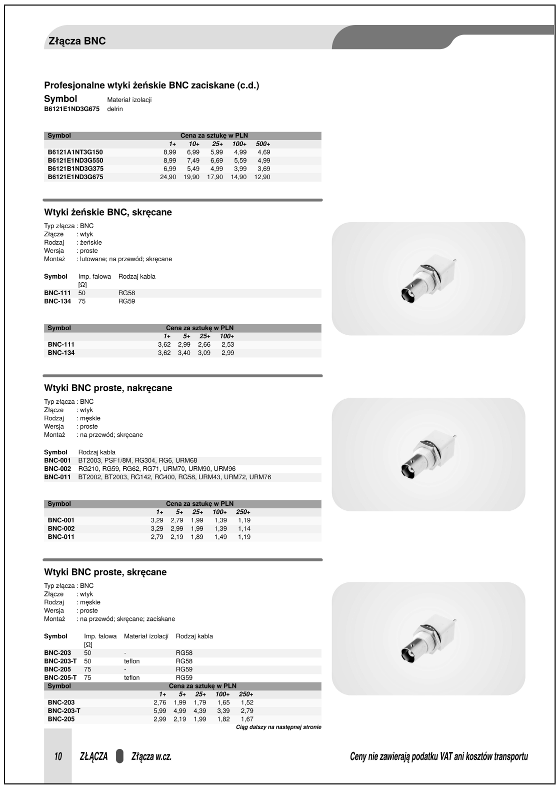
Symbol	Kabeltyp
TNC-001	RG58
TNC-003	RG59
TNC-004	RG6

Symbol	Preis pro Stück in EUR			
	1+	5+	25+	100+
TNC-001	0,65	0,46	0,32	0,25
TNC-003	0,84	0,77	0,70	0,56
TNC-004	0,65	0,51	0,44	0,35

Professionelle TNC-Crimp-Stecker, gerade

Verbindungstyp	: TNC
Verbindung	: Stecker
Art	: Stecker
Version	: gerade
Montage	: Crimp
Nennspannung	: 500 V
Wellen-Wdst.	: 50 Ω

The MK template



Złącza BNC

Profesjonalne wtyki żeńskie BNC zaciskane (c.d.)

Symbol	Materiał izolacji
B6121E1ND3G675	delrin

Symbol	Cena za sztukę w PLN				
	1+	10+	25+	100+	500+
B6121A1NT3G150	8,99	6,99	5,99	4,99	4,69
B6121E1ND3G550	8,99	7,49	6,69	5,59	4,99
B6121B1ND3G375	6,99	5,49	4,99	3,99	3,69
B6121E1ND3G675	24,90	19,90	17,90	14,90	12,90

Wtyki żeńskie BNC, skręcane

Typ złącza	: BNC
Złącze	: wtyk
Rodzaj	: żeńskie
Wersja	: proste
Montaż	: lutowane; na przewód; skręcane

Symbol	Imp. falowa [Ω]	Rodzaj kabla
BNC-111	50	RG58
BNC-134	75	RG59

Symbol	Cena za sztukę w PLN			
	1+	5+	25+	100+
BNC-111	3,62	2,99	2,66	2,53
BNC-134	3,62	3,40	3,09	2,99

Wtyki BNC proste, nakręcane

Typ złącza	: BNC
Złącze	: wtyk
Rodzaj	: męskie
Wersja	: proste
Montaż	: na przewód; skręcane

Symbol	Rodzaj kabla
BNC-001	BT2003, PSF1/8M, RG304, RG6, URM68
BNC-002	RG210, RG59, RG62, RG71, URM70, URM90, URM96
BNC-011	BT2002, BT2003, RG142, RG400, RG58, URM43, URM72, URM76

Symbol	Cena za sztukę w PLN				
	1+	5+	25+	100+	250+
BNC-001	3,29	2,79	1,99	1,39	1,19
BNC-002	3,29	2,99	1,99	1,39	1,14
BNC-011	2,79	2,19	1,89	1,49	1,19

Wtyki BNC proste, skręcane

Typ złącza	: BNC
Złącze	: wtyk
Rodzaj	: męskie
Wersja	: proste
Montaż	: na przewód; skręcane; zaciskane

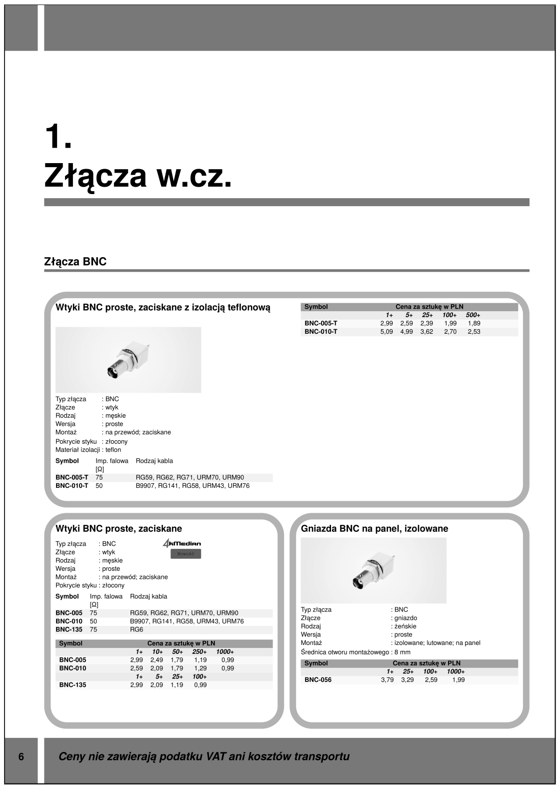
Symbol	Imp. falowa [Ω]	Materiał izolacji	Rodzaj kabla
BNC-203	50	-	RG58
BNC-203-T	50	teflon	RG58
BNC-205	75	-	RG59
BNC-205-T	75	teflon	RG59

Symbol	Cena za sztukę w PLN				
	1+	5+	25+	100+	250+
BNC-203	2,76	1,99	1,79	1,65	1,52
BNC-203-T	5,99	4,99	4,39	3,39	2,79
BNC-205	2,99	2,19	1,99	1,82	1,67

Ciąg dalszy na następnej stronie

10 ZŁĄCZA Złącza w.cz. *Ceny nie zawierają podatku VAT ani kosztów transportu*

The IL template, left page



1.
Złącza w.cz.

Złącza BNC

Wtyki BNC proste, zaciskane z izolacją teflonową

Typ złącza	: BNC
Złącze	: wtyk
Rodzaj	: męskie
Wersja	: proste
Montaż	: na przewód; zaciskane
Pokrycie styku	: złocony
Materiał izolacji	: teflon

Symbol	Imp. falowa [Ω]	Rodzaj kabla
BNC-005-T	75	RG59, RG62, RG71, URM70, URM90
BNC-010-T	50	B9907, RG141, RG58, URM43, URM76

Symbol	Cena za sztukę w PLN				
	1+	5+	25+	100+	500+
BNC-005-T	2,99	2,59	2,39	1,99	1,89
BNC-010-T	5,09	4,99	3,62	2,70	2,53

