TUGboat

Volume 37, Number 1 / 2016

TeX Users Group

TUGboat (ISSN 0896-3207) is published by the TeX Users Group. Web: `http://tug.org/TUGboat`.

Memberships and subscriptions

2016 dues for individual members are as follows:

- Regular members: $105.
- Special rate: $75.

The special rate is available to students, seniors, and citizens of countries with modest economies, as detailed on our web site. Also, anyone joining or renewing before March 31 receives a $20 discount.

- Regular members (early bird): $85.
- Special rate (early bird): $55.

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 detailed 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 for 2016 is $110.

Institutional membership

Institutional membership is primarily a means of showing continuing interest in and support for TeX and the TeX Users Group. It also provides a discounted membership rate, site-wide electronic access, and other benefits. For further information, see `http://tug.org/instmem.html` or contact the TUG office.

Trademarks

Many trademarked names appear in the pages of *TUGboat*. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which commonly appear in *TUGboat* should not be considered complete.

TeX is a trademark of American Mathematical Society.
METAFONT is a trademark of Addison-Wesley Inc.
PostScript is a trademark of Adobe Systems, Inc.

[printing date: April 2016]

Printed in U.S.A.

Addresses

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

Telephone
+1 503 223-9994

Fax
+1 815 301-3568

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`

We make the mistake of referring to a specific computer program as a tool. . . .
Each tool becomes several tools and each requires an underlying body of knowledge for it to be used successfully.
And there is very little that is user friendly.

Frank Romano
Too Many Tools, in *Color Publishing*
(Volume 1, Number 4, Winter 1991)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
Editor Barbara Beeton

Volume 37, Number 1 • 2016
Portland • Oregon • U.S.A.

TUGboat editorial information

This regular issue (Vol. 37, No. 1) is the first issue of the 2016 volume year.

TUGboat is distributed as a benefit of membership to all current TUG members. It is also available to non-members in printed form through the TUG store (`http://tug.org/store`), and online at the *TUGboat* web site, `http://tug.org/TUGboat`. Online publication to non-members is delayed up to one year after print publication, to give members the benefit of early access.

Submissions to *TUGboat* are reviewed by volunteers and checked by the Editor before publication. However, the authors are assumed to be the experts. Questions regarding content or accuracy should therefore be directed to the authors, with an information copy to the Editor.

Submitting items for publication

Proposals and requests for *TUGboat* articles are gratefully received. Please submit contributions by electronic mail to `TUGboat@tug.org`.

The second 2016 issue will be the proceedings of the TUG'16 conference (`http://tug.org/tug2016`); the deadline for receipt of final papers is August 8. The third issue deadline is September 29.

The *TUGboat* style files, for use with `plain` TeX and LaTeX, are available from CTAN and the *TUGboat* web site, and are included in common TeX distributions. We also accept submissions using ConTeXt. Deadlines, templates, tips for authors, and other information is available at:
`http://tug.org/TUGboat/location.html`

Effective with the 2005 volume year, submission of a new manuscript implies permission to publish the article, if accepted, on the *TUGboat* web site, as well as in print. Thus, the physical address you provide in the manuscript will also be available online. If you have any reservations about posting online, please notify the editors at the time of submission and we will be happy to make special arrangements.

Other TUG publications

TUG is interested in considering additional manuscripts for publication, such as manuals, instructional materials, documentation, or works on any other topic that might be useful to the TeX community in general.

If you have such items or know of any that you would like considered for publication, send the information to the attention of the Publications Committee at `tug-pub@tug.org`.

TUGboat advertising

For advertising rates and information, including consultant listings, contact the TUG office, or see:
`http://tug.org/TUGboat/advertising.html`
`http://tug.org/consultants.html`

Trademarks

TUG 2016 ∽ Toronto, Canada
July 25–27, 2016
excursions before and after
http://tug.org/tug2016

Editorial comments

Barbara Beeton

R.I.P. Sebastian Rahtz, 1955–2016

SPQR — Sebastian Patrick Quintus Rahtz — succumbed to a long illness on 15 March 2016. He came by the memorable initials honestly, as the fifth child of a professor in archaeology.

Sebastian was involved in more signal TEX projects than one can imagine — TEX Live, two LATEX Companions (Graphics and the Web), hyperref and many other packages,[1] as well as numerous *TUGboat* articles.[2] He gave a good account of himself in a 2009 interview.[3]

Professionally, he started out to follow the family tradition in archaeology, and was involved in typesetting books on classical subjects, but became fascinated by the capabilities of TEX when it became available in the early 1980s. His efforts to make TEX available to others led him to work on the UK TEX Archive, and on CTAN when that succeeded the UK Archive. From this emerged the TEX Live distribution, which he created, named, and edited until 2004. (The full TEX Live Guides for the 3rd and 4th editions[4] were published in *TUGboat*.) As a key contributor to the architecture of the TEX Directory Structure,[5] his legacy remains important to TEX users to this day.

I first met Sebastian at a TEX meeting in the UK sometime during the 1980s. His enthusiasm and dedication were obvious, as were his sharp intelligence and dry wit. By the 1990s, he was active in the organized user groups, joining the TUG board from 1994–1997, becoming secretary in 1995 and chair of the technical council in 1996, relinquishing both positions at the end of his board tenure.

He was a prime instigator in arranging for the first TUG meeting in Kerala, India, in 2002, and for the formation of TUG India.

TUG 2000 took place at Wadham College, Oxford University, where by then Sebastian was on the computing support staff, having been brought there to "work on bringing documentation into a common format using XML".[3] Unlike TEX, XML can be validated (i.e., forced to conform to a predefined structure); for scholarly publications especially in the humanities, this is not as restricting as it is for mathematics, and even for math, a number of publishers have adopted its use. The XML bug bit Sebastian quite as effectively as the TEX bug had earlier, and with less time to devote to TEX, he threw himself fully into the effort to develop and encourage the use of TEI (the Text Encoding Initiative) among academics.

Sebastian will be greatly missed both by the TEX and TEI communities and by his former colleagues.

George Greenwade, 1956–2003

Very belatedly, we learned of the death of George Greenwade, a former board member (1994–1997) and treasurer of TUG (1994–1995). He died unexpectedly on 2 August 2003.

George was an economist, on the faculty of Sam Houston State University, in Huntsville, Texas. He had, for the early 1990s, an unusually strong interest in the possibilities of the World Wide Web for economics, and established a website that quickly became recognized as an important resource both by the economics community and by *PC Magazine*, which featured it on a map of outstanding internet sites in 1994.

Along with interest in the Web came an interest in effective communication and publication, hence George's interest in TEX. The rationale for and realization of CTAN were a product of these interests, introduced by George at the 1993 annual meeting and published in the proceedings.[6] At first, CTAN was based on the FTP protocol, and the site `ftp.shsu.edu` quickly became a familiar target for exploration. Even now, George's paper makes for interesting reading, identifying the participants in the archive effort (including the part played by Sebastian Rahtz).

George's participation as an active member of the TUG board was cut short by responsibilities of his primary field; we were sorry to lose his energy and

[1] `http://ctan.org/author/rahtz`

[2] `http://tug.org/TUGboat/Contents/listauthor.html#Rahtz,Sebastian`

[3] `http://tug.org/interviews/rahtz.html`

[4] `ftp://tug.org/historic/systems/texlive/1998/tldoc/live.pdf`
The early Guides included an exhaustive annotated listing of the contents, a feature no longer needed now that on-line documentation is so readily available.

[5] `http://tug.org/TUGboat/tb16-4/tb49tds.pdf`

[6] `http://tug.org/TUGboat/tb14-3/tb40green.pdf`

expertise, but his legacy remains in an irreplaceable resource for the TEX community.

Peter Breitenlohner, 1940–2015

Peter's legacy to DANTE and his German friends is honored elsewhere in this issue by Joachim and Marion Lammarsch. Here we can reveal one of Peter's activities that was purposely kept "under wraps".

Don Knuth has, since the creation of TEX, offered a reward to the first finder of a bug in the code. Peter was the recipient of a number of such checks, including one of the last "big ones"; this is mentioned in Knuth's *Digital Typography*, on page 658 of "The final errors of TEX".

Behind the recognition of errors is a corps of knowledgeable volunteers, individuals whose understanding of TEX internals is thorough enough that Don trusts them to vet reports and provide explanations that will demonstrate to the submitters of non-bugs that their finding does not pass muster. For many years, Peter was one of these "go-to" volunteers. His analysis of bug reports was thorough, accurate, and understandable by even a non-programmer. That was my principal area of interaction with Peter, and it made the task of "TEX entomologist" much easier and more enjoyable. Thank you, Peter.

biblatex — Request for feedback

The biblatex team are looking for feedback on what *real* users are doing, particularly in terms of what back-ends they are using. A *very* short survey is posted at `https://surveymonkey.com/r/X2FWPNR` If you are a biblatex user, your participation would be most welcome, per Joseph Wright. (And spread the word, please.)

No "drop dead" date was given, but it may be assumed that feedback should be registered no later than 1 July 2016.

LATEX courses for credit

It seems rare enough that formal courses in (LA)TEX are offered in convenient locations, but the idea that such a course might be offered for credit is even more surprising — although it would certainly be a useful study topic for someone pursuing a degree in math or physics, for example. It has come to light that there are in fact such courses in a few places.

A course at the University of Verona offers two credits, and is well attended; 66 students were counted at the first lecture and an additional 20 were expected for the second (scheduling conflicts precluded their attendance the first week). At Cedarville University (Cedarville, Ohio), a course in Technical Writing using LATEX earns one semester hour of credit; the current professor is "second generation", having been taught by the former professor who taught the class for 18 years before his retirement. In some other places, although no credit is given, the material is presented as a "soft skill" course which is required to graduate.

If anyone reading this knows of other places where such courses are offered, please send feedback. We will try to compile a list and make it available for reference.

Cooper Type

While checking Don Knuth's "news" page to get more information about an "All Questions Answered" session in Stockholm, I noticed that he will be giving a public lecture on Tuesday, May 17, at the San Francisco Public Library on "32 Years of METAFONT". This is part of the Type@CooperWest program organized by Sumner Stone.

CooperWest is the west coast branch of the Cooper Union postgraduate certificate program in typeface design. The "main office", so to speak, is the Continuing Education department of The Cooper Union in New York City, where a parallel lecture program sponsored by the Herb Lubalin Study Center takes place. Many of the New York–based lectures have been filmed and are made available on line. The listing of topics is wide-ranging, and can be explored at `http://coopertype.org/lectures/nyc`. (The first one I viewed is entitled "How Aldus Manutius Saved Western Civilization", a lecture by G. Scott Clemons. A good place to start.)

There's no indication that the west coast lectures are also being recorded, but we can hope.

More typography videos

Listed below are a few videos on type-related topics that I've stumbled upon recently or were recommended by friends. Such pointers will be collected on a page on the TUG website for easy reference, at `http://tug.org/video`. I'll go back and include relevant items that have been mentioned in previous columns.

- "Teaching to See", by Inge Druckrey, is posted on Edward Tufte's website. `http://www.edwardtufte.com/tufte/`
- David Brailsford tells the story of the "jailbreak" of the Linotron 202. `https://www.youtube.com/watch?v=CVxeuwlvf8w` and `https://www.youtube.com/watch?v=HdModNEK_1U`
- The trailer for a movie "Carl Dair at Enschedé, The last days of metal type". (The full movie will be shown as part of the program at TUG 2016.)

https://www.sheridancollege.ca/news-and-events/typeforming.aspx

- A TED talk, "Pace matters", by stone-carver Nick Benson. https://www.youtube.com/watch?v=A8IeEYwVQSA
- Casting type: Five videos by Stan Nelson showing how punches are made and how type produced using them. https://www.youtube.com/playlist?list=PLD1C918AD04AF88E0

If you come across other videos along these lines, please let us know!

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

Peter Breitenlohner, 1940–2015

Joachim Lammarsch, Marion Lammarsch

With deep regret and heavy sadness I learned that our DANTE member Peter Breitenlohner unexpectedly passed away in October. Peter wasn't only a member, but a helper whenever there was a need.

Peter's TeXnical story started before the time when the DANTE e.V. association was founded. He was involved in TeX when it was published. He found several errors in the software, thus one of the people who got several checks from Donald E. Knuth (DEK). He developed a program to manipulate DVI files and also a special TeX version which was able to write from right to left. Once DEK commented that probably Peter knew the TeX code better than he himself.

When I got the PublicTeX software from Klaus Thull and was looking for people who were willing to help me to revise the software and to distribute it to our members, he joined immediately. His overriding interest was to make the TeX system better. He developed a improved TeX version called ε-TeX which was able to act like the original TeX, but contained a number of TeXnical innovations to improve and to make it better.

Furthermore, when we started to design NTS, Peter joined immediately to help. In 1994, during the TUG conference in Santa Barbara, California, when I organized an appointment with DEK, for him it wasn't any problem to drive quickly 300 miles to Stanford. Together with other colleagues they discussed the idea of NTS with DEK, and after that he drove back to UCSB. He was a very active member of the NTS team.

First published in *Die TeXnische Komödie* 1/2016, pp. 10–11. Translation by the first author. Reprinted with permission.

It would be easy to tell you a lot more from Peter. During the time when I was the president of DANTE e.V. he was always available. He was absolutely modest, as we learned when we tried to get official money for the NTS project from the European Union. There were some professors in different counties who were willing to support the project, but unfortunately none from Germany. So we were sitting in the ZOO restaurant considering what to do. Until Peter unexpectedly asked us: why don't you ask me. And so after several years we found out that Peter was a professor.

Peter had his very special humor and everyone who knew him a little bit better, knew about the special history about Professor Kabelschacht. What exists behind that story one can read on the Internet or ask at one of the next DANTE meetings. [Ed. note: Prof. Kabelschacht also published once in *TUGboat*: vol. 8, no. 2, pp. 184–185, tug.org/TUGboat/tb08-2/tb18kabel.pdf.] At DANTE meetings we always enjoyed these stories from Peter.