Wtyki BNC proste, zaciskane

Nowość

Typ złącza	: BNC
Złącze	: wtyk
Rodzaj	: męskie
Wersja	: proste
Montaż	: na przewód; zaciskane
Pokrycie styku	: złocony

Symbol	Imp. falowa [Ω]	Rodzaj kabla
BNC-005	75	RG59, RG62, RG71, URM70, URM90
BNC-010	50	B9907, RG141, RG58, URM43, URM76
BNC-135	75	RG6

Symbol	Cena za sztukę w PLN				
	1+	10+	50+	250+	1000+
BNC-005	2,99	2,49	1,79	1,19	0,99
BNC-010	2,59	2,09	1,79	1,29	0,99
	1+	5+	25+	100+	
BNC-135	2,99	2,09	1,19	0,99	

Gniazda BNC na panel, izolowane

Typ złącza	: BNC
Złącze	: gniazdo
Rodzaj	: żeńskie
Wersja	: proste
Montaż	: izolowane; lutowane; na panel
Średnica otworu montażowego	: 8 mm

Symbol	Cena za sztukę w PLN			
	1+	25+	100+	1000+
BNC-056	3,79	3,29	2,59	1,99

6 *Ceny nie zawierają podatku VAT ani kosztów transportu*

The Mod(ular) template

Figure 7: Samples of the PARCAT templates’ output

is that the picture is *not* boxed with adjacent text (which would kill the flexibility of `\vskip`s) but put in a “smashed” box preceding the text, and proper page breaking (i.e., forbidding breaks until subsequent text at least reaches the height of the picture) is ensured by a local change to `\pagegoal`.

The template presented in the lower right corner of Fig. 7 (Mod), is quite distinct from all the others. It positions a logical frame in a geometric frame consisting of some number of modules *sensu graphico*, i.e., rectangles of a grid. The grid on the illustration is 2×3.

This template is under construction. Our goal is to make it typeset a frame in a proper shape according to free space left on page. So far it typesets frames in a “greedy multicolumn” shape, either “vertical-first” or “horizontal-first” by default or in a shape specified by a key. For instance, the middle frame of the example has (in the optional argument of the `ramka` environment) a key `2x3=1:1`, which means that on the 2×3 grid it should be put in two horizontally adjacent rectangles.

The keys are handled by the pgfkeys package by Till Tantau, which I find much easier to learn than xkeyval, thanks to the path-like structure of the keys.

Although in this article we present illustrations in grayscale, real templates allow colours, of course. Moreover, all the colours are adjustable by the end user. But that belongs to the next story:

3.3 Changing the layout parameters

3.3.1 ParcatColours

The PARCAT templates provide a straightforward mechanism for setting colours. The end user doesn’t have to define the colour of each graphical element separately but is given a set of colour variables that are initialised hierarchically.

This mechanism is realised with the commands `\NewParcatColor` and `\SetParcatColor`, whose names are self-explanatory. These commands use the mechanisms of colour definition of the xcolor package, so a very wide range of assignments, reassignments, mixing and shading is available.

In most of the templates the basic colour is `nadroz`, a legacy name derived from “nadrozdział”, ‘super-chapter’ in Polish. Other colours are tints of it by default. There are also colours (colour names) for the distinctive frame of inserted advertisements, for the backgrounds of pictures and so on.

A nearly parallel mechanism handles other parameters such as dimensions of graphical elements, values of Boolean switches, etc.

3.3.2 ParcatParameters

The main commands are:

- `\NewParcatParameter`,
- `\SetParcatParameter`,
- `\RenewParcatParameter` and
- `\OldParcatParameter`.

These commands not only declare or set PARCAT parameters but also put them on checklists to issue an error message if a parameter isn’t set at the point of `\AtBeginDocument` (which we discuss in section 3.4).

The first three names are self-explanatory. The last serves to include standard (LA)TEX parameters in PARCAT’s checklists (with the information about their types). E.g., `parcat.cls` has a declaration specifying `\parindent` as a value of type `dimen`:

```
\OldParcatParameter\parindent
  \dimen[0pt]
```

This allows the standard `\parindent` to be specified with `\SetParcatParameter` in a `settings_...tex` file. It doesn’t *have* to be set because the optional argument sets it to `0pt` by default.

Among the available parameter types are `dimen`, `count`, `skip`, and the corresponding `\dimexpr`, `\nu¦mexpr`, `\glueexpr` to denote a proper *text* for the respective ε-TEX primitives, and `\edim`.

This last item stands for ‘evaluated `\dimexpr`’, where evaluation (`\the`-expansion) of the expression is performed `\AtBeginDocument`.

As with `ParcatColors`, `\edim`s are organised in a sort of inheritance hierarchy.

3.4 PARCAT messages

As mentioned earlier, PARCAT’s TEX back-end generates information, warning and error messages in an XML format, which is intended for parsing by the front-end PARCAT software to provide information to the end user, who most probably is not acquainted with TEX.

In particular, such messages are issued if some parameters are not set while they should be, or if an attempt is made to assign a parameter value not appropriate for it. As an example, the `parcat.cls` class declares

```
\NewParcatParameter\TitleOnBg
  \boolean[true]
```

which means the only allowed values for the parameter are (case-insensitive) `true` and `false`. But suppose the `settings_...tex` file contains a typo:

```
\SetParcatParameter\TitleOnBg{fals}
```

Then TEX outputs an error message:

```
!
<ParcatError>
 <ParcatErrorFile>settings_Modular.tex
  </ParcatErrorFile>
 <ParcatErrorLine>108
  </ParcatErrorLine>
 <ParcatErrorForDummies>
  The only values suitable for ***\TitleOnBg***
  are ***true*** and ***false***
  while you gave ***fals***.
 </ParcatErrorForDummies>

 <ParcatErrorForTeXies>
 </ParcatErrorForTeXies>
</ParcatError>

.
Type  H <return>  for immediate help.
...
l.108 \SetParcatParameter\TitleOnBg{fals}
```

The XML is subsequently parsed by the front-end programme to be presented to the user in a nice form.

3.5 Modularisation of the code

Another goal of redesigning the code was to make it suitable for new clients, who would want not all the layouts in any possible configuration but only a few or even just one. It was immediately evident that dividing the code responsible for distinct parts of the templates into separate files (*modules*) would be a good idea.

For that purpose we created a mechanism based on these three commands:

- `\DeclareParcatModules`,
- `\ProvideParcatModule` and
- `\LoadParcatModule`.

The last two names are self-explanatory, given that a module (*sensu programmatico*) is much like a LaTeX macro package or a document class options file. But why is the first in plural? It is because from the very beginning of the operation (Operation *Divide et Impera* ;-)) it was clear to us that there would be more than one module (variant) for each logical part of a template.

And indeed, so far there are about 10 modules of page layout, 6 modules of headings, 3 modules of index, 5 modules of frames, 3 modules of tables, &c.

We intend to keep the modules compatible with one another, e.g., any of the frame modules can work with any of the table modules (as far as logically possible). The tests performed so far seem to confirm that such a condition is preserved.

Not the least in preserving that compatibility is the manner of writing the code: all in one source file (using the gmdoc package, with an abundance of commentary) and generating the working files with docstrip directives.

But what about that "industry" in the title? — one may ask.

Our clients are companies with large product bases. To mention just one of them, its printed catalogue for the year 2011 is over 1700 A4 pages in thousands of copies in each of eight languages. That *is* industry and being able to handle it with (XƎ)TeX makes us proud.

Another and probably not the least aspect of the "industrial strength" of PARCAT's TeX back-end is its flexibility for the different needs of different clients, achieved with modularisation and variation of the parameters.

The PARCAT project is developing dynamically and dedicated to treating every client individually so hopefully there'll be many new features to present in the future.

More information about the PARCAT project can be found at `www.parcat.eu`.

◇ Wiktor Dziubiński
`w.dziubinski at parcat (dot) eu`

◇ Marcin Woliński
`wolinski at gust (dot) org (dot) pl`

◇ Grzegorz Murzynowski
`g.murzynowski at parcat (dot) eu`

TUG Libre Font Fund, Google Web Fonts, and Kickstarter

Dave Crossland

Google Web Fonts provides a catalog containing hundreds of libre-licensed fonts (`http://www.google.com/webfonts`). The team recently began supporting a funding experiment for libre-licensed fonts, in which TUG plays a key part, using Kickstarter.

Kickstarter.com is a popular service for funding all kinds of creative projects. It works by setting out a vision for a project and a target amount of money to raise within a set time period. As the clock counts down, people who want to see the project succeed can pledge donations. Their money is only spent if enough other people also contribute and the target amount is reached by the end of the campaign period. Successfully funded projects often raise more than their target.

Each month there are many typeface designs proposed to the Google Web Fonts team for publication, and financial support. The team has a budget and can't support everything; even with the best quality proposals, it can be hard to decide about those that are quite similar to ones already published. Perhaps the best judge of which web fonts the public wants to use is the public. So I invited the designers of three recent proposals to try out Kickstarter, so we could see how it might work for font projects.

First is Marcello Magalhaes's Folk, which transforms the vernacular lettering of Sao Paulo into a font. Already popular as a web font, it has been used by The Independent Film Channel and Mozilla — but it only included an uppercase set of glyphs, and not all the symbols and accents that Google Web Fonts requires. For this project, Marcello is completing the font to the Basic Latin character set used by Google Web Fonts, and has designed a poster to go with the new release that was offered as a reward.

Next is Fast Brush Script. This is the working name for a font by Pablo Impallari. Pablo's first font, Lobster, is one of the most popular Google Web Fonts, having been served over 2 billion times. Pablo is offering a very unusual reward — choosing the name. Normally the name of a font is sacred to the designer, but Pablo is opening up the opportunity for corporate patronage of his work. The development name of 'Fast Brush Script' reflects the core concept of the typeface. This font was in a very early development stage when it was announced on Kickstarter, with only the lowercase letters fully prototyped. Pablo offered the current development version on his Kickstarter project page.

Finally there is Montserrat, an extremely high quality sans serif text typeface by Julieta Ulanovsky. Advanced substantially during her studies at the prestigious University of Buenos Aires' Masters degree in Typeface Design, the design revives the historical type of the Montserrat neighbourhood where Julieta lives and works. This genre of type has been a popular trend in recent years and this typeface in particular stands out with its excellent quality. Setting it apart are the set of alternative caps, which add a little fun to a very functional text typeface.

Folk and Fast Brush Script just made their funding targets with a lot of help from the Google Web Fonts team. But in the final few days of the funding drive, Montserrat was chosen for highlighting in Kickstarter's weekly email newsletter to its users, and placed on the Kickstarter home page. It went on to raise nearly double the target amount!

FOLK
TIPOGRAFIA
POPULAR
EMCORES
!!! OPEN TYPE TRUE TYPE
(#) MAC & PC
COMBINE SOLID & SOLID
F
Folk

abcdefghijklm
nopqrstuvwxyz
Fast Brush

WHO
MAREO
Julieta
GILLES
ZOCOTROCO
QUIA
Montserrat

Kickstarter uses Amazon Payments to handle the financial transactions involved in its projects. This service will only pay out to bank accounts held in the USA. Since all the designers (so far) live in South America, the TUG Libre Font Fund enabled them to make use of the service. In return, I hope you will be able to make use of these fonts.

This approach to funding is ongoing; please feel free to peruse and/or contribute to the past and current projects at `http://fundwebfonts.appspot.com`. At this writing, highlighted projects seeking funding include Euphoria Script (self-explanatory), Exo Sans (a contemporary geometric design), and Open Educational Resources for Typography (a free typography/font learning resource, à la Wikipedia).

(Disclaimer: I am currently working as consultant for Google on their Web Fonts project. This article is entirely my personal opinion and does not represent the views of Google, Inc., in any way.)

◇ Dave Crossland
`dave (at) lab6 dot com`
`http://tug.org/fonts/librefontfund.html`

Book Reviews

Book review: *Bodoni, Manual of Typography — Manuale tipografico (1818)*

Boris Veytsman

Giambattista Bodoni, *Manual of Typography — Manuale tipografico (1818).* Stephan Füssel, editor. Taschen, 2010. 1208 pp. Hardcover, US$69.99. ISBN 978-3-8365-0553-6

One of the aims of Unicode is to make possible and relatively straightforward to create, for example, an English text with quotes in German, Arabic, Hebrew, and many other languages. A typesetter understands, however, that it is not enough to have a uniform digital representation of the "letters": one also should have a font (or collection of fonts) with letterforms for all these scripts. Therefore an important task for font designers is to create fonts with large collections of glyphs for different scripts. Only when this is done can we say that we can make multi-language books with truly harmonious and beautiful pages.

One may think that this task became relevant only in recent times. Giambattista Bodoni (1740–1813) is a great reminder that this impression is just wrong.

Typophiles may remember Bodoni for his great Latin fonts, which still are widely used today, or for his influential editions of classics. However, another aspect of Bodoni's heritage becomes increasingly more important in today's interconnected world: his extensive work on non-Latin scripts. These scripts interested Bodoni throughout his career. Starting at the age of eighteen Bodoni worked with the exotica division of the Tipografia Poliglotta Vaticana where his work spanned Arabic, Tibetan and many other scripts. Later, in Parma, he published a collection of bridal poems in 25 Oriental languages with Latin translations (1775). As a mature typographer, Bodoni published *Oratio Dominica,* the *Lord's prayer* in 155 languages (1806). He personally cut 55,000 matrices for his multi-language books. Bodoni strived to combine calligraphic traditions of these scripts with the beauty and clarity of his superb typography.

TeX users may want to take a look at GFS Bodoni fonts (`http://www.ctan.org/pkg/gfsbodoni`) with full support for Greek, Latin and math. They are included in TeX Live and MiKTeX, as well as being available directly from CTAN.