Here also we will miss him as a special individual as well as active member in the TeX world. For myself and for many other people in our association a man passed away who we monumentally appreciated. But just as much for many of us he was more than a good colleague: he was a very good friend.

◇ Joachim Lammarsch,
Marion Lammarsch
jola (at) psychologie dot uni-heidelberg dot de

The continuing TUG membership drive

Boris Veytsman

Any organization is only as strong as its membership. Thus one of the important metrics of the health of TUG is the number of our members. The historical data for this metric are shown in Figure 1. They tell a worrying story of a steady decline. There are many possible explanations for this decline. One of the most probable might be the growth of the Internet. In the 1990s, the best decade for TUG, the membership was provided a number of clear benefits. First, there were CDs (and later DVDs) with the latest TeX distributions, which every TUG member got in the mail. Second, *TUGboat* and TUG conferences were among the rare possibilities for exchanging TeX experiences and learning about TeX.

Nowadays, almost everyone can download a TeX system for free: the traffic cost is (usually) low and the connections are (usually) fast. While *TUGboat* is still a great journal (full disclaimer: I am an Associate Editor, so my opinion may be biased), and TUG meetings remain highly useful, the growth of virtual meeting places like StackExchange has created many alternatives. It is now easily possible to become a knowledgeable TeX user without being a TUG member. Thus while TeX usage as measured by CTAN downloads and traffic on TeX-related sites is far from declining, TUG membership is slowly dropping.

Nonetheless, it seems that the role of TUG as the steward and advocate of the development of TeX and TeX-related software is still important (this role was the topic of a heated discussion at the Darmstadt conference; see Stefan Kottwitz, "TUG 2015 conference report", in *TUGboat* 36:2, `tug.org/TUGboat/tb36-2/tb113kottwitz.pdf`). The loss of *TUGboat* and TUG meetings would be felt by the community. Thus the TUG Board and the Membership Committee are trying to make our society known and attractive to the new generation of TeX users.

In doing so we try to learn from the experience of other professional societies. For some time, the American Chemical Society has conducted a very interesting campaign which recruits the existing members to promote ACS. Those members who attract new ones get modest thank you gifts. At the end of 2014 we on the Membership Committee decided to try this approach.

Thus, we started the 2015 membership campaign, publicizing the following rules:

1. Any new member can indicate on the membership form the name of the existing member who invited or inspired her to join TUG (the "inviter").
2. Each inviter gets a beautiful postcard printed by Peter Wilson on a letterpress (Figure 2).
3. At the end of the campaign a randomly chosen winner would get the main prize: a specially commissioned drawing by Duane Bibby (Figure 3). All inviters except members of the TUG board would be eligible for the drawing, with a chance of winning proportional to the number of invited members; each institutional member counted as eight individual ones.

Inviter	Points
Kai Von Fintel	1
Julian Gilbey	1
Jim Hefferon	0
Humberto Madrid	1
Fritz Scholz	1
Juan Luis Varona Malumbres	1
Edward Baudrez	1
Peter J. Pupalaikis	1
Francoise Mulhauser	1
Joseph Wright	1
Jean-luc Doumont	1
Michael Henning Juul Stæhr	1
Robert L Knighten	1
C.V. Radhakrishnan	8
JFQA	1

Table 1: Inviters in 2015. The "points" column corresponds to drawing rules, see text.

The campaign was announced at the end of 2014 and continued throughout 2015. At the end of the year, we counted fifteen inviters, two of them bringing institutional members. Evidently the campaign contributed to the small uptick at the right end of the curve on Figure 1.

The inviters are listed in Table 1. The column "Points" corresponds to final drawing:

1. Jim Hefferon and C.V. Radhakrishnan each invited an institution, which gives eight points in the scoring.
2. Jim, being a board member, was ineligible for the final prize, so in the end he got zero points.

I volunteered to devise the procedure for the final drawing. I wanted it to be open source, verifiable and TeX-related. At the end I decided to use the popular free program *R* (`r-project.org`). The package *knitr* provides a literate way to program in *R* and outputs the results in LaTeX, so it is well suited

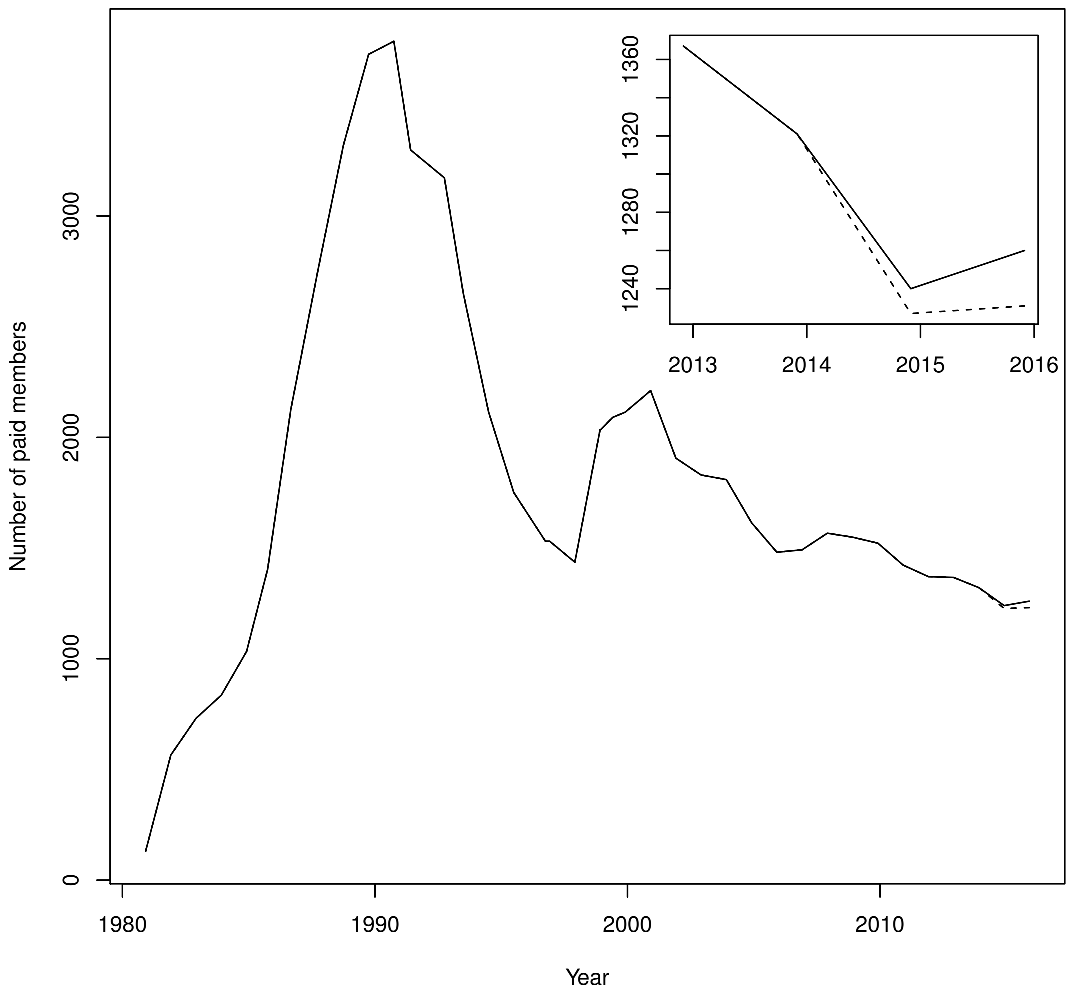


Figure 1: TUG membership at the end of a calendar year (I am grateful to Karl Berry and Robin Laakso for the data for this figure). The dotted line is the predicted membership without invited members.

for the task (I reviewed a book about the package in *TUGboat* 35:1, 2014, `tug.org/books/reviews/tb109reviews-xie.html`). *R* has a handy function `sample` which does weighted random sampling from a given list of entries. The only remaining problem was verifiability. To achieve it we need to explicitly set the seed of the random number generator. Using the current version of TeX (from the current TeX Live) written as an integer seemed a good idea.

The full program in *R* for the random drawing and its result are shown below:

```
inviters <-
  c('Kai Von Fintel',
    'Julian Gilbey',
    'Jim Hefferon',
    'Humberto Madrid',
    'Fritz Scholz',
    'Juan Luis Varona Malumbres',
    'Edward Baudrez',
    'Peter J. Pupalaikis',
    'Francoise Mulhauser',
    'Joseph Wright',
    'Jean-luc Doumont',
    'Michael Henning Juul St\\ae hr',
    'Robert L Knighten',
    'C.V. Radhakrishnan',
    'JFQA'
    )
weights <- c(1, 1, 0, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 8, 1)
```

```
set.seed(314159265)
winner <- sample(inviters,
                 size=1,
                 prob=weights)
cat('And the winner is: \\bfseries',
    paste0(winner, '!'))
```

And the winner is: **Jean-luc Doumont!**

Congratulations to the winner and many thanks to all participants!

Onward into 2016

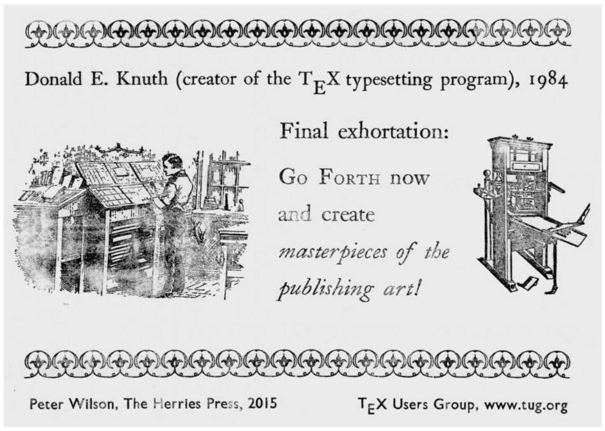
Now to the present. It seems that our 2015 campaign had at least some success. Thus we decided to extend it through 2016. We again ask TUG members to invite TEX users and typography lovers to join TUG. As in the previous year, each inviter will get Peter Wilson's beautiful postcard (Figure 2).

Some words about the main prize, which, as in the previous year, will be given to a randomly-chosen winner at the end of 2016. In 2016, it will be a limited edition book, *Manuale Zapficum*, printed in 2009 on the occasion of the ninetieth birthdays of Hermann Zapf and Gudrun Zapf (Hermann's death in 2015 was a profound loss for the worldwide typographic community). The book was designed by Jerry Kelly. About it, the publisher says (`ritpress.rit.edu/publications/books/manuale-zapficum.html`):

> The *Manuale Zapficum*'s innovative specimen pages employ timeless Zapf faces such as Diotima, Optima, Palatino, and Zapfino, while including fresh uses of proprietary typefaces such as Hallmark Uncial and Hallmark Textura. A variety of the specimens were letterpress printed using historic metal type from the Cary Collection. Each is printed in traditional red and black on Hahnemühle Biblio paper. The Campbell-Logan Bindery of Minneapolis, MN, bound an edition of 100 copies in luxurious vellum and Fabriano paper.

We need to make our community stronger. A healthy and vibrant TUG is one of the ways that supports our technology and ensures the future of TEX as well as computer typography. Thus let me finish this note by asking our members to go forth and spread the word about TEX and TUG.

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



Donald E. Knuth (creator of the TEX typesetting program), 1984

Final exhortation:

Go Forth now and create *masterpieces of the publishing art!*

Peter Wilson, The Herries Press, 2015 TEX Users Group, www.tug.org

Figure 2: Peter Wilson's postcard, for all inviters.

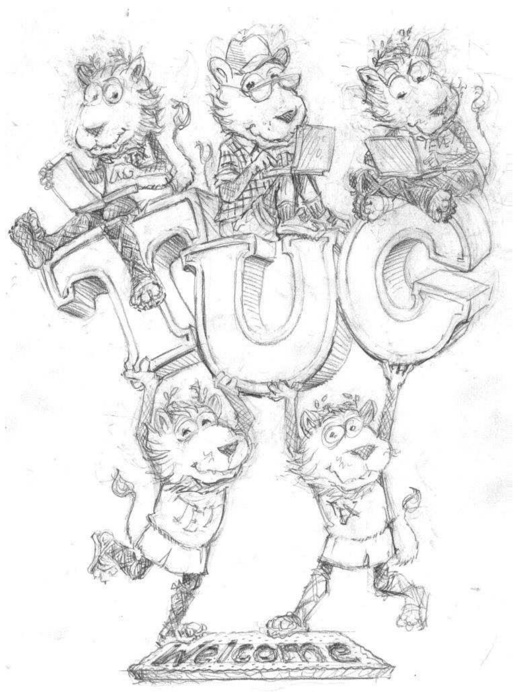


Figure 3: 2015 main prize: Duane Bibby's drawing.

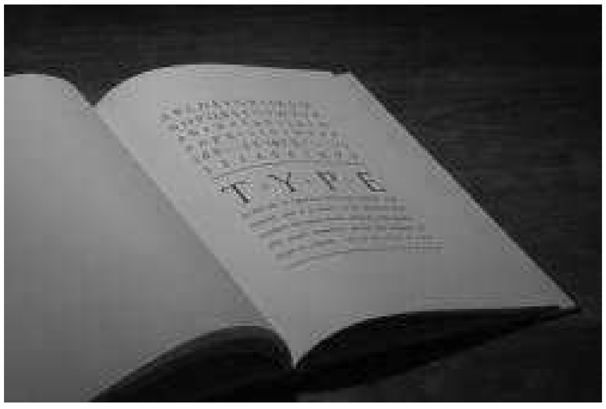


Figure 4: 2016 main prize: *Manuale Zapficum.*

ATypI 2016 with GUST participating

Jerzy B. Ludwichowski

Abstract

A note concerning the planned GUST participation at ATypI'16, the annual *Association Typographique Internationale* conference in Warsaw, Poland.

GUST at ATypI'16

October 2016 in Warsaw will see ATypI'16, the annual conference of *Association Typographique Internationale*, with the theme of *Convergence* (`atypi.org`). GUST has been invited to participate, thanks to Andrzej Tomaszewski and Adam Twardoch, both well known to BachoTEX participants.

The details of our participation are still in the making at this writing, with a deadline of March 31. The presentations will be rehearsed at BachoTEX 2016 (`gust.org.pl/bachotex`).

The following are planned:

- A workshop or rather demonstration of automation with TEX, using presentation and testing of fonts as an example. This should give the participants an idea of the possibilities for using TEX for similar tasks, e.g., dictionaries, catalogs, mass production like invoices, multimedia publications (print, PDF, HTML, EPUB, et al.).
- Another workshop or demonstration: *A lesson in math font anatomy*, i.e., math fonts in general and detail, based on a particular font, its anatomy, constituent alphabets and additional information within the MATH table.
- At the *Technology forum*, which should have a larger audience, a communique by Bogusław Jackowski on math fonts, and ways of automating their production, with a short mention of MetaType1: block schema and an overview of the guiding principles.
- Also planned for the *Technology forum*, a presentation by Piotr Strzelczyk on the merits and demerits of Unicode in connection with typography.
- At the *General session*, a two-voiced presentation *Slurs, Braces, Radicals — Obstacle or Challenge* with *1. Obstacle* by Bogusław Jackowski and *2. Challenge* by Marek Ryćko. This is about a "philosophy of typography", a proposal of a new typesetting paradigm, in the making by both gentlemen since some time. BachoTEX regulars should know what to expect. We hope the presentations to be both eye- and mind-opening.
- Also at the *General session*, a short promotional presentation on TEX, GUST, the e-foundry and its math fonts, most likely presented by myself.
- GUST will also be present at the *Fonts for Science* exhibition, with math fonts posters.

There is an undercurrent of activity directed toward some other members of the TEX math fonts community to coerce them into participating, but nothing more is settled yet.

A little bit of history

This will be the second ATypI conference in Warsaw. The first, in 1975, is notable for two events.

Peter Karow and Hermann Zapf for the first time publicly presented the idea of outline fonts.

The other was due to Marek Decowski and Ryszard Dulewicz from the Polish Academy of Sciences, who described and presented software for a scanner for texts typed with Frutiger's OCR-B. The scanner was used in a semi-automated process at DSP, a printing house in Warsaw. Poland, after the US and UK, became the third country producing OCR scanners.

There is a connected story involving Adrian Frutiger and Roman Tomaszewski, Andrzej Tomaszewski's father, a renowned Polish printing master and editor. Roman Tomaszewski was ATypI delegate for Poland and its Board Member in the years 1965–1992. Under his leadership *Ośrodek Pism Drukarskich* (then Poland's central institution for typefaces) produced *Litera*, a magazine for which such prominent figures as, among others, Jan Tschichold, Adrian Frutiger, Hermann Zapf, Max Caflisch or Matthew Carter wrote. They all were Tomaszewski's colleagues and often friends. *Ośrodek Pism Drukarskich* complemented Frutiger's OCR-B with Polish diacritics. The problem was to get the so-augmented IBM Selectric Typewriter printing heads to Poland, past the then-prevailing COCOM import restrictions. Frutiger caused some twenty such heads to be made at the IBM factory in Paris, and simply put them into the pockets of a Polish Airlines pilot, to be received by Roman Tomaszewski at the Warsaw airport.

At the 1975 Warsaw ATypI conference, Frutiger was given a medal commemorating 500 years of printing in Poland. Only a few participants were aware of the underlying main feat.

◇ Jerzy B. Ludwichowski
Migdałowa 21
87-100 Toruń
Poland
`Jerzy.Ludwichowski at gmail dot com`

Letters

Editor's note: Material printed in the Letters section does not necessarily reflect the opinion of the editors or the TUG board and is the sole responsibility of the authors.

The letter below has not been edited in any way.

The Board's suspension of the President

Jonathan Fine

1 The crisis

Kaveh Bazargan took office as President of TUG in May 2015. I was shocked to read the Board's announcement, on 13 October, that they had suspended him from office. This, if not reversed, would lead to his removal. I had many questions and some advice to offer.

On 13 December, I wrote to acting President Jim Hefferon (the defeated candidate for President) and the Board. My letter is the next section. I got no reply other than acknowledgement. The rest of the article discusses the situation.

By the way, I first joined TUG in about 1990, was on the TUG Board 2009–13, and am a former Chair of UK TUG. I rejoined TUG this year.

This article uses primary documents published by Kaveh, via a link on the comp.text.tex newsgroup. The Board has published no such documents.

2 My message to the TUG Board

Dear Jim

You are, while the elected President Kaveh Bazargan is suspended, acting President on the TeX Users Group. Please ask the TUG Board to think again about this suspension. First some background.

In May 2015 Bazargan was elected President of TUG, 307 votes to 110. It was the first contested election since 2005. He took office in July 2015. You were Bazargan's opponent in this election. The Board's decision has reversed the outcome of the election. This is a difficult situation. Also, you are potentially in a conflict of interest.

I do not know if Bazargan has exercised his right to ask the Board to reconsider. I hope he has, and that the Board treats this email as additional information they should consider.

Is the Board sure that it has behaved properly? Please ask the Board to carefully consider the following five issues. I know that they are somewhat procedural, and don't get to the heart of the matter. However, they raise real concern that the Board itself has not got to the heart of the matter.

1. On 20 October, on behalf of the Board, you told me that the Board voted on

> whether Kaveh is "deemed to be no longer working in the interests of TUG" and thus should be suspended [...]

The TUG Bylaws require this decision be made on the basis of the Board members actions (or inactions). But the motion voted on gives no examples of actions (or inactions) that justify the conclusion that Bazargan was not working in the interests of TUG. Is it fair and reasonable?

I'm not aware of any actions (or inactions) done by Bazargan, in his capacity as President of TUG, that would justify his being removed from office. The Board's statement, after the suspension, does not provide any.

2. Also on 20 October you wrote "FYI, per the TUG bylaws, this was a special vote, not a normal motion." The TUG Bylaws allow the Board to conduct business by email. The procedure is proposal of a motion, seconder, one-week discussion period, voting for up to a week. There is no provision for "a special vote".

It seems that the procedure in the Bylaws was not followed. If not followed then perhaps Bazargan was substantially disadvantaged, and the vote taken is null and void. If so, this would leave Bazargan as President.

3. A law-suit between Bazargan and CV Radhakrishnan (CVR) was central to the Board's case for suspension. This, like any other business interest, can give rise to a conflict of interest. According to `https://www.councilofnonprofits.org/tools-resources/conflict-of-interest`,

> A conflict of interest policy should (a) require those with a conflict (or who think they may have a conflict) to disclose the conflict/potential conflict, and (b) prohibit interested board members from voting on any matter in which there is a conflict.

Did the Board attempt to apply such principles, to ensure that Bazargan's business interests did not conflict with his responsibilities to TUG? Again, the Board's statement does not give any examples of where it might.

4. It seems that CVR has a close relationship with at least one Board member, and that through that relationship CVR gave the Board information that was used to damage to Bazargan. Did any Board members disclose any potential conflict of interest relating to the suspension of Bazargan, and if so what action did the Board take in response?

5. TUG is US 501(c)(3) tax-exempt charitable organisation, which gives tax benefits for donations to TUG. It also means that TUG is not a 'members club', owned by and organised for the benefit

of its members. The TUG Board, arguably, suspended the President to help protect a member of TUG from legal action. Did the Board take legal or other expert advice, particularly on conflict of interest, before suspending the President?

By the way, if the Board it did not take and follow such advice, then Board members individually might not be protected by any directors and officers liability insurance held by TUG, in respect to their suspending the President. In other words, they might be personally liable for any harm done.

Please ask the Board to consider carefully the five issues above. I intend to publish this letter, and I hope the Board will publish a response.

One final point. The Board's statement on the suspension of President said that it "potentially affects the entire TeX community". I am not a member of TUG (although I am a past Board member). Please do not use this as a reason to not communicate with me. I am affected by this, as are many other non-members of TUG.

3 The Board's case

On 6 October 2015 someone sent an email message from `board@tug.org` to Kaveh Bazargan, the President of TUG. It wrote:

> As TUG president, you have a duty to represent all TUG members to the best of your ability (just as we do as TUG directors). It is not possible to fulfill this responsibility when you are involved in a lawsuit against another TUG member.

This is the basis for the Board's actions. The message closes by threatening suspension if Kaveh does not resign. It is signed 'TUG Directors'.

4 The duties of a TUG Board members

TUG is a USA 501(c)(3) tax-exempt non-profit organisation. Its articles of incorporation state that it "shall be operated *exclusively* for charitable, educational and scientific purposes" associated with technical typesetting, font design and so forth.

The US National Council of Nonprofits states that the legal duties of nonprofit Board members are (I summarize, and adapt for TUG):

- **Duty of due care** in use of TUG assets, overseeing its activities, and advancing TUG effectiveness and sustainability.
- **Duty of loyalty** to the best interests of TUG (rather than personal interests).
- **Duty of obedience** to laws, ethical practices, and promotion of typesetting, font design, etc.

There is no duty to avoid lawsuits with other members, unless it follows from these three duties.

5 The legal action

Kaveh and CVR were in business together. A falling out led to a dispute, which Kaveh has taken to court for resolution. This is, as Kaveh's appeal says, a universal right. The Board's action interferes with this right. It may even bring TUG into the lawsuit.

Imagine the mischief that could be caused if a nonprofit organisation, such as the Linux Foundation, had as a Bylaw that a Board member involved a lawsuit with another member must resign.

6 Conflict of interest

When a Board member's personal interest conflicts with his duties as a Board member the conflicted party should report it, and the *rest of the Board* should remove that member from the decision. The lawsuit, like any other business relationship, could give rise to a conflict of interest.

Surely there is a better way to cure the lawsuit headache, than to cut off TUG's head, its President.

7 TUG Bylaws

The Board members have a duty of obedience to the Bylaws of TUG. They allow the Board to conduct business by email. The procedure is proposal of a motion, seconder, one-week discussion period, with voting for up to a week.

The Bylaws also provide every TUG member with access to Board minutes. On 6 October 2015 someone used `board@tug.org` to send a message, in the name of 'The Directors', threatening Kaveh with suspension. I would like to see the motion that authorised this cowardly action.

If there is none, then is the suspension be a legal action of the Board? Perhaps Kaveh never stopped being President of TUG.

8 Conclusion and questions

Kaveh Bazargan was elected 307 to 110 in the first contested election since 2005, and the acting President is now Kaveh's opponent Jim Hefferon. Did the Board represent to the best of their ability the roughly 25% of members who voted for Kaveh? Did CVR get special and preferential treatment?

Please think carefully as to whether the Board members followed their duties of due care, loyalty, and obedience. Put it another way, have they behaved properly? Did the suspension promote TUG's charitable, educational, and scientific purposes? Is TUG now more effective and sustainable, as a result of this decision?

◇ Jonathan Fine
Milton Keynes
England
`jfine2358@gmail.com`

The libertine gets mathematical

Khaled Hosny

Linux Libertine is a popular libre font family (released under the SIL Open Font License) and one of the early typefaces to be designed from the ground up as a libre project. Together with its organic sans-serif companion, Linux Biolinum, they serve as an excellent type family for many scholarly works.

Due to their attractiveness, Linux Libertine and Linux Biolinum fonts are often used for mathematical and other scientific works heavy in mathematical typesetting, but the lack of a mathematical companion forced people to try to find matching mathematical typefaces with varying degrees of success. A mathematical companion was thus one of the most requested features for this type family.

In late 2012 I started playing with the idea of creating a mathematical companion for this family using the existing text characters for basic mathematical alphabets, adding an OpenType MATH table, and some symbols needed for basic mathematical typesetting. I got the basics working, but then had to put it on the back burner for a while. In 2014 TUG showed interest in funding the development of this mathematical companion and work resumed. Development took a while, as usual, but in January 2016 the typeface was finally released.

Linux Libertine and Linux Biolinum are big typefaces with thousands of glyphs that were developed over the course of almost 10 years, and it had accumulated many issues over the time. While working on the mathematical companion, I couldn't help but try to fix many of these issues, and after a while it became clear that this is becoming a full fork of the original project, not just an additional member of the family.

With this in mind, I went ahead and made it a full fork. Since the original typefaces use the OFL "reserved names" clause, I had to rename my modified fonts. I first wanted to just drop the "Linux" part and release under the name "Libertine" (why should a typeface be named after an operating system kernel?), but it turns out I can't use either of the words "Linux" or "Libertine" in my fork, not just the combined name. Frédéric Wang suggested the Latin word "Libertinus" which I liked much, so Linux Libertine became Libertinus Serif and Linux Biolinum became Libertinus Sans, with the brand new mathematical companion being named Libertinus Math.

Libertinus Math is an OpenType font with a MATH table, so it can be used only with a mathematical typesetting engine that supports such fonts,

$$(x+\alpha)^n = \sum_{k=0}^{n} \binom{n}{k} x^k \alpha^{n-k}$$

$$|x| = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$\begin{aligned} \nabla \cdot \nabla\psi &= \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2} \\ &= \frac{1}{r^2 \sin\theta} \left[\sin\theta \frac{\partial}{\partial r} \left(r^2 \frac{\partial\psi}{\partial r} \right) \right. \end{aligned}$$

Figure 1: Sample of Libertinus Math

Libertinus Serif
Libertinus Serif Italic
Libertinus Serif Semibold
Libertinus Serif Semibold Italic
Libertinus Serif Bold
Libertinus Serif Bold Italic
Libertinus Serif Display
Libertinus Sans
Libertinus Sans Italic
Libertinus Sans Bold
Libertinus Math
`Libertinus Mono`
Libertinus Keyboard

Figure 2: List of Libertinus fonts

like XeTeX, LuaTeX, Mozilla Firefox, and MS Office, among others. Unfortunately, it cannot be readily used with traditional TeX implementations that lack support for such fonts.