The book *Manuale Tipografico* is a major work of the great typographer. He did not live to see it printed: his widow Margherita and his foreman Luigi Orsi completed the edition and published it five years after the master's death.

The first volume of the manual contains a huge collection of samples of Latin fonts: roman, italic and cursive, with variants, weights, sizes. Margherita wrote in her preface that Bodoni considered it necessary for good typography to carry a collection of main fonts large enough so the difference between the adjacent sizes is not easily seen by a trained eye — a feat almost unheard of before the advent of digital typography.

The second volume has an extensive collection of Greek and Cyrillic fonts, along with a huge number of other scripts: Hebrew, Tibetan, Arabic, Armenian, Coptic, Georgian and many, many others (curiously enough, black letter is also considered to be an "exotic script" and put in the second volume as "Tedesco", German — between "Malabarico", Malayalam, and "Russo", Russian). Bodoni himself in his preface evidently takes a considerable pleasure in listing the fonts, carefully observing the difference between Square Hebrew and Rabbinical Hebrew, and noting that in Arabic writing *. . . the intricate* Sulsi *letter is employed in frontispieces and beginning, the hanging* Tajik *is very fashionable in Persia, while Turks love the reverse* Divân. The preface also contains important musings by Bodoni about the fundamental of typographic art.

The book also includes many pages of astronomical, mathematical and medical symbols, decorative rules, ornaments and even musical sheets in several styles.

Besides being a huge source of invaluable information, the book is a pleasure to read; just turning its pages may make a great evening for a typophile.

Since its publication in 1818 the *Manuale* became a very rare book. In 1965 it was reprinted in 900 numbered copies, which, of course, immediately became a prized rarity themselves. Thus a new mass market edition by Taschen is a very welcome project. A well preserved copy from Staatsbibliothek zu Berlin is reproduced here in all its glory. For those readers who are not fluent in Italian the book contains a booklet in a pocket glued to the inside cover with the translation of the texts and a very interesting preface by Stephan Füssel, director of the Institute of the History of the Book at the Johannes Gutenberg University of Mainz and the editor of the publication (most of the facts about Bodoni's biography cited above can be found in this erudite and well-written foreword). The book is printed on a good thick paper and is well bound. Probably in order to keep the price lower this edition contains both volumes of the original *Manuale* in one large book (albeit with two tassels). This makes it rather heavy (about nine pounds) and somewhat difficult to handle. However, the binding allows one to open the book at any place without much problem.

This book is a must for a font specialist or a typographer. Its relatively low price and beauty make it a good addition to a library of a book lover.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
USA
`borisv (at) lk dot net`
`http://borisv.lk.net`

Book review: *LaTeX and Friends*

Boris Veytsman

Marc van Dongen, *LaTeX and Friends*, Springer, Feb. 29, 2012, x+330pp., 145 ill., 15 in color. Hardcover, $69.95 approx., ISBN 978-3-642-23815-4.

It is difficult to write a new and original introductory book about LaTeX today. The author must compete with a number of great books, including freely available ones like [1]. Some these books were written by the authors or primary developers of the language [2–5]. Still, people have been writing mathematics textbooks for several thousand years, and nevertheless new ones continue to be written.

The way one can judge introductory LaTeX textbooks is similar to how figure skating is judged. There are several "required elements" which must be present in any book, like the explanation of LaTeX macros and workflow. There are also several "free elements" like additional packages or tricks the author chooses to include. The book can be evaluated by the pedagogical skill with which the author performed the "required elements", introducing the fundamentals of LaTeX, and by his choice of the "free" ones.

The new book by Marc van Dongen deserves high scores in both categories.

The first two parts of the book, *Basics* and *Basic Typesetting* discuss the material one can expect to find in any LaTeX textbook: the organization of LaTeX documents, the language constructs, the alphabet, etc. They are explained lucidly and well. What distinguishes this book from many other texts is that the author stresses the fact that LaTeX is a programming language, and therefore a LaTeX file is actually a program that instructs the computer to create the final product: the typeset pages on paper or on screen. *This means,* writes the author*, that you can use software engineering techniques such as top-down design and stepwise refinement.*

This is a very important fact, which must be explained to the students of TeX and LaTeX, especially those who are accustomed to WYSIWYG documents. A WYSIWYG document presents itself as "the final product" (this is a deception of sorts because, as any experienced user can attest, "what you finally get" is emphatically not "what you see"), while a TeX document is best viewed as a program to produce "the final product". The author returns to this many times, discussing such software engineering concepts as maintainability applied to TeX documents. This approach would likely be very comfortable and familiar for engineers and software developers.

The author's use of Unix-like syntax for his examples adds to the impression that the target audience of the book is the people who are not afraid of compilers and can write some code. This is a very welcome development. Too often publishers prefer to print computer-related books intended for "dummies" or even "complete idiots". While geniuses arguably do not need introductory texts, there is a perceptible dearth of books for the reasonably intelligent person. *LaTeX and Friends* is definitely one of such books, and it makes for pleasant and useful reading.

Since *The TeXbook* [6], many books about TeX discuss not only the typesetting program, but also other aspects of typographic art and science, discussing the rules for book design and the best practices. *LaTeX and Friends* follows this tradition, and Bringhurst's immortal *Elements* [7] is one of the most often cited books in the text. The reader learns many useful typographic facts, such as setting the punctuation symbols at the border of two types in the brighter type, the spacing in abbreviations and initials, etc. Many people from the intended audience get their first exposure to the typography from TeX-related books, and this one provides a good introduction to the subject.

The author chooses PDF mode as the main way to produce the result — probably a sensible choice nowadays. He does discuss DVI mode as a quick way to get a preview of the typeset pages. He obviously prefers *biblatex* to the "traditional" BibTeX interface to the bibliography. Still, a description of the *natbib* package would be a useful addition to the discussion of the author–year bibliographies.

The third part of the book discusses *Tables, Diagrams and Data Plots.* It contains a detailed introduction to the *TikZ* suite — probably one of the best existing descriptions of this highly useful package. This description alone makes the book worth buying. The section on tables, however, is smaller and less detailed; the reader who needs more should probably turn to the recent book by Herbert Voß [8] dedicated to typesetting tables in LaTeX.

The fourth part of the books is called *Mathematics and Algorithms.* The author discusses the use of *amsmath* and the related packages from the American Mathematical Society. Again, a reader who needs a detailed description of these packages may want to consult the books which deal with them as a primary topic, such as the two volumes by George Grätzer [9, 10]. A chapter or two in a general textbook, of course, cannot be a comprehensive description of these large packages. One can always argue with the author's choice: for example, van Dongen discusses the `split` environment, but does not mention `multline`, while I find the latter more useful than the former. I also would argue that the only "discussion" of `eqnarray` environment should be the warning: "never use this ugly monster!" A stranger omission happens in the discussion of variable sized delimiters, where the author explains the usage of `\left` and `\right` keywords for automatic sizing (with a useful trick of `\vphantom` for the proper sizing of multi-line expressions), but does not mention the manual sizing with `\Biggl`, `\Bigl`, `\bigl` commands and their "right" complements. The description of the *listings* package is rather short and does not mention many of its useful features. On the other hand, the discussion of the *algorithm2e* package is very detailed and well written.