The work on the mathematical companion involved using the existing serif and sans-serif glyphs to provide corresponding mathematical alphabets, drawing a missing lower case for the blackboard alphabet, and adding many missing mathematical symbols and big variants to cover symbols used in plain TeX and the LaTeX kernel. Not all alphabets are covered, though. There are no script, fraktur, sans-serif bold italic, or typewriter alphabets.

The original Linux Libertine fonts were developed using free software, mainly FontForge. That tradition continues in this fork, but replacing FontForge by its fork Sorts Mill Editor (`bitbucket.org/sortsmill/sortsmill-tools`), and adding (a fork of) FontTools (`github.com/behdad/fonttools`) to the mix.

The new font files can be downloaded from:
`github.com/khaledhosny/libertinus/releases`
`ctan.org/pkg/libertinus`

◇ Khaled Hosny
`khaledhosny.org`

LaTeX News

Issue 24, February 2016

Contents

LuaTeX support

This release refines the LuaTeX support introduced in the 2015/10/01 release. A number of patches have been added to improve the behavior of ltluatex (thanks largely to code review by Philipp Gesang). The kernel code has been adjusted to allow for changes in LuaTeX v0.85–v0.88. Most notably, newer LuaTeX releases allow more than 16 write streams and these are now enabled for use by `\newwrite`, but also the experimental `newtoken` Lua library has been renamed back to `token` which required small adjustments in the LuaTeX setup.

The biggest change in LuaTeX v0.85–v0.87 compared to previous versions is that all the primitives (originally defined in pdfTeX) dealing with the PDF "back end" are no longer defined, being replaced by a much smaller set of new primitives. This does not directly affect the core LaTeX files in this release but has required major changes to the `.ini` files used by TeX Live and similar distributions to set up the format files. These changes in the LuaTeX engine will affect any packages using these back end commands (packages such as graphics, color, hyperref, etc.). Until all contributed packages are updated to the new syntax users may need to add aliases for the old pdfTeX commands. A new luapdftexalias package has been contributed to CTAN (not part of the core LaTeX release) that may be used for this purpose.

See also the sections below for related changes in the tools and graphics bundles.

Unicode data

As noted in LaTeX News 22, the 2015/01/01 release of LaTeX introduced built-in support for extended TeX systems. In particular, the kernel now loads appropriate data from the Unicode Consortium to set `\lccode`, `\uccode`, `\catcode` and `\sfcode` values in an automated fashion for the entire Unicode range.

The initial approach taken by the team was to incorporate the existing model used by (plain) XeTeX and to pre-process the "raw" Unicode data into a ready-to-use form as `unicode-letters.def`. However, the relationship between the Unicode Consortium files and TeX data structures is non-trivial and still being explored. As such, it is preferable to directly parse the original (`.txt`) files at point of use. The team has therefore "spun-out" both the data and the loading to a new generic package, unicode-data. This package makes the original Unicode Consortium data files available in the `texmf` tree (in `tex/generic/unicode-data`) and provides generic loaders suitable for reading this data into the plain, LaTeX 2_ε, and other, formats.

At present, the following data files are included in this new package:

- `CaseFolding.txt`
- `EastAsianWidth.txt`
- `LineBreak.txt`
- `MathClass.txt`
- `SpecialCasing.txt`
- `UnicodeData.txt`

These files are used either by LaTeX 2_ε or by expl3 (i.e. they represent the set currently required by the team). The Unicode Consortium provides various other data files and we would be happy to add these to the generic package, as it is intended to provide a single place to collect this material in the `texmf` tree. Such requests can be mailed to the team as usual or logged at the package home page: `https://github.com/latex3/unicode-data`.

The new approach extends use of Unicode data in setting TeX information in two ways. First, the `\sfcode` of all end-of-quotation/closing punctuation is now set to 0 (transparent to TeX). Second, `\Umathcode` values are now set using `MathClass.txt` rather than setting up only letters (which was done using an arbitrary plane 0/plane 1 separation). There are also minor refinements to the existing code setting, particularly splitting the concepts of case and letter/non-letter category codes.

For XeTeX, users should note that `\xtxHanGlue` and `\xtxHanSpace` are *no longer defined*, that no assignments are made to `\XeTeXinterchartoks` and that no `\XeTeXintercharclass` data is loaded into the format. The values which were previously inherited from the plain XeTeX setup files are *not* suitable for properly typesetting East Asian text. There are third-party packages addressing this area well, notably those in the CTeX bundle. Third-party packages may need adjustment to load the data themselves; see the unicode-data package for one possible loader.

More support for east European accents

As noted in LaTeX News 23, comma accent support was added for `s` and `t` in the 2015/10/01 release. In this release a matching `\textcommaabove` accent has been added for U+0123 (`\c{g}`, ģ) which is the lower case of U+0122 (`\c{G}`, Ģ). In the OT1 and T1 encodings the combinations are declared as composites with the `\c` command, which matches the Unicode names "latin (capital|small) letter g with cedilla" and also allows `\MakeUppercase{\c{g}}` to produce `\c{G}`, as required. In T1 encoding, the composite of `\c` with `k`, `l`, `n` and `r` are also declared to use the comma below accent rather than cedilla to match the conventional use of these letters.

The UTF-8 `inputenc` option `utf8` has been extended to support all latin combinations that can be reasonably constructed with a (single) accent command an a base character for the T1 encoding so ģ, ų and similar characters may be directly input using UTF-8 encoding.

Changes in Graphics

The changes in LuaTeX v0.87 mean that the color and graphics packages no longer share the `pdftex.def` file between LuaTeX and pdfTeX. A separate file `luatex.def` (distributed separately) has been produced, and distributions are encouraged to modify `graphics.cfg` and `color.cfg` configuration files to default to the `luatex` option if LuaTeX v0.87 or later is being used. The team has contributed suitable `.cfg` files to CTAN to be used as models.

Normally it is best to let the local `graphics.cfg` automatically supply the right option depending on the TeX engine being used; however the color and graphics (and so graphicx) packages have been extended to have an explicit `luatex` option comparable to the existing `pdftex` and `xetex` options.

The trig package has been updated so that pre-computed values such as sin(90) now expand to digits (`1` rather than the internal token `\@one` in this case). This allows them to be used directly in PDF literal strings.

Changes in Tools

LuaTeX from version v0.87 no longer supports the `\write18` syntax to access system commands. A new package shellesc has been added to tools that defines a new command `\ShellEscape` that may be used in all TeX variants to provide a consistent access to system commands. The package also defines `\write18` in LuaTeX so that it continues to access system commands as before; see the package documentation for details.

Improving support for Unicode engines

Stability concerns are always paramount when considering any change to the LaTeX 2$_\varepsilon$ kernel. At the same time, it is important that the format remains usable and gives reliable results for users. For the Unicode TeX engines XeTeX and LuaTeX there are important differences in behavior from classical (8-bit) TeX engines which mean that identical default behaviors are not appropriate. Over the past 18 months the team has addressed the most pressing of these considerations (as detailed above and in LaTeX News 22 and 23), primarily by integrating existing patches into the kernel. There are, though, important areas which still need consideration, and which *may* result in refinements to kernel support in this area in future releases.

The default font setup in LaTeX 2$_\varepsilon$ at present is to use the `OT1` encoding. This assumes that hyphenation patterns have been read using appropriate codes: the `T1` encoding is assumed. The commonly-used hyphenation patterns today, hyph-utf8, are set up in this way for 8-bit engines (pdfTeX) but for Unicode engines use Unicode code points. This means that hyphenation will be incorrect with Unicode engines unless a Unicode font is loaded. This requires a concept of a Unicode font encoding, which is currently provided by the fontspec package in two versions, `EU1` and `EU2`. The team is working to fully understand what is meant by a "Unicode font encoding", as unlike a classical TeX encoding it is essentially impossible to know what glyphs will be provided (though each slot is always defined with the same meaning). There is also an overlap between this area and ideas of language and writing system, most obviously in documents featuring mixed scripts (for example Latin and Cyrillic).

As well as these font considerations, the team is also exploring to what extent it is possible to allow existing (8-bit) documents to compile directly with Unicode engines without requiring changes in the sources. Whether this is truly possible remains an open question.

It is important to stress that changes will only be made in this area where they do *not* affect documents processed with ε-TeX/pdfTeX (i.e. documents which are written for "classical" 8-bit TeX engines). Changes will also be made only where they clearly address deficiencies in the current setup for Unicode engines

LaTeX News

Issue 25, March 2016

LuaTeX

This LaTeX release sees several internal changes designed to ensure that the system is still usable with LuaTeX versions greater than 0.80, which have introduced many changes into the engine, most notably the removal or renaming of most of the primitive commands introduced by pdfTeX. Also the lists of Lua callbacks handled by the callback allocation mechanism has been updated to match the callbacks defined in LuaTeX version 0.90.

These changes have also required updates in tools and amsmath as described below.

This is the first release of LaTeX for which the test suite reports no failures when used with LuaTeX.

Documentation checksums

The doc package has always provided two mechanisms that were mainly intended to guard against file truncation or corruption when files were commonly distributed by email through unreliable mail gateways: a Character Table of the ASCII character set could be inserted (and checked) and a "checksum" (count of the number of backslashes in the code sections) could be checked. These features are not really needed with modern distribution mechanisms and can be a distraction when reading the source code and so have been removed. The doc package has been updated so that if you use a `\CheckSum` command then, as before, the number is checked; however, if you omit the command then no error or warning is given.

Updates to inputenc

The UTF-8 support in inputenc has been further extended with support for non-breaking hyphens and more dashes.

Updates in Tools

The varioref package has been updated with improved documentation of multilingual support, and avoiding unnecessary warnings in some cases with `\reftextfaraway`.

The tabularx package's handling of `\endtabularx` in environment definitions has been fixed to again match its documentation.

The bm package has been updated as required by the changes to `\mathchardef` in LuaTeX.

amsmath

Since the launch of LaTeX 2_ε in 1993, the amsmath bundle has been part of the *required* packages in the core LaTeX distribution, with bug reports handled by the LaTeX bug database at `https://latex-project.org/bugs-upload.html`.

The amsmath packages and the amscls classes have been maintained by the American Mathematical Society.

With this release a new arrangement has been agreed between the American Mathematical Society and the LaTeX3 project. The LaTeX3 project will take over maintenance of the amsmath bundle, with the American Mathematical Society retaining maintenance of amscls.

The recommended installation of these files in the TeX directory structure remains unchanged as `tex/latex/amsmath` and `tex/latex/amscls` respectively.

This release of amsmath includes several updates so that amsmath does not generate errors when math is used with LuaTeX v0.87+, which has changes to `\mathchardef` that are incompatible with the previous version of amsmath. It also improves `\dots` handling so that `\long` macros are correctly handled (for example, `\dots \Rightarrow` now uses centered dots), as well as commands expanding to character tokens (for example, `\times \dots \times` will use centered dots with `\times` defined as in the unicode-math package).

Related updates

In addition to the updates in the core LaTeX release, some files in the CTAN "contrib" area have also been updated. Notably there have been further updates to the unicode-data files; also, the files required to build plain and LaTeX formats have now been submitted to CTAN as tex-ini-files. The addition of a new `luatex` option for graphics-related packages (luatex-def on CTAN) has required updates to the configuration files to select a default option and these have similarly been uploaded to CTAN as graphics-cfg. (Previously these files were maintained directly in the TeX Live repository, and were not available on CTAN.)

On managing large documents

Thomas Thurnherr

Abstract

Whether you are working on a book, a doctoral thesis, or an extensive report, large documents can be difficult. The source code becomes confusing; you scroll back and forth to find what you are looking for; and processing the document can take minutes rather than seconds. In this article, I describe some tools and thoughts which might help to keep large documents organized so you can focus on the essential: the content.

1 Focus on one thing at a time

Focusing on one thing is generally more efficient than trying to do multiple things at the same time. For example, I find it better to work on the document layout, or edit a figure or table, or focus on writing a chapter. It does not mean that I work on the chapter until it is done. Rather, I concentrate on a chapter for a while and then switch to working on the document layout or a figure when I need a break from writing.

2 Keep things separate

A good way to keep focused is to have multiple source files, especially to keep the preamble separate from all contents. To that end, I often create a file `main.tex` with the preamble. In the preamble, I load all required packages, define new macros, set the header and footer, and configure other features of the document layout. Also in the main source file, I load all document contents. This is illustrated in the brief example below.

```
\documentclass{report} % main.tex
% Preamble
% Load packages and set document style
\usepackage{microtype}
\usepackage{biblatex}
\bibliography{references}
\...
% Main document
% Include all content
\begin{document}
        \include{titlepage}
        \include{contents} % toc/lof/lot
        \include{chapter1}
        \include{chapter2}
        \include{...}
        \printbibliography
        \include{appendix}
\end{document}
```

2.1 Include and input macros

In the example above, I used the `\include` macro. The macro loads content from a source file. For example, suppose I physically saved the content of the first chapter in `chapter1.tex`. Now, I can add its content to the document using `\include{chapter1}`. Moreover, `\include` adds a page break before the added content. This is great for chapters, which are physically separated from previous and subsequent contents. To add smaller chunks of content, I can use `\input{filename}` instead, which does not force page breaks. For example, `\input` is a good choice to add a formula or table from a separate source file. Moreover, this command is a great way to reuse code. Finally, the `\input` macro can be nested, which is not possible with `\include`.

Taken together, `\include` and `\input` help to stay on top of things in a large document by splitting the source into multiple files. An additional advantage is that I can reduce processing time of the document while working on it, by using `\includeonly` or by commenting out all `\include` and `\input` macros with content I am not currently editing.

3 Keep the project directory uncluttered

In order to keep the main project directory clean, I usually place figure files in a `figures` sub-folder and chapter source files in a `chapters` sub-folder. This helps to reduce the number of files in the main project directory. To load these files, I now have to provide their path as seen from the main project directory.

```
%Load figure from figures sub-folder
\includegraphics{figures/figure-filename}

%Load chapter from chapters sub-folder
\include{chapters/chapter1}
```

With some additional effort, I can move all meta-files into a `metafiles` sub-folder. Meta-files are files generated by the TeX engine and other programs invoked to generate the output. These have file endings: `log`, `aux`, `toc`, `lof`, `lot`, `bbl`, `bcr`, etc. To save meta-files to a sub-folder, `pdflatex` and other programs need to be called with various options. These are described in their respective manual pages.

4 Use a script for document processing

If you set out to write a lengthy document, it may be a good idea to learn `latexmk` [4]. The idea of `latexmk` was borrowed from `Makefiles`, which are frequently distributed with source code for C programs. A `latexmk Makefile` contains instructions on how to process a document by the TeX engine.

latexmk is distributed with major LaTeX distributions and therefore most likely available on your system. For more information, take a look at the documentation or search for tutorials and example Makefiles online.

Alternatively, with some basic bash or similar knowledge, you could write your own script or (original) Makefile to process the source. The script should at least have two options: 1) to typeset text only; and 2) to properly process references, the bibliography, and glossaries.

5 Mind the package order

Large documents frequently depend on a long list of packages. Sometimes the order in which packages are loaded matters, although this is generally less of a problem nowadays. One notable case remains, however: the hyperref package [2] tends to cause conflicts when used with other packages. As a rule of thumb, hyperref should be last, with some exceptions [6]. Usually, package conflicts are documented in the package documentation or online.

6 Labels and cross-referencing

A label is used to cross-reference a numbered element of a document. The quantity of labels increases with the size of a document and it quickly becomes difficult to remember all label names and to omit duplicates. It is good practice to use a label prefix, such as fig: for figures and tab: for tables, as in fig:workflow. There are no predefined prefixes, but often the first three letters of the command to reference are used. In addition, no prefix is recognized by the TeX program; their usage is entirely for the author's convenience. Finally, I have not seen prefixes used for bibliographic references. Instead, for BibTeX entries, I recommend concatenating the name of the first author, the year, and the first word of the title to form unique citation identifiers.

The showlabels package [5] can help keep track of labels. It prints their names in the margins of the processed document. The package either shows all labels (default) or only labels of specific commands (see example below). To generate the final version of a document, I can either manually remove the package or mute all its functions through its final option.

```
\usepackage[nolabel]{showlabels}
\showlabels{cite}
```

7 Draft mode

The draft option of the document class macro produces a visible mark for Overfull hbox warnings. These are positions in the document where the text or other content reaches into the margins. In addition, the option may alter the behavior of loaded packages. For example, with the draft option set, the graphicx package [1] indicates a figure with a black canvas instead of loading the actual figure. This can drastically reduce document processing time.

```
\documentclass[draft]{report}
```

The ifdraft package [3] allows additional customization of the behavior in draft mode. For example, I might want to define a todo command to mark unfinished content. By defining todo as empty command in final mode, I make sure no todo output is produced in the final version of the document.

```
\usepackage{xcolor}
\usepackage{ifdraft}
\ifdraft{% with draft option
        \newcommand{\todo}[1]{%
                \textcolor{red}{[TODO: #1]}}
}{% with final option
        \newcommand{\todo}[1]{}
}
```

References

[1] graphicx — enhanced support for graphics. https://www.ctan.org/pkg/graphicx. Accessed: 2016-02-25.

[2] hyperref — extensive support for hypertext in LaTeX. https://www.ctan.org/pkg/hyperref. Accessed: 2016-02-25.

[3] ifdraft — detect “draft” and “final” class options. https://www.ctan.org/pkg/ifdraft. Accessed: 2016-02-25.

[4] latexmk — fully automated LaTeX document generation. https://www.ctan.org/pkg/latexmk. Accessed: 2016-02-25.

[5] showlabels — show label commands in the margin. https://www.ctan.org/pkg/showlabels. Accessed: 2016-02-25.

[6] Collection of LaTeX package conflicts. http://www.macfreek.nl/memory/LaTeX_package_conflicts. Accessed: 2016-02-25.

⋄ Thomas Thurnherr
thomas.thurnherr (at) gmail dot com
http://texblog.org

`medstarbeamer`: A new beamer class

Anagha Kumar

Abstract

Beamer's popularity as a document class for creating slides has grown considerably since its release in 2003. This paper is intended to serve as a guide to beamer users on how to go about writing their own beamer document class. A new beamer document class, `medstarbeamer`, is presented as an example.

1 Introduction

With its many handy features, Till Tantau's beamer class is widely used for making presentations. While I find the pre-existing beamer themes pleasing, I recognize that not everyone shares the sentiment. Further, even those who find the existing themes satisfactory may feel the need to exercise fine control over the layout of their slides. While the beamer user guide is an excellent resource, I thought a paper outlining the process and providing an example of a beamer document class constructed from scratch would be a useful addition to the existing literature. This paper goes over `medstarbeamer`, an original beamer document class written by me. The package is available at `https://www.ctan.org/pkg/medstarbeamer`.

In addition to what is covered in this paper, a useful resource for those contemplating writing their own document class is the website `http://www.r-bloggers.com`. A post from November 2011 titled "Create your own Beamer template" provides a nice introduction to how beamer themes are typically written. In writing the beamer class, Till Tantau defined several inner, outer, color, and font themes. Extensive documentation on these can be found in the beamer user guide. Additional information is available at `https://en.wikibooks.org/wiki/LaTeX/Presentations`. Perhaps most importantly, the style files for these themes are easily downloadable from CTAN and can prove a valuable resource while customizing one's own theme.

In writing the `medstarbeamer` package, I chose to write a color theme (in the package, this is the style file `beamercolorthemeMedStarColors.sty`) and a separate `.cls` file (`medstarbeamer.cls`). Maintaining distinct files makes writing packages easier since it is less cumbersome than sorting through command after command in one very long `.cls` file. Further, users who have defined several themes of their own, be they inner, outer, color or font themes, find it easier to mix-and-match these themes while creating slides if the themes are saved in separate files. On the other hand, for simpler themes with fewer features, one file should suffice.

Section 2 covers how the document class was written, section 3 covers salient features of the color theme, and section 4 provides examples of usage. The reader is urged to download and compile the example presentation (`example.tex`) provided with the package on CTAN.

2 Writing the document class

I chose to write the document class by borrowing features from the beamer themes `infolines` and `miniframes`. The `.sty` files for both of these themes are downloadable from CTAN. To define the footline, I customized the `footline` command from the `infolines` theme. Here is the customized version:

```
\defbeamertemplate*{footline}{}{\hbox{%
  \begin{beamercolorbox}[center,
     wd=.3333333\paperwidth,ht=0.25cm,dp=0.2cm]
    {author in head/foot}%
    \usebeamerfont{author in head/foot}%
    \insertshortauthor
  \end{beamercolorbox}%
  \begin{beamercolorbox}[center,
     wd=.3333333\paperwidth,ht=0.25cm,dp=0.2cm]
    {title in head/foot}%
    \usebeamerfont{title in head/foot}%
    \insertshorttitle
  \end{beamercolorbox}%
  \begin{beamercolorbox}[center,
     wd=.3333333\paperwidth,ht=0.25cm,dp=0.2cm]
    {date in head/foot}%
    \usebeamerfont{date in head/foot}%
    \insertshortdate{}\hfill
     \insertframenumber{} /
     \inserttotalframenumber
  \end{beamercolorbox}%
}}
```

For my document class, I altered the height (`ht`) and depth (`dp`) and deleted some of the other features included in the `infolines` theme, such as the inclusion of the institute name. Since I feel more comfortable using (constant) centimeters as a metric of measurement, I used `cm` instead of `ex`. Additionally, the width for each `beamercolorbox` was set to one-third the paper width.

Notice the first `beamercolorbox` corresponds to the leftmost box at the bottom of the slides provided in section 4. The second `beamercolorbox` contains the title of the talk. Lastly, the third `beamercolorbox` contains the date as well as the current frame number. The reader should be aware that certain commands such as `totalframenumber` and `framenumber`, `shortauthor`, etc. have been predefined in the beamer document class.

While the code above gives the reader an idea of how the document class was constructed, it is also important for the reader to grasp how it's used. Here is the preamble from the `example.tex` file included with the package:

```
\documentclass{medstarbeamer}
\title[Applied Bayesian Data Analysis]
{Binary Response Regression and Model Selection}
\institute[]{}
\author[Anagha Kumar]{Anagha Kumar}
\date{August 26th, 2015}
```

One sees that the text provided in square brackets appears in the footline. The institute is deliberately left blank as I chose to exclude it from the footline in the construction of this document class.

I modified the headline from the `miniframes` theme similarly, as follows:

```
\defbeamertemplate*{headline}{}{%
  \begin{beamercolorbox}{section in head/foot}
    \vskip3pt\insertnavigation{\paperwidth}%
    \vskip3pt
  \end{beamercolorbox}%
  \ifbeamer@theme@subsection
    \begin{beamercolorbox}[ht=0.25cm,dp=0.2cm,
       leftskip=0.5cm]
      {subsection in head/foot}
      \usebeamerfont{subsection in head/foot}%
      \insertsubsectionhead
    \end{beamercolorbox}%
  \fi}
```

My headline is a greatly simplified version of the headline definition from `miniframes.sty`. Again, I altered the height (`ht`) and depth (`dp`) values to suit my liking. The first `beamercolorbox` corresponds to the topmost yellow (I defined a new color `medstaryellow` which will be described in detail in the following section) band in the slides presented in section 4. The section name (in bold) along with the other sections are presented in this topmost color box. Next, if a subsection exists, the second navy blue band (`medstarblue`) contains the subsection heading (notice the command `\insertsubsectionhead`).

Next, I customized the layout even further by specifying margin sizes:

```
\setbeamersizetext{margin left=0.5cm,
  text margin right=0.75cm}
```

I wanted to include the MedStar logo in every frame:

```
\logo{\includegraphics[height=1cm, width=2.5cm]
  {medstarlogo}}
```

I chose to suppress the navigation symbols and number the captions in my slide deck:

```
\setbeamertemplate{navigation symbols}{}
\setbeamertemplate{caption}[numbered]}
```

Finally, I defined two new commands as follows, `\sizecontentsoutline`, `\sizecontentscurrent`:

```
\newcommand{\sizecontentsoutline}[1]{%
  \ifnum#1=1
    \begin{frame}
      \tableofcontents
    \end{frame}
  \fi
  \ifnum#1=2
    \begin{frame}
      {\footnotesize \tableofcontents}%
    \end{frame}
  \fi
}
\newcommand{\sizecontentscurrent}[1]{%
  \AtBeginSection{%
    \ifthenelse{\thesection > 1}{%
      \ifnum#1=1
        \begin{frame}
          \tableofcontents[currentsection]
        \end{frame}
        \fi
     \ifnum#1=2
       \begin{frame}
         {\footnotesize
          \tableofcontents[currentsection]}%
       \end{frame}
      \fi}%
}}
```

Each command consists of two `if` statements which are set up such that if the parameter value is 1, then the table of contents (of the whole document for the first command, of the current section for the second) appears in the normal font. If the parameter is 2, on the other hand, the table of contents is reduced by using the font size `\footnotesize`. Users need only specify one of

```
\sizecontentsoutline{1}
\sizecontentsoutline{2}
```

or

```
\sizecontentscurrent{1}
\sizecontentscurrent{2}
```

respectively, in their `.tex` file. Notice that there is no need to enclose these commands in `\begin{frame}` and `\end{frame}` since these options have already been specified in the body of the command definitions. This option has proven a handy feature in presentations with several sections and subsections since a lengthy table of contents can be displayed in a smaller font size if needed.

Another useful feature of this document class is a new command called `class`. I wrote this command after noticing that when I give lectures, the students find it useful to have space for taking notes included in the handouts themselves. Therefore, I wrote a simple `if` statement:

```
\newcommand{\class}[1]{%
\ifnum #1=1
  \begin{frame}{Notes}\end{frame}
\fi
\ifnum #1=2
  \begin{frame}{Notes}\end{frame}
  \begin{frame}{Notes}\end{frame}
\fi
\ifnum #1=3
  \begin{frame}{Notes}\end{frame}
  \begin{frame}{Notes}\end{frame}
  \begin{frame}{Notes}\end{frame}
\fi}
```

The above command takes one parameter as an argument. The parameter specifies the number of blank slides to be inserted, up to three. I chose to limit the number of slides inserted at once to three since I have yet to encounter a situation where more than three pages of notes would be required to be taken by students after a given slide. That said, of course the code can easily be altered to suit individual requirements.

As for usage, one need merely type `\class{1}`, `\class{2}` or `class{3}` to insert the desired number of slides. As before, there is no need to enclose this command in `\begin{frame}` and `\end{frame}`.

3 Designing a color theme

My highest priority in writing this document class was to design an interesting color theme. I chose to use the MedStar logo as inspiration. The first step was to define my own colors. I elected to define three colors: `medstaryellow`, `medstarblue`, and `medstarred`. The colors were defined so as to match the colors in the MedStar logo. Should one want to use existing LaTeX colors, the website `http://latexcolor.com` provides color definitions and syntax for a very extensive range of colors and should prove an invaluable resource.