The fifth part of the book, *Automation*, deals with the definition of new macros. It is probably intended for a more advanced reader than the rest of the book. The discussion of branching, loops and switching there is rather interesting, as well as containing some introductory remarks on the TeX interface (as opposed to the LaTeX interface).

The last part, *Miscellany,* includes a couple of chapters on various topics that do not fit into the other parts. It has a short but well written introduction to *beamer* presentations, a chapter on writing classes and packages, and a chapter on using OpenType fonts. The chapter on writing classes and packages is probably not as good as the famous guide [11], and does not cover the use of `.dtx` file for self-documenting code. The chapter on OpenType fonts contains an interesting discussion of a more esoteric topic. I wonder whether some script to automate this, e.g., using *fontinst* [12, 13] for some parts of the process, would help.

The book is well typeset using the *FF Nexus* font family. Unlike many other books on TeX, it has a detailed colophon, adding to the pedagogical value of the book. It has a good index separated into categories, and a short dictionary of typographic jargon.

This is a very useful book which can be recommended as a textbook on LaTeX for an introductory course or for self-education. It has chapters interesting for beginners and for experienced TeXnicians, and will be a welcome addition to either bookshelf.

References

[1] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε*, April 2011. `http://mirrors.ctan.org/info/lshort`.

[2] Leslie Lamport. *LaTeX: A Document Preparation System,* second edition. Addison-Wesley Publishing Company, Reading, MA, 1994. Illustrations by Duane Bibby.

[3] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion.* Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley Professional, Boston, second edition, 2004.

[4] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents With TeX and PostScript.* Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, 1997.

[5] Michael Goossens, Sebastian Rahtz, Eitan M. Gurari, Ross Moore, and Robert S. Sutor. *The LaTeX Web Companion: Integrating TeX, HTML, and XML.* Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison Wesley Longman, Reading, MA, 1999.

[6] Donald Ervin Knuth. *The TeXbook.* Computers & Typesetting A. Addison-Wesley Publishing Company, Reading, MA, 1994. Illustrations by Duane Bibby.

[7] Robert Bringhurst. *The Elements of Typographic Style.* Hartley & Marks, Publishers, Vancouver, BC, Canada, 2004.

[8] Herbert Voß. *Typesetting Tables with LaTeX.* Cambridge UIT, 2011.

[9] George Grätzer. *Math into LaTeX.* Birkhäuser, Boston, third edition, 2000.

[10] George Grätzer. *More Math into LaTeX.* Springer, New York, fourth edition, 2007.

[11] LaTeX3 Project. *LaTeX 2ε For Class and Package Writers*, 2006. `http://mirrors.ctan.org/macros/latex/doc/clsguide.pdf`.

[12] Philipp Lehman. *The Font Installation Guide*, December 2004. `http://mirrors.ctan.org/info/Type1fonts/fontinstallationguide`.

[13] Alan Jeffrey, Rowland McDonnell, and Lars Hellström. *Fontinst: Font Installation Software for TeX*, December 2004. `http://mirrors.ctan.org/fonts/utilities/fontinst`.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030, USA
`borisv (at) lk dot net`
`http://borisv.lk.net`

The Treasure Chest

This is a list of selected new packages posted to CTAN (`http://ctan.org`) from July–October 2011, with descriptions based on the announcements and edited for brevity.

Entries are listed alphabetically within CTAN directories. A few entries which the editors subjectively believed to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions.

We hope this column and its companions will help to make CTAN a more accessible resource to the TEX community. Comments are welcome, as always.

◇ Karl Berry
`http://tug.org/ctan.html`

fonts

`bickham` in `fonts`
Virtual fonts for Adobe Bickham Script Pro as a math calligraphic font.

`calligra-type1` in `fonts`
Type 1 version of the Calligra font.

`dejavu` in `fonts`
DejaVu fonts, in TrueType and Type 1.

`opensans` in `fonts`
OpenSans fonts, in TrueType and Type 1.

`persian-modern` in `fonts`
Text fonts from the FarsiTEX project, in TrueType.

`pxtxalfa` in `fonts`
Virtual math fonts based on `pxfonts` and `txfonts`.

graphics

`braids` in `graphics/pgf/contrib`
Drawing braid diagrams with PGF/Ti*k*Z.

`mpcolornames` in `graphics/metapost/contrib/macros`
Color names from X11, SVG, Dvips, and `xcolor`.

`tqft` in `graphics/pgf/contrib`
Drawing topological quantum field theory (TQFT) diagrams with PGF/Ti*k*Z.

info

`biblatex/de` in `info/translations`
German translation of `biblatex` documentation.

`enumitem/de` in `info/translations`
German translation of `enumitem` documentation.

`europecv/de` in `info/translations`
German translation of `europecv` documentation.

`fifinddo-info` in `info`
German HTML beamer presentation using `nicetext`.

macros/latex/contrib

`bhcexam` in `macros/latex/contrib`
Exam class for high school math in China.

`business-research` in `macros/latex/contrib`
LATEX class for the journal *Business Research*.

`ghab` in `macros/latex/contrib`
Typeset argument in a box with a decorated frame.

`gitinfo` in `macros/latex/contrib`
Support Git version control metadata in LATEX.

`ifetex` in `macros/latex/contrib`
Test whether ε-TEX is available.

`keyval2e` in `macros/latex/contrib`
Lightwight and robust facilities for managing keys.

`meetingmins` in `macros/latex/contrib`
Format written minutes of meetings.

`musous` in `macros/latex/contrib`
Style for the Department of Music at the University of Osnabrück, Germany.

`mversion` in `macros/latex/contrib`
Tracking different versions of your LATEX document.

`pagecolor` in `macros/latex/contrib`
Macros for the current background (page) color.

`quoting` in `macros/latex/contrib`
Consolidated environment for displayed text.

`realboxes` in `macros/latex/contrib`
Read arguments to box commands as boxes rather than macro arguments.

`rviewport` in `macros/latex/contrib`
Viewport sizes as fractions of natural image sizes.

`sidenotes` in `macros/latex/contrib`
Allow typesetting of general text in the margins for, e.g., science textbooks.

`tablefootnote` in `macros/latex/contrib`
Support footnotes in tables.

`tagging` in `macros/latex/contrib`
Generate multiple documents from a single source.

`tram` in `macros/latex/contrib`
Support for “tram boxes”, using patterns of dots.

macros/latex/contrib/babel-contrib

`russian` in `m/l/c/babel-contrib`
Updated Russian support for Babel.

macros/latex/contrib/beamer-contrib

`beameraudience` in `m/l/c/beamer-contrib`
Assemble frames for different audiences.

support

`nlatexdb` in `support`
C#/.`NET` preprocessor to execute SQL queries and format the results in LATEX.