I chose to stick with white as the background color but users can alter this using the command

`\setbeamercolor{background canvas}{bg=`⟨*color*⟩`}`

Table 1 presents a summary of how the colors for the theme were specified. These commands are in the file `beamercolorthemeMedStarColors.sty`. The next section has examples of what the slides look like.

- The primary palette controls the foreground and background colors for the rightmost `colorbox` in the footline.
- The secondary palette controls the foreground and background for the leftmost `colorbox` in the footline, and the second strip in the headline.

Table 1: Summary of commands used in `beamercolorthemeMedStarColors.sty`.

Author	`\setbeamercolor{author} {fg=medstarblue}`
Date	`\setbeamercolor{date} {fg=medstarblue}`
Title	`\setbeamercolor{title} {fg=medstarblue}`
Subtitle	`\setbeamercolor{subtitle} {fg=medstarblue}`
Normal text	`\setbeamercolor{normal text} {fg=medstarblue}`
Captions	`\setbeamercolor{caption} {fg=medstarblue}`
Caption names	`\setbeamercolor{caption name} {fg=medstarblue}`
Items	`\setbeamercolor{item} {fg=medstarblue}`
Primary palette	`\setbeamercolor{palette primary} {fg=white, bg=medstarblue}`
Secondary palette	`\setbeamercolor{palette secondary} {fg=white, bg=medstarblue}`
Tertiary palette	`\setbeamercolor{palette tertiary} {fg=medstarblue,bg=medstaryellow}`
Date in footline	`\setbeamercolor {date in head/foot} {parent=palette primary}`
Author in footline	`\setbeamercolor {author in head/foot} {parent=palette secondary}`
Title in footline	`\setbeamercolor {title in head/foot} {parent=palette tertiary}`
Frame Title	`\setbeamercolor{frametitle} {fg=medstarblue,bg=medstaryellow}`

- The tertiary palette controls the foreground and background for the middle `colorbox` in the footline, and the topmost strip in the headline.

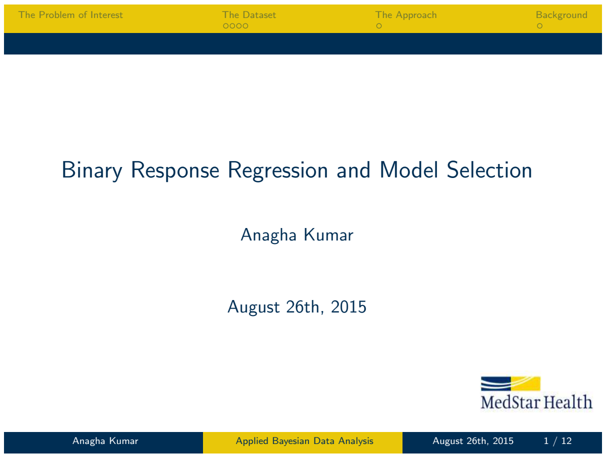
As shown in the table, the date, author, and title in the footline were set to the primary, secondary, and tertiary palettes respectively.

In addition to the above, sections, subsections, and subsubsections in the table of contents were set to `medstarblue` using the commands

```
\setbeamercolor{section in toc}
  {fg = medstarblue}
\setbeamercolor{subsection in toc}
  {fg = medstarblue}
\setbeamercolor{subsubsection in toc}
  {fg = medstarblue}
```

respectively. Similarly, I inserted a shaded table of contents at the beginning of each section, with the current and shaded sections shown in `medstarblue`, using the commands:

```
\setbeamercolor{section in toc shaded}
```


The Problem of Interest The Dataset The Approach Background

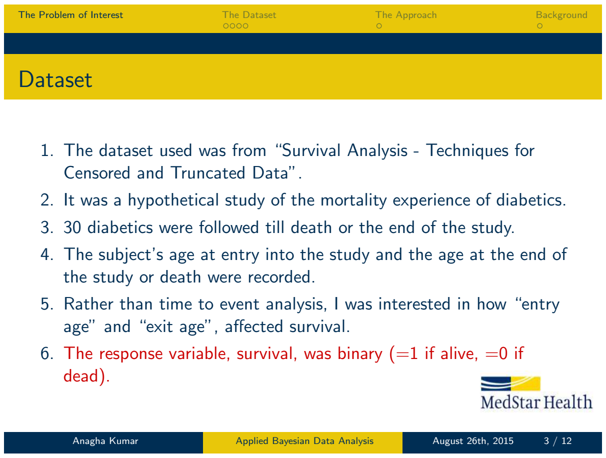
Binary Response Regression and Model Selection

Anagha Kumar

August 26th, 2015

Anagha Kumar | Applied Bayesian Data Analysis | August 26th, 2015 1 / 12

Figure 1: Title slide.

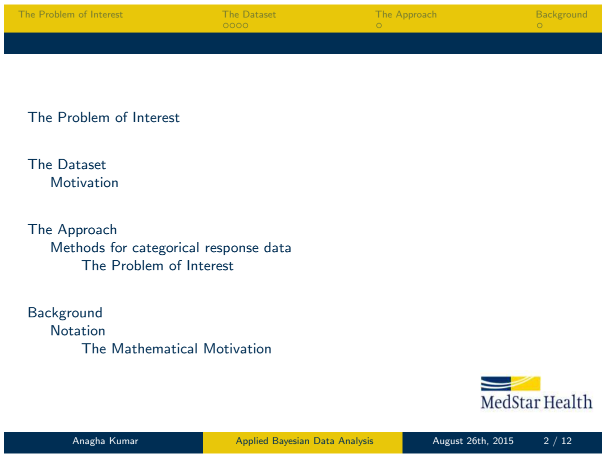

The Problem of Interest The Dataset The Approach Background

Dataset

1. The dataset used was from "Survival Analysis - Techniques for Censored and Truncated Data".
2. It was a hypothetical study of the mortality experience of diabetics.
3. 30 diabetics were followed till death or the end of the study.
4. The subject's age at entry into the study and the age at the end of the study or death were recorded.
5. Rather than time to event analysis, I was interested in how "entry age" and "exit age", affected survival.
6. The response variable, survival, was binary (=1 if alive, =0 if dead).

Anagha Kumar | Applied Bayesian Data Analysis | August 26th, 2015 3 / 12

Figure 3: Typical content slide, with title.


The Problem of Interest The Dataset The Approach Background

The Problem of Interest

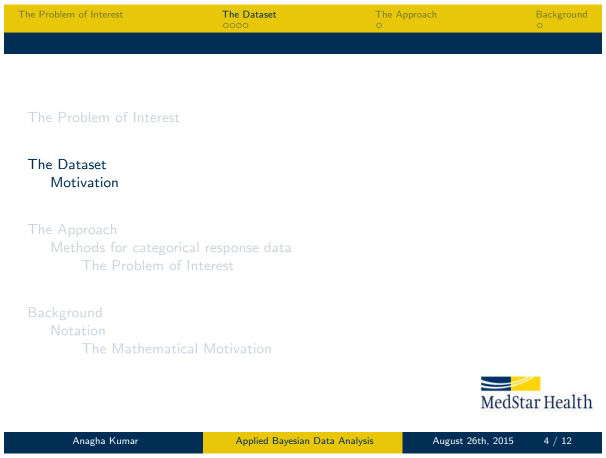
The Dataset
Motivation

The Approach
Methods for categorical response data
The Problem of Interest

Background
Notation
The Mathematical Motivation

Anagha Kumar | Applied Bayesian Data Analysis | August 26th, 2015 2 / 12

Figure 2: Overview slide.


The Problem of Interest The Dataset The Approach Background

The Problem of Interest

The Dataset
Motivation

The Approach
Methods for categorical response data
The Problem of Interest

Background
Notation
The Mathematical Motivation

Anagha Kumar | Applied Bayesian Data Analysis | August 26th, 2015 4 / 12

Figure 4: Shaded table of contents slide.

```
  {fg = medstarblue}
\setbeamercolor{subsection in toc shaded}
  {fg = medstarblue}
\setbeamercolor{subsubsection in toc shaded}
  {fg = medstarblue}
```

It is usually considered good practice to stick to two, or perhaps three, colors while designing a color theme though readers should feel free to tailor themes to their individual needs and preferences. Several other features can be fine-tuned in a given color theme. Readers can explore color themes at even greater length to construct one to their liking.

4 Examples

Figures 1, 2, 3, and 4 present examples of slides made using this document class. (Grayscaled for the printed *TUGboat*.)

5 Discussion

Since LaTeX users often need to customize their slides to suit institutional or personal needs, this paper intends to shed some light on the often daunting task of creating a custom beamer document class. By introducing a novel document class and detailing each command used in its construction, I hope to have provided a high-quality example of how to go about doing so. With the help of the many resources mentioned in this paper, users should find it possible to write their own beamer document classes.

As always, I urge users to upload their contributions to CTAN and accompany such uploads with appropriate documentation.

◇ Anagha Kumar
1711 35th Street NW Apt. 23
Washington DC 20007
`anaghakumar2405 (at) gmail dot com`

Glisterings: Assemblies; Table talk

Peter Wilson

It did a ghastly contrast bear
To those bright ringlets glistering fair.

Marmion, Sir Walter Scott

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.

Eric, or, Little by Little

Book title, Frederick W. Farrar

1 Assemblies

1.1 Adding to a macro

On occasions it is useful to be able to extend a pre-existing macro. For instance, to assemble a list of the names of the members of some organization, or the reviewers of some article, and then print them. In simple cases the LaTeX kernel
`\g@addto@macro{⟨macro⟩}{⟨addition⟩}`
can be used for this.

```
\makeatletter
\newcommand*{\member}[1]{%
  \@ifundefined{@members}{%
% a new list of members
% define it as the argument (member name)
    \newcommand{\@members}{#1}}{%
% a list exists, add the argument to it
    \g@addto@macro{\@members}{, #1}}}
\newcommand*{\showmembers}{%
  \ignorespaces\@members}
\newcommand*{\themembers}{\showmembers
  \let\@members\relax}
\makeatother
```

The macro `\member{⟨name⟩}` can be used several times to add ⟨*name*⟩ to the `\@members` macro. The macro `\themembers` can then be called to print the contents of `\@members` and clear `\@members` so a new list may be started. If you want to print the list more than once then use `\showmembers` which prints, but does not clear, the list.

```
\member{Fred} \member{Joe}
\member{Susan} \member{Faye}
```

`\themembers` ⇒ Fred, Joe, Susan, Faye

For more complex additions, for instance when the macro to be extended takes arguments, then the patchcmd package [4] could be the answer.

Once having created a list of members it might have to be changed because one or more members have left. This is more complicated and I present it only as an example of what could be done.

The `\deletemember{⟨name⟩}` will go over the list of members, creating a new temporary working list with the exception of the ⟨*name*⟩ member, then replace the original list with the working one.

```
\makeatletter
\let\xpf\expandafter%  just a shorthand
\newcommand*{\deletemember}[1]{%
  \let\@tempmembers\relax
  \def\@dm@num{1}%
  \@for\member@:=\@members\do{%
    \ifnum\@dm@num<2\relax
      \def\t@mp@b{#1}%          initial entry
      \ifx\member@\t@mp@b%
        \def\@dm@num{0}%
      \else
        \def\t@mp@b{\space #1}% later entries
        \ifx\member@\t@mp@b%
          \def\@dm@num{0}%
        \fi
      \fi
    \fi
    \ifnum\@dm@num=0\relax
      \def\@dm@num{2}%
    \else
      \xpf\xpf\xpf\transfer\xpf{\member@}%
    \fi}%
  \let\@members\@tempmembers}
\makeatother
```

The coding of `\deletemember` is not straightforward. The LaTeX kernel's `\@for` construct is used to loop over the comma-separated entries in `\@members`, putting, in turn, each entry into the `\member@` macro. Due to the way that `\@members` is constructed, the name of the initial entry is recovered as `name`, while a later entry is recovered as `␣name`; hence the two tests for the argument ⟨*name*⟩ against the recovered `\member@` name.

The `\@dm@num` macro is used to track the state of the process. At the start it is set to 1. If it is less than 2, attempts are made to match the argument with the current list name and if a match is found then `\@dm@num` is set to 0. After the argument check, if the argument is matched (`\@dm@num` = 0) then `\@dm@num` is reset to 2, otherwise the current member name is added to the working list. This all means that once the list name matches the argument then no further attempts at matching are needed or done, and the remaining original members are simply added to the working list. At the end the original list is set to the temporary working list.

The tricky part is that the current contents of `\member@`, not the macro itself, should be added to the working list.[1] The bunch of `\expandafter`s around the call to `\transfer` expands `\member@` to its definition before it gets handed over as the argument to `\transfer`.

The macro `\transfer{`⟨*name*⟩`}` adds ⟨*name*⟩ to the macro `\@tempmembers` containing a list of comma separated names. It has the same general form as the earlier `\member` macro.

```
\makeatletter
\newcommand*{\transfer}[1]{%
  \@ifundefined{@tempmembers}{%
    \newcommand*{\@tempmembers}{#1}%
  }{%
    \g@addto@macro{\@tempmembers}{,#1}%
  }}
\makeatother
```

Here are some examples of adding and deleting members to and from the original member list above.

```
\member{Alice} \member{Bob} \member{Claire}
\member{David} \member{Erica}
```

`\showmembers` ⇒ Fred, Joe, Susan, Faye, Alice, Bob, Claire, David, Erica
`\deletemember{David}\showmembers` ⇒ Fred, Joe, Susan, Faye, Alice, Bob, Claire, Erica
`\deletemember{Fred}\showmembers` ⇒ Joe, Susan, Faye, Alice, Bob, Claire, Erica

```
\member{Xerxes} \member{Zeno}
```

`\showmembers` ⇒ Joe, Susan, Faye, Alice, Bob, Claire, Erica, Xerxes, Zeno
`\deletemember{Miriam}\showmembers` ⇒ Joe, Susan, Faye, Alice, Bob, Claire, Erica, Xerxes, Zeno

1.2 Piecing a paragraph

Ron Aaron wanted a different kind of assembly. He wrote [1]:
What I wish to do is accumulate text into a paragraph 'as I go'. My simple approach is to allocate a box, and then unbox and add the text. But this doesn't work as I intend:

```
\newbox\textbox
\def\addbox#1{%
  \setbox\textbox\vbox{
    \unvbox\textbox#1}}
\addbox{Hello}
\addbox{there!}
\box\textbox
```

What I get is each appended bit of text in a separate line. I've tried to '`\unskip`*' and '*`\unkern`*' etc. after the* `\unvbox` *but whatever I do I get a list of lines ...*

Trying out Ron's example the result is:

Hello
there!