ArsTEXnica #11–12 (2011)

ArsTEXnica is the journal of GuIT, the Italian TEX user group (`http://www.guit.sssup.it/`).

ArsTEXnica #11 (April 2011)

Gianluca Pignalberi, Editoriale [From the editor]; pp. 3–4

Antonello Pilu, La creazione di una prova d'esame [How to create an exam test]; pp. 5–14

This article will show how to realise, in a fast and simple way, a test with LATEX 2$_\varepsilon$, using its exam package. With this package we will be able to elaborate many different tests, from the easiest to the more complex ones, always keeping the high graphical quality of LATEX 2$_\varepsilon$.

Gianluca Pignalberi, Salvatore Schirone, Jerónimo Leal, Introduzione agli strumenti per grecisti classici [Introduction to tools for hellenists]; pp. 15–30

Hellenists' work is as hard, or even harder, than that of any other literary scholars. Thus, they need powerful and versatile tools. This paper shows two editors, two typesetters and some useful techniques for this job. The authors of this paper effectively use all of them.

Marco Crivellaro, Un dialogo tra GNU R e LATEX: `xtable` e Sweave [Dialogue between GNU R and LATEX: `xtable` and Sweave]; pp. 31–36

The aim of this article is to show the ways by which R, software for statistical computation, can be combined with LATEX for the writing of high quality statistical reports. This article is from my experience during the writing of my thesis in marketing at University Ca' Foscari in Venice.

Giovanni Mascellani, Uno strumento per gestire progetti LATEX cooperativi [A tool to manage cooperative LATEX projects]; pp. 37–40

Using and modifying available free and open source tools, we made a portal that students can use to collaborate in writing notes and text with LATEX. It also features an automatic tool that compiles the sources and makes the PDF files easily reachable directly from the Internet site.

Jean-Michel Hufflen, Passare da LATEX a XSL-FO [Switching from LATEX to XSL-FO]; pp. 41–53

[Published in *TUGboat* 29:1.]

Claudio Beccari, La virgola intelligente [Smart decimal separator]; pp. 54–56

The decimal fractional part of a number must be separated from the integer part by means of a decimal separator. ISO regulations specify a different sign for different languages; the internal LATEX mathematical character codes do not help treating this sign in a simple way. Here we describe a few ways to handle this problem.

Claudio Beccari, The unknown `picture` environment; pp. 57–64

The old `picture` environment, introduced by Leslie Lamport into the LATEX kernel almost 20 years ago, appears to be neglected in favor of more modern and powerful LATEX packages that eliminate all the drawbacks of the original environment. Nevertheless, it is still being used behind the scenes by a number of other packages. Lamport announced an extension in 1994 that should have removed all the limitations of the original environment; in 2003 the first version of this extension appeared, in 2004 the first stable version was released; in 2009 it was extended with new functionality. Nowadays the `picture` environment can perform similarly to most simple drawing programs, but it has special features that make it invaluable.

Luigi Scarso, Extending ConTEXt MkIV with PARI/GP; pp. 65–74

This paper shows how to build a binding to PARI/GP, a well known computer algebra system, for ConTEXt MkIV, showing also some examples on how to solve some common basic algebraic problems.

Eventi e novità, Events and news; p. 75

ArsTEXnica #12 (October 2011)

Gianluca Pignalberi, Editoriale [From the editor]; p. 5

Claudio Beccari, XƎLATEX and the PDF archivable format; pp. 6–11

Up to now XƎLATEX produces a final PDF output file but it gets this output by transforming an XDV (extended DVI) intermediate file. This excludes most of the possibilities offered by pdfLATEX that, at least since version 1.40.9, and with the help of an extension file `pdfx.sty`, can directly produce a PDF/A-1b compliant output. This paper explains how to get through this bottleneck by resorting to the ubiquitous Ghostscript program.

Agostino De Marco and Roberto Giacomelli, Creare grafici con `pgfplots` [How to create plots with `pgfplots`]; pp. 12–38

This article presents `pgfplots`, the package for the creation of plots based on PGF. The user manuals of both these extensions of the LATEX language are very detailed and voluminous and users are often discouraged from studying these two documents.

In particular, those wishing to create high quality graphs have a difficult time disentangling the details of the many options and customizations. The article suggests a practical approach: It will be shown how a graph can be prepared by means of a correct initial setting of axes, thus introducing the basic commands and options. Then the way to plot curves and surfaces will be explained, suggesting how to possibly customize their appearence.

Agostino De Marco and Paolo Eugenio Cresci, LaTeX nella pubblica amministrazione: La web application FAciLE per la produzione automatica di comunicazioni interne del Comune di Napoli [LaTeX in the public administration: The FAciLE web application to automatically produce City of Naples internal letters]; pp. 39–56

This article describes the web application FAciLE in use at the administration of the City of Naples. The application was released in March 2011. It is integrated into the intranet software tools of the municipality and is currently under testing. Users are able to produce official internal communication letters which comply with the City of Naples Corporate Identity specifications. The application is designed to be perceived as a very simple tool and presents itself to the user as a web form for the production of a PDF document. Behind the scenes FAciLE runs the XeLaTeX typesetting engine together with an ad hoc parser developed in the PHP language. The key aspect of the application is the design of a LaTeX source template based on the `scrlttr2` document class from the KOMA-script bundle.

Presentazioni da ArsTeXnica numero 11, Presentations from ArsTeXnica no. 11; p. 57

Jean-Michel Hufflen, An introductory presentation of XSL-FO; p. 58

This short statement aims to sketch the broad outlines of the presentation at the GuIT 2011 meeting.

[Received from Gianluca Pignalberi.]

Die TeXnische Komödie 3/2011

Die TeXnische Komödie is the journal of DANTE e.V., the German-language TeX user group (`http://www.dante.de`). [Editorial items are omitted.]

Dominik Wagenführ, Variable Argumente in LaTeX nutzen [Using variable arguments with LaTeX]; pp. 10–20

By defining one's own commands in LaTeX, reoccuring tasks and formats can be created easily without having to write the same code again and again. Another advantage is that if a change is needed only the command itself needs to be adjusted and not individual places inside the document. This article shows how, with the help of `xkeyval`, optional arguments can be used without getting lost. The `xkeyval` package is an extension of the `keyval` package which can be used for most examples of the article as well. There are also other packages such as `pgfkeys` and `kvoptions`/`kvsetkeys` that provide key–value combinations for options.

Axel Kielhorn, Viele Ziele — Multi-Target Publishing [Many targets — Multi-target publishing]; pp. 21–32

[Translation published in this issue of *TUGboat*.]

Enrico Gregorio, Installation of TeX Live 2011 on Ubuntu; pp. 33–45

[Published in *TUGboat* 32:1]

Reinhard Kotucha, Installation nicht ganz freier Fonts [Installation of not-completely free fonts]; pp. 46–48

During the work on TeX Live 2005 some Type 1 fonts had been discovered which had to be removed due to license restrictions. Although these fonts may be used freely, the licence requires that no money must be charged for their distribution, which cannot be prevented during distribution on CD or DVD. Since these fonts were accidentally available in earlier TeX Live versions one must assume they have been used in various documents. While discussing the matter with Karl Berry the idea was born to provide the fonts via network installation.