The squashed vertical spacing between the lines is real, not an artifact of this article.

In responding, Philip Taylor [5], having said that using a `\vbox` would be difficult, then gave two suggestions; either use an `\hbox` directly or a token-list register. His `\hbox` solution (and my example) is:

```
\newbox\textbox
\def\addbox #1{%
  \setbox \textbox = \hbox
  \bgroup
    \unhbox \textbox #1%
  \egroup}
\addbox{Hello}
\addbox{ World!}
\addbox{ Now isn't that a rather
         common saying?}
\unhbox\textbox
```

with the result:

Hello World! Now isn't that a rather common saying?

At various points after this I have used code like
`\addbox{ (n) text}`
as an example of assembling a paragraph piece by piece and at the end showing the result via:
`\unhbox\textbox`

```
\addbox{(1) Start of a paragraph.}
```

Philip's second solution uses a token register:

```
\newtoks\texttoks
\def\addtoks #1{%
  \texttoks =
  \expandafter {\the \texttoks #1}}
\addtoks {Goodbye}
\addtoks { \emph{vain} world. Ah, the
          weariness in that statement
          does one no good.}
\the \texttoks
```

and the example produces:

Goodbye *vain* world. Ah, the weariness in that statement does one no good.

```
\addbox{ (2) After an interruption
        add more.}
```

Note that with both of Philip's solutions you have to explicitly incorporate spaces where you want

[1] If the macro is added then the list will consist of nothing but a series of `\member@`, thus all expanding to the identical name (the current definition of `\member@` when the list is printed).

them to occur in the assembled paragraph. It seemed, though, that Ron really wanted to use a `\vbox` but I have neither seen nor been able to come up with satisfactory code.

```
\addbox{ (3) This is the end
        of the piecewise
        paragraph.}
\unhbox\textbox
```

Now print the piecewise paragraph giving:

(1) Start of a paragraph. (2) After an interruption add more. (3) This is the end of the piecewise paragraph.

Beneath those rugged elms, that yew-tree's shade,
Where heaves the turf in many a mouldering heap,
Each in his narrow cell for ever laid,
The rude forefathers of the hamlet sleep.

Elegy Written in a Country Churchyard, Thomas Gray

2 Table talk

Arbo [2] wanted a tabular layout like the one shown in Figure 1 and tried using code like this to produce it.

```
\begin{tabular}{r|c|l}
\hline
First & Second & Third \\
Text &
  \multicolumn{1}{c|c|c|c|}%
                {C 1 & C 2 & C 3 & C 4}
  & More text \\
Words &
  \multicolumn{1}{c|c|c}%
                {C 5 & C 6 & C 7}
  & Text \\
Title &
  \multicolumn{1}{c|c}%
             {C 8 & C 9}
  & Some text \\
\hline
\end{tabular}
```

If you try it you will find, like Arbo, that it doesn't work, resulting in a string of error messages beginning with:

First	Second	Third
Text	C 1 │ C 2 │ C 3 │ C 4	More text
Words	C 5 │ C 6 │ C 7	text
Title	C 8 │ C 9	Some text

Figure 1: Desired tabular layout

```
! Missing } inserted.
<inserted text>
                }
l.6945  {C 1 & C 2 & C 3 & C 4}
```

The problem is that `\multicolumn` merges multiple columns into one whereas the requirement here was to split one column into several.

Donald Arseneau [3] responded that *'They don't look aligned at all, so don't call them columns'*, and provided code for an `\addcell` macro. Arbo modified it very slightly to center the `\vlines`, with the final version as follows:

```
\newcommand{\addcell}{\unskip\hfill
  \hspace\tabcolsep\vline\hspace\tabcolsep
  \hfill % added by Arbo
  \ignorespaces}
```

Using this, the tabular in Figure 1 is created by:

```
\begin{tabular}{|r|c|l|}\hline
First & Second & Third \\
Text & C 1
       \addcell C 2 \addcell C 3 \addcell C 4
     & More text \\
Words & C 5
        \addcell C 6 \addcell C 7
      & text \\
Title & C 8 \addcell  C 9 & Some text \\
\hline
\end{tabular}
```

References

[1] Ron Aaron. How to append text to a paragraph (in an existing vbox)? Post to `xetex` mailing list, 16 July 2010.

[2] Arbo. How to produce multiple columns within a multicolumn. Post to `comp.text.tex` newsgroup, 2 November 2010.

[3] Donald Arseneau. Re: How to produce multiple columns within a multicolumn. Post to `comp.text.tex` newsgroup, 2 November 2010.

[4] Michael J. Downes. The patchcmd package, 2000. `http://ctan.org/pkg/patchcmd`.

[5] Philip Taylor. Re: How to append text to a paragraph (in an existing vbox)? Post to `xetex` mailing list, 16 July 2010.

◇ Peter Wilson
12 Sovereign Close
Kenilworth, CV8 1SQ
UK
`herries dot press (at)`
`earthlink dot net`

Randomising assignments with SageTeX

Sabri W. Al-Safi

Abstract

SageTeX allows for the processing of SageMath code embedded in a TeX document, and the automated generation of TeX code to display SageMath objects. This article reports on the use of SageTeX to generate individualised coursework assignments (with corresponding answers) for students in an undergraduate mathematics module.

1 Motivation

One of the modules I teach to third year undergraduate mathematics students involves an individual coursework assignment; this is perhaps the type of assessment that is most vulnerable to concerns about plagiarism. In previous years, the same assignment was given to every student, which did little to help things. Hence I was curious to know if there was a relatively easy way to randomise the assignment so that each student is given a unique paper, and their solutions could be marked according to a corresponding markscheme (a.k.a. list of answers).

Various packages exist which can be used to generate randomised problems. The e-assessment system Numbas [4] is a popular example in which authors can define randomised variables when displaying mathematical objects in a question. Students can refresh the question to re-randomise these variables at will, and an automated script can be used to mark answers according to the variables' values.

Numbas is well-suited to computer-based assessment, but there are obstacles to using it for the formal assessment of advanced mathematics. Firstly, I wanted to hand out a neatly typeset document for an examination or assignment. Whilst Numbas can handle TeX syntax (and can even be tweaked to create printable PDF worksheets and markschemes), its typesetting capabilities are limited, and I would not have as much fine control over the output as when using TeX on my own computer.

A second drawback to Numbas is the lack of a sophisticated computer algebra system. The package contains a custom algebra system which includes some basic methods, but does not possess the advanced capabilities of, say, Mathematica. I would be forced to use a degree of effort and cunning to test my students' ability to transform a matrix into Jordan Canonical Form.

2 SageTeX

The TeX package SageTeX [1, 2] overcomes these drawbacks by integrating SageMath into LaTeX documents. SageMath [5] is a powerful computer algebra system which is a free, open source alternative to Maple, Mathematica, Matlab or Magma. SageTeX allows a user to embed SageMath code into a TeX document, have that code executed, and automatically generate TeX code based on the results of those computations.

TUGboat has previously featured a comprehensive review of SageTeX [3], so I will cover only the salient aspects. I'll assume that both LaTeX and SageMath are installed on the system (it's possible to run SageMath remotely, but we won't get into that here). The following line must be added to the preamble:

```
\usepackage{sagetex}
```

The SageMath code is usually written into either a `sageblock` or a `sagesilent` environment in the TeX document. When LaTeX is run, all text within these environments is copied to a single, separate `.sage` file for execution (`sageblock` additionally typesets the code verbatim into LaTeX's output). The code within these environments is collectively considered as a single SageMath program. As an example, if I want to assign to the variable A a diagonalizable 3×3 matrix with randomised integer entries and integer eigenvalues, I can write, anywhere in the TeX document (broken across lines here for *TUGboat*):

```
\begin{sagesilent}
import sage.matrix
A = random_diagonalizable_matrix(
    MatrixSpace(ZZ, 3))
\end{sagesilent}
```

In SageMath, each object has associated TeX code which renders an accurate representation of that object in math mode. One can automatically generate this code via the `\sage` macro. This allows the results of calculations to be displayed without having to do those calculations yourself, which is especially useful if your objects have been randomly generated. Thus, continuing, if I want to display a list of the eigenvalues of our matrix A later in the same document, I can write:

```
The eigenvalues of
$\sage{A}$ are $\sage{A.eigenvalues()}$.
```

The `\sageplot` command instructs Sage to generate graphical plots which are then included in the document. This circumvents some of the need to fuss with graphics files or `\includegraphics` commands (since SageTeX does it for you). Again, this can also be useful when applied to functions that depend on randomly generated variables.

```
\begin{sagesilent}
r = ZZ.random_element(10)
\end{sagesilent}
\sageplot{plot(sin(r*x), x, 0, 2*pi)} .
```

Getting SageMath to evaluate the SageMath code, produce plots and generate TeX code requires an additional couple of steps when compiling the document. When LaTeX is first run on the `.tex` file, an auxiliary file with the extension `.sagetex.sage` is generated. The user must then run SageMath on this file, before running LaTeX once again on the original `.tex` file. This series of steps can become tedious, although it is straightforward to automate (as I describe in the next section) and is only necessary when the SageMath commands are changed.

3 Method for individualising assignments

The assignment and the markscheme were written as separate TeX documents (`Questions.tex` and `Answers.tex`) to avoid having to split several PDF files into two. They were individualised by heavy use of randomly generated SageMath objects, so that although the students were required to use the same mathematical techniques, they would be applying them to different numbers, matrices and functions.

I needed the markscheme to use the same randomised objects as the assignment, therefore I needed control over the random state of the SageMath program. This was achieved by setting the same random seed at the beginning of `Questions.tex` and `Answers.tex`, which was read from a text file named `random_seed.txt`:

```
\begin{sagesilent}
set_random_seed(
  int( open('random_seed.txt').read() ) )
\end{sagesilent}
```

In addition, any creation of a random object in `Questions.tex` had to be mirrored by the exact same operation in `Answers.tex`. I was otherwise free to use SageMath to compute and display anything I wanted in the markscheme. This even included plotting functions, against which the student's own plots could be checked. The random seed approach had the added bonus that in order to re-generate the assignment or markscheme for a given student at any time, one could simply write the appropriate number into `random_seed.txt`.

The random seeds were chosen to correspond to the students' university numbers so that they were unique to each student and easy to recover. It was straightforward to export the relevant list of student numbers from the university's online learning environment, strip out all alphabetic characters, and save it to the file `student_numbers.csv`. The subsequent 6-digit numbers were used in turn to set the random seed at the beginning of both `Questions.tex` and `Answers.tex`. For both of these files the "TeX–SageMath–TeX" compilation procedure was carried out as outlined in the previous section, and the output PDF was renamed according to the student number. This whole process was entirely automated by the following Python script, named (for myself) `compile.py`:

```
import subprocess
import os

def CompileSageTex(fileName):
 subprocess.call(['pdflatex', fileName+'.tex'])
 subprocess.call(['sage',
                 fileName+'.sagetex.sage'])
 subprocess.call(['pdflatex', fileName+'.tex'])

for seed in open('student_numbers.csv')
             .read().split(','):
 open('random_seed.txt', 'w+').write(seed+'\n')
 CompileSageTex('Questions')
 CompileSageTex('Answers')
 os.rename('Questions.pdf',
          'Assignment_'+seed+'.pdf')
 os.rename('Answers.pdf',
          'Markscheme_'+seed+'.pdf')
```

The assignments were then ready to be sent to the students using a mail merge.

4 Observations

One of the drawbacks to SageMath and SageTeX is the initial difficulty of installation. SageMath is most at home in a Unix environment; using it on a Windows system requires a virtual machine, which makes SageTeX a much less viable option. It is possible to run SageMath remotely, but it may not be very easy to do this in an automated fashion as presented here. However, once I had configured SageTeX on my system, I found it intuitive and powerful.

SageTeX adds a significant amount of time to the compilation process. On my laptop running GNU/Linux, a test run of `compile.py` using a string of ten student numbers took a little over two minutes to produce assignment and markscheme documents with sizes roughly 123kb and 152kb respectively. If the CompileSageTex function was stripped down to a single call to `pdflatex`, it took about five seconds.

5 Discussion

SageMath is not the only language that can be integrated with TEX. Haskell and R are two notable examples for which similar tools exist to mix their code with TEX code, and generate TEX code on the fly. I used SageTEX primarily because:

- I already had some familiarity with Python and SageMath;
- I knew that SageMath was adept at linear algebra and symbolic computation;
- I could instantly see how to use SageTEX to achieve my goals.

I haven't tried the other alternatives myself, and a cursory search does not reveal any rigorous comparison of the available tools. Before deciding on one or another, consideration should probably be given to the differing capabilities of, and to one's prior familiarity with, each language.

My use of text files and a Python script is unlikely to be the fastest or most efficient method, although it served my immediate purposes. If the size of the assignment and the number of students is large, then disk space may be an issue. In this case, it may be desirable to print or email to each student the assignment immediately after compilation, and then delete the associated files.