Rolf Niepraschk, Experimentelle Trennmuster [Experimental hyphenation patterns]; pp. 49–51

In the following it is shown how the new experimental hyphenation patterns for LaTeX can be used in a document. For background and further information please consult the respective German documentation.

Herbert Voss, Latin Modern Math; pp. 52–60

More than once this journal has reported on the use of mathematical fonts with (La)TeX. The only — more or less complete — fonts containing text- and mathematical characters for pdfLaTeX are Computer Modern and `kpfonts`, which are limited to 256 characters per font. All other mathematical fonts matching frequently used text fonts such as Times Roman, Helvetica, Lucida and Palatino are not public domain. The revision of Computer Modern led to Latin Modern for which a complete mathematical font, the Latin Modern Math, is available. It is part of TeX Live 2011 and completes the available glyphs of the Latin Modern family.

Herbert Voss, LuaLaTeX und Schriften [LuaLaTeX and fonts]; pp. 62–67

Depending on the operating system the number of available fonts may be so huge that seeing an overview may be hard. In this article we show how a Lua function can display the available fonts of an OpenType or TrueType font family, together with a test string.

[Received from Herbert Voß.]

The PracTeX Journal 2011-1

The PracTeX Journal is an online publication of the TeX Users Group. Its web site is `http://tug.org/pracjourn`. All articles are available there.

Issue theme: LaTeXniques.

Lance Carnes, Editorial — LaTeXniques

Editor's introduction to the issue.

The Editors, News from Around

TeX and LaTeX blog; Typography video blog; Internet 20th Anniversary.

Fabien Leboeuf, LaTeX patient summaries

In this article we demonstrate the LaTeX techniques used to produce high-quality technical reports with multimedia features. At our hospital, the analysis of biomechanical data is done by a multidisciplinary team and involves a large amount of inter-related information in a variety of electronic formats. Therefore, it is essential that there be a user-friendly interface to present this data to the team for analysis. In this context, LaTeX was used to create a comprehensive report, with all text, graphics, and video contained in a single PDF file. We have been using LaTeX to produce this report since 2006, and have received positive feedback from the hospital staff.

L. Garcia-Forte and C. Leon-Hernandez and C. Rodriguez-Leon, Integrating LaTeX and Moodle questionnaires

Creating teaching material requires the generation of both static (unreactive) data-documents and dynamic (reactive) program-documents based on different technologies. Teaching a subject often implies the maintenance of a large number of both types of documents, usually written in a variety of languages and stored in different formats. Ergo, a natural goal for the lecturer is to minimize the amount of work invested during the development and maintenance of the material. There are acceptable solutions regarding the transformation between formats with the same kind of reactivity. This work discusses the problem of integrating Moodle (an open source learning management system) and LaTeX (a document preparation system), proposes a methodology to pursue this goal, and presents a tool to assist in the translation of Moodle Quiz documents to LaTeX.

Ryan Higginbottom, Teaching LaTeX at a liberal arts college

This brief report describes a course I developed for teaching LaTeX to a diverse undergraduate audience. Of special note are the changes and improvements I made to this class after the first time it was taught.

Lenore Horner, LaTeX teaching techniques

As first a physics professor and now a math and physics high school teacher, my teaching materials are always evolving and I am always looking for ways to make this easier for myself and to avoid reinventing the wheel (often my own wheel). Over the last three years, LaTeX has been a key part of that process.

Robert Ipanaqué Chero and Gloria Solvey Crespo Guerrero, Tesis de pregrado en LaTeX con FcUnp class [FcUnp LaTeX thesis style]

FcUnp is a LaTeX class for writing the bachelor thesis used at the Science Faculty of the National University of Piura, Perú. The goal of FcUnp is to provide a bachelor thesis format with a consistent layout that conforms to the rules of the Faculty so that students can concentrate solely on the content. It provides a set of commands to create the cover page, the title page, the signatures page, the dedication page and the acknowledgments page. When required, there is another set of commands to create the conclusions, the annexes, the appendices, and the abstract. In addition, this class allows generating PDF output, using either `dvipdfm` or directly with `pdflatex`. (Article in Spanish.)

Claudio Beccari, Intelligent commas

The decimal fractional part of a number must be separated from the integer part with a decimal separator. The ISO regulations specify a different sign for different languages; the internal LaTeX mathematical character codes do their best to avoid a simple treatment of the decimal separator. Here we describe a few ways to handle this problem.

Lenore Horner, Speedy LaTeX on the Mac

Here are the Mac-specific tools I use to make the typesetting as fast as possible so I can spend my time on content rather than on formatting.

The Editors, Ask Nelly

Page numbers in bibliography?; Teacher vs. student course materials?

The Editors, Distraction: Course outline

TEX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at `http://tug.org/consultants.html`. If you'd like to be listed, please see that web page.

Aicart Martinez, Mercè
Tarragona 102 4^o 2^a
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) ono.com`
Web: `http://www.edilatex.com`
We provide, at reasonable low cost, LATEX or TEX page layout and typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990. For more information visit our web site.

Dangerous Curve
PO Box 532281
Los Angeles, CA 90053
+1 213-617-8483
Email: `typesetting (at) dangerouscurve.org`
Web: `http://dangerouscurve.org/tex.html`
We are your macro specialists for TEX or LATEX fine typography specs beyond those of the average LATEX macro package. If you use XƎTEX, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical TEX and LATEX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a TEX book.

Hendrickson, Amy
Brookline, MA, USA
Email: `amyh (at) texnology.com`
Web: `http://www.texnology.com`
LATEX macro writing our speciality for more than 25 years: macro packages for major publishing companies, author support; journal macros for American Geophysical Union, Proceedings of the National Academy of Sciences, and many more.

Hendrickson, Amy (cont'd)

Scientific journal and e-journal design and production.

LATEX training, at MIT, Harvard, many more venues. Customized on site training available.

Please visit our site for samples, and get in touch. We are particularly glad to take on adventurous new uses for LATEX, for instance, web based report generation including graphics, for bioinformatics or other applications.

Latchman, David
4113 Planz Road Apt. C
Bakersfield, CA 93309-5935
+1 518-951-8786
Email: `david.latchman (at) gmail.com`
Web: `http://www.elance.com/s/dlatchman`
Proficient and experienced LATEX typesetter for books, monographs, journals and papers allowing your documents and books to look their possible best especially with regards to technical documents. Graphics/data rendered either using Ti*k*Z or Gnuplot. Portfolio available on request.

Peter, Steve
295 N Bridge St.
Somerville, NJ 08876
+1 732 306-6309
Email: `speter (at) mac.com`
Specializing in foreign language, multilingual, linguistic, and technical typesetting using most flavors of TEX, I have typeset books for Pragmatic Programmers, Oxford University Press, Routledge, and Kluwer, among others, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. In addition, I've helped publishers write, maintain, and streamline TEX-based publishing systems. I have an MA in Linguistics from Harvard University and live in the New York metro area.

Shanmugam, R.
No. 38/1 (New No. 65), Veerapandian Nagar, Ist St.
Choolaimedu, Chennai-600094, Tamilnadu, India
+91 9841061058
Email: `rshanmugam92 (at) yahoo.com`
As a Consultant, I provide consultation, training, and full service support to individuals, authors, typesetters, publishers, organizations, institutions, etc. I support leading BPO/KPO/ITES/Publishing companies in implementing latest technologies with high level automation in the field of Typesetting/Prepress, ePublishing, XML2PAGE, WEBTechnology, DataConversion, Digitization, Cross-media publishing, etc., with highly competitive prices. I provide consultation in building business models &

Shanmugan, R. (cont'd)
technology to develop your customer base and community, streamlining processes in getting ROI on our workflow, New business opportunities through improved workflow, Developing eMarketing/E-Business Strategy, etc. I have been in the field BPO/KPO/ITES, Typesetting, and ePublishing for 16 years, handled various projects. I am a software consultant with Master's Degree. I have sound knowledge in TEX, LATEX2ε, XMLTEX, Quark, InDesign, XML, MathML, DTD, XSLT, XSL-FO, Schema, ebooks, OeB, etc.

Sievers, Martin
Im Treff 8, 54296 Trier, Germany
+49 651 4936567-0
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`
As a mathematician with ten years of typesetting experience I offer TEX and LATEX services and consulting for the whole academic sector (individuals, universities, publishers) and everybody looking for a high-quality output of his documents.

From setting up entire book projects to last-minute help, from creating individual templates, packages and citation styles (BIBTEX, biblatex) to typesetting your math, tables or graphics — just contact me with information on your project.

Veytsman, Boris
46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`
TEX and LATEX consulting, training and seminars. Integration with databases, automated document preparation, custom LATEX packages, conversions and much more. I have about sixteen years of experience in TEX and twenty-nine years of experience in teaching & training. I have authored several packages on CTAN, published papers in TEX related journals, and conducted several workshops on TEX and related subjects.

TUG Institutional Members

American Mathematical Society, *Providence, Rhode Island*

Aware Software, Inc., *Midland Park, New Jersey*

Banca d'Italia, *Roma, Italy*

Center for Computing Sciences, *Bowie, Maryland*

Certicom Corp., *Mississauga, Ontario, Canada*

CSTUG, *Praha, Czech Republic*

diacriTech, *Chennai, India*

Florida State University, School of Computational Science and Information Technology, *Tallahassee, Florida*

IBM Corporation, T J Watson Research Center, *Yorktown, New York*

Institute for Defense Analyses, Center for Communications Research, *Princeton, New Jersey*

LAMFA CNRS UMR 6140, *Amiens, France*

MacKichan Software, Inc., *Washington/New Mexico, USA*

Marquette University, Department of Mathematics, Statistics and Computer Science, *Milwaukee, Wisconsin*

Masaryk University, Faculty of Informatics, *Brno, Czech Republic*

MOSEK ApS, *Copenhagen, Denmark*

New York University, Academic Computing Facility, *New York, New York*

Springer-Verlag Heidelberg, *Heidelberg, Germany*

StackExchange, *New York City, New York*

Stanford University, Computer Science Department, *Stanford, California*

Stockholm University, Department of Mathematics, *Stockholm, Sweden*

University College, Cork, Computer Centre, *Cork, Ireland*

Université Laval, *Ste-Foy, Québec, Canada*

University of Ontario, Institute of Technology, *Oshawa, Ontario, Canada*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

Calendar

2012

Jan 27	"The Design of Understanding", St. Bride Library, London, England. `stbride.org/events`
Jan 31	PracTeX Journal 2012-1, deadline for submission of articles on "LaTeX in the IT world".
Mar 7 – 9	DANTE Frühjahrstagung and 46th meeting, HTWK Leipzig, Germany. `www.dante.de/events/dante2012.html`
Apr 29 – May 3	BachoTeX 2012: 20th BachoTeX Conference, Bachotek, Poland. `www.gust.org.pl/bachotex/2012`
May 1	**TUG 2012** presentation proposal deadline. `tug.org/tug2012`
May 15	**TUG 2012** early bird registration deadline. `tug.org/tug2012`
Jun 1	**TUG 2012** preprints deadline. `tug.org/tug2012`
Jun 4 – Jul 27	Rare Book School, University of Virginia, Charlottesville, Virginia. Many one-week courses on type, bookmaking, printing, and related topics. `www.rarebookschool.org/schedule`
Jun 26 – 29	SHARP 2012, "The Battle for Books", Society for the History of Authorship, Reading & Publishing. Dublin, Ireland. `www.sharpweb.org`
Jun 30 – Jul 1	The Tenth International Conference on the Book, Universidad Abat Otiba CEU, Barcelona, Spain. `booksandpublishing.com/conference-2012`
Jul 9 – 13	"Towards a Digital Mathematics Library" (DML 2012), Bremen, Germany. `www.fi.muni.cz/~sojka/dml-2012.html`

TUG 2012
Boston, Massachusetts.

Jul 16 – 18	The 33rd annual meeting of the TeX Users Group. Presentations covering the TeX world. `tug.org/tug2012`

Jul 16 – 22	Digital Humanities 2012, Alliance of Digital Humanities Organizations, University of Hamburg, Germany. `www.digitalhumanities.org/conference`
Jul 31 – Aug 5	TypeCon 2012, Milwaukee, Wisconsin. `www.typecon.com`
Aug 5 – 9	SIGGRAPH 2012, Los Angeles, California. `s2012.siggraph.org`
Aug 23 – 26	TeXperience 2012 (5th TeXperience Conference, organized by CSTUG), Malenovice, The Czech Republic. `katedry.osu.cz/kma/texperience2012`
Oct 8 – 12	EuroTeX 2012 and the sixth ConTeXt user meeting, Breskens, The Netherlands. `meeting.contextgarden.net/2012`
Oct 10 – 14	Association Typographique Internationale (ATypI) annual conference, Hong Kong. `www.atypi.org`

Status as of 29 November 2011

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 206 203-3960, e-mail: `office@tug.org`). For events sponsored by other organizations, please use the contact address provided.

A combined calendar for all user groups is online at `texcalendar.dante.de`.

Other calendars of typographic interest are linked from `tug.org/calendar.html`.

TUG 2012

The 33rd Annual Meeting of the TeX Users Group
Presentations covering the TeX world

July 16–18, 2012 ▪ Boston, Massachusetts, USA

http://tug.org/tug2012 ▪ tug2012@tug.org

April 30, 2012 — bursary application deadline
May 1, 2012 — presentation proposal deadline
May 15, 2012 — early bird registration deadline
June 1, 2012 — preprint submission deadline
July 16–18, 2012 — conference
July 30, 2012 — deadline for final papers

Sponsored by the TeX Users Group and DANTE e.V.