Any use of SageMath aids its proliferation as a free alternative to proprietary software. The lead developer, William Stein, has written about his considerable efforts to improve and sustain its user base, especially for undergraduate teaching [6, 7]. If the method described in this article were somehow to be made simpler and faster, I believe teachers would find it very appealing as a way of cutting down on plagiarism, generating original problem sets, and maintaining strict correctness between questions and solutions.

References

[1] Drake, Dan, et al. "The SageTEX Package" (2010). `http://w.astro.berkeley.edu/~domagalski/latex/sagetexpackage.pdf`

[2] Drake, Dan, et al. SageTEX on CTAN. `http://ctan.org/pkg/sagetex`

[3] Joshi, Manjusha. A dream of computing and LATEXing together: A reality with SageTEX. `http://tug.org/TUGboat/tb32-3/tb102joshi.pdf`

[4] Numbas. `https://numbas.mathcentre.ac.uk`

[5] SageMath. `http://www.sagemath.org`

[6] Stein, William. "What is SageMath's strategy?" `http://sagemath.blogspot.co.uk/2015/09/what-is-sagemaths-strategy.html`

[7] Stein, William. "You don't really think that Sage has failed, do you?" `http://sagemath.blogspot.co.uk/2014/08/you-dont-really-think-that-sage-has.html`

◇ Sabri W. Al-Safi
School of Science & Technology,
Nottingham Trent University,
Burton Street, Nottingham,
NG1 4BU, UK
`Sabri dot Alsafi (at) ntu dot ac dot uk`

Indexing: Goals, strategies and tactics

Ronald J. Fehd

Abstract

In this article I review my index production notes from a previous project and make a checklist of goals, strategies and tactics for my next book. I compare the writing roles of author, editor and proofreader with the indexing team of author and indexer.

The purpose of this article is twofold: the first is to provide a workflow for documents with indexes that makes debugging, proofreading and polishing of index entries across multiple files easier. The second is to provide a template document that supports that workflow. The packages `chappg`, `fancyverb`, and `indextools` are used in the template.

The expected audience of this article is authors and technical writers who want to become proficient in the art and science of indexing.

1 Introduction

What is an index? A simple description is *an index assists the reader in finding information in a document.* The term *index* has two components, *what* and *where.* The *what* is the *entry*; the *where* is the *locator*, usually a page number. An index is an asset to the reader. Its presence is a selling point for the document. Consider the reader before sale, as a customer, and after sale, as a user.

A knowledgeable reader picks up a book and wants to access the quality of the index. A coarse metric of quality is quantity, the simple easy-to-calculate ratio of the number of pages of the index divided by the number of pages of the book. The range of this fraction, as a percentage, is 2–15%, where 2% indicates a trivial index and 10+% shows a concerted effort, as for a reference book or technical manual, that will be both useful and often used.

Mulvany [15, table 3.2, pg. 72], shows the correlation between this percentage and the professional indexer's estimated work effort; this metric is known as *entries per page* (E/pg) discussed below in the section 'Budget and quality'.

Team work In writing there are three people on the team: author, editor and proofreader. The editor's role has two tasks: micro, concerned with producing accurate and consistent syntax; and macro, concerned with focus, winnowing and elimination of redundancy. The proofreader's role is to detect inconsistency and do fact checking.

In indexing, there are apparently two people on the team: the author and the indexer. In project management we have the three choices of *fast, accurate, or cheap.* Professional indexers are paid to be the pair of choices of goodness: fast and accurate. Their job description encompasses both of the writers' team roles of editor and proofreader for the index.

The task of this article is to examine the work of editing and proofreading an index and present a workflow for writers which makes those tasks obvious, defined and easy to accomplish. Additional intermediate documents are needed for the index review; an example template which supports the review process is shown on page 35.

A professional definition Mulvany [15, pg. 8], provides this definition.

> An index is a structured sequence — resulting from a thorough and complete analysis of text — of synthesized access points to all the information contained in the text. The structured arrangement of the index enables users to locate information efficiently.

Concepts In mathematics we have the concept of an *array* which contains a set of values; an index is an integer, a pointer to an element in the array. The concepts of array and index are separate from the values in the array; the concepts are a method of accessing the values in an iterative loop.

In natural languages we can apply the concept of an *associative array.* An abbreviation or acronym is used as shorthand to refer to a longer word or concept; e.g., two-letter abbreviations for country names used in urls and two-letter state abbreviations used by the United States Postal Service.

In *database* theory, we have two types of tables, fact tables and dimension tables. A fact table is a record of a transaction, and contains two types of columns: foreign keys which are pointers to rows in dimension tables, and facts which are the measurements or quantities of the transaction. A row may be uniquely identified by a composite key, a set of the foreign keys. A dimension table contains two types of columns: a primary key (row identifier) which matches the values of the fact tables' foreign keys and columns of information about the key.

In the associative array examples, given an abbreviation, the primary key (country code, state abbreviation, etc.), we can retrieve the information about the entity — capital city, location, etc.

Index, my definition I propose that a document is a *knowledge base*, that the index is a database and that the words of an index entry are an item of an associative array, a composite key, of and for that document, that points to the information that the reader is seeking. The *locator* is the unique *row* *identifier*, the page number, that the reader uses to access the information in the document. Just as the table of contents is a sequential exposition of the ideas in the document, the index is an alphabetical listing of the concepts, ideas, and tasks in the document. The index, in that sense, is a further extension and expansion of the table of contents.

Implications of this idea are discussed below.

1.1 How professionals work

For professional indexers the project of producing an index for a document consists of the tasks shown in Table 1.

Table 1: The professional indexer's process

```
0. read the complete document
1. identify, record ranges
2. for each page:
     identify terms or tasks
     recording: insert new or update old
3. polishing the database:
     add redirects
     contract single item!sub-item
       to item, sub-item
     expand items with too many references
       by adding sub-items
     contract series into ranges
4. transform database to output
```

Professional software offers support for the polishing processes during the item recording process, by saving the entries in a database so that each new entry is compared to its predecessors and a decision must be made to use an existing entry and add a locator, or create a new entry.

Compare this to the markup process shown in Table 2 where entries are written to an `.idx` file so we have a sequential list of entries, which are then sorted, duplicates removed, and formatted in one step.

1.2 Review of indexing markup commands

There are three types of indexing commands in LaTeX: *words(s)*, *ranges* and *redirects*.

■ **word(s)** Words may be marked in three styles, *plain*, *special font*, or *special locator*.

plain: The indexer marks up plain entries in a document with this form:

```
\index{word(s)}
```

The `word(s)` can be any string. In support of later polishing, `word(s)` can be subdivided into as many as three levels with `!` characters:

```
\index{item!sub-item}
\index{item!sub-item!sub-sub-item}
```

special font: For the case where a word is to be set in another font the syntax is:

```
\index{item@\texttt{item}}
```

special locator: Special pages may be marked so that the *locator* appears in a different font by adding a vertical bar (`|`) plus the font shape (italic) or series (bold).

```
\index{item|textbf}
```

It is customary to mark a page containing a definition in bold. Pages in a glossary may be marked in italic. Any such special locator emphasis ought to be explained in a prologue to the index. For an example of an index prologue see *Guide to LaTeX* [12]. The `indextools` package [17] contains the command `\indexprologue` for this purpose.

■ **ranges** To mark up a page range use a pair of commands, the first with `|(` and the second with `|)`:

```
\index{autoexec|(}%range begin
...
\index{autoexec|)}%range end
```

No markup is necessary for the consolidation of sequential locators. The `makeindex` processor consolidates sequential page numbers, e.g., `2, 3, 4` is changed to `2--4`.

■ **redirects** A locator can be in one of three forms, either a single page number, or a page range such as 8–13, or a *redirect*, a reference to another index entry, without a page number. The syntax is

```
\index{canines|see{dogs}}
```

This produces an index entry without a page number, as in:

canines, *see* dogs

1.3 The `makeindex` process

Producing an index of a document using LaTeX and `makeindex` consists of the steps shown in Table 2. I highlight two intermediate steps in this process, sorting and removing duplicates, both of which are done in memory by `makeindex`. They can be split out into the creation of the two temporary files, *sorted.idx*, and *nodups.idx*. See Listing 2 for the system commands to produce the *sorted.idx* file.

Listing 1 shows a demonstration file. Processing the document produces output file `\jobname.idx` with the *locator* information (page numbers) added:

```
\indexentry{fox}{1}
\indexentry{dog}{1}
```

Processing with `makeindex` produces an output file named `\jobname.ind`, the formatted output:

Table 2: The `makeindex` process

Processor	Input, Action	Output
(pdf)latex	jobname.tex	jobname.idx
makeindex	jobname.idx	
	sort:	*sorted.idx**
	remove dups:	*nodups.idx**
	nodups.idx	jobname.ind
(pdf)latex	jobname.tex	jobname.pdf

* temporary file

Listing 1: Document with simple indexing

```
1  %this is the file demo-1-makeindex.tex
2  \documentclass{article}
3  \title{Example of Simple Indexing}
4  \author{R.J. Fehd}
5  \usepackage{makeidx}\makeindex
6  \begin{document}\maketitle
7  The quick brown fox\index{fox} %*
8  jumped over the lazy dog\index{dog}.
9  \printindex
10 \end{document}
```

* The index markup is placed after the word as recommended by the package documentation.

```
\begin{theindex}
  \item dog, 1
  \item fox, 1
\end{theindex}
```

The `\printindex` command (line 9) reads that `.ind` file if it exists:

```
\IfFileExists{\jobname.ind}
  {\input{\jobname.ind}}
  {}
```

2 Discussion of my experience

I will use a project of my own as an example. I worked on writing a book over a period of about a year. There were six chapters of 15–30 pages each. I spent about a month writing a chapter. Since I knew I would produce the index myself, I added the markup as I wrote.

In early chapters I used a form for noun in category `{noun category}` that I later changed to `{category!noun}`, e.g., from `{scan function}` to `{functions!scan}`.

In the middle chapters, I was echoing these entries, marking both

```
{noun category}
{category!noun}
```

In the last chapters I decided that I wanted `{category!noun}`, so I marked `{noun category}` as a redirect:

```
{noun category|see{category, noun}}
```

in those chapters. Since each redirect has a separate *locator* the duplicate removal process leaves multiple entries, only one of which is to be retained. At the end I changed all the `{noun category}` entries to *redirects* and placed them in a separate file.

2.1 Lessons learned from my mistakes

In this section I provide a summary of the categories of mistakes that I made.

These are the categories: *macros*, *multitasking*, *ranges*, *redirects*, *repair*, *testing*, and *vocabulary*.

macros I used macros for a while but gave up on them for two reasons. The first was that macros introduced extra spaces, which I was not a sophisticated enough TEXnician to fix; and second, complexity. Because I was marking up item, sub-item and sub-sub-item, with font changes, I realized that managing the set of macros might consume as much time as plain long-form markup. Swapping out the macros with long-form markup contributed to the rework budget.

multitasking Writing, editing and polishing are very different intellectual activities from indexing. Adding markup while I was polishing produced different entries e.g., *directories* and *folders*. Identifying synonyms, deciding which one to use, and then finding and fixing the other occurrences contributed to the rework budget.

ranges One of my most common problems in getting the index working for a chapter was incomplete ranges; these errors are noted in the `\jobname.ilg` file. Ranges may cross one or more sections and the pair of markup commands may be many pages apart.

redirects The markup syntax for *redirects* is the same as for *word(s)*. What is different is that the *locator* in the printed result is not a page number, but a reference to another word. My mistake was in thinking that the `makeindex` process removed duplicate *redirects*. Moving these entries to a central file contributed to the rework budget.

repair where? All the items listed here required that I return to one of the many chapter files in order to fix the problem entries. The index entry repair problem is discovering in which file the entry is located. The problem is that the *locator* is the page number of the complete document. Finding the name of the chapter file is a two-step process, first, note the page number, then return to the table of contents to find the page range of the chapter.

testing During the retrospective for this paper I realized that I spent a lot of time reviewing the complete index. My solution for future work will be to do unit tests for each chapter and one final complete integration review.

vocabulary As noted above in the multitasking paragraph, choice of vocabulary is a key task to accomplish before beginning indexing. Indexing science addresses this idea with the concept of *controlled vocabulary*. Much to my surprise, words in the entries did not occur as phrases in the text.

Summary In this section I have described a number of mistakes. Time spent on this work can be reduced by discovering ways to implement these tasks. In short:

- change page number to include chapter number
- control vocabulary used in entries
- centralize *redirects*
- manage ranges early
- process each chapter independently, for both writing and indexing
- review temporary files: `\jobname.idx`, `\jobname.ilg`, `sorted.idx`
- polish the complete index only once

3 Budget and quality

Budget and daily job diary Table 3 is from Frederick Brooks' classic book, *The Mythical Man-Month* [3]. This table highlights the under-estimated items in the budget: unit and integration testing. I show this table in my presentations and point out the difference in the time spent coding, where much time is spent debugging — getting the code to work — and the two aspects of testing: testing the program against its expectations — unit testing — and how a program works as a caller or callee: integration testing.

Table 3: Time spent in program development (from [3])

Phase	Time	Action	Time
design	1/2	understand problem: education, research	1/3
		development coding	1/6
testing	1/2	unit test	1/4
		integration test	1/4

For the task of indexing I want to carry forward the ideas of planning and markup, typing the entries, unit test of entries in a chapter, and the integration test of all the chapters of a document. Editing and proofreading of index entries in a chapter are separate tasks from proofreading the index of the whole document.

Table 4: Estimates of percentage of index pages and entries per page (E/pg), by book type*

	index size as	
Book type	% of book	E/pg
light text, few details	2–5	3–5
reference	7–8	6–8
documentation, light	10	8–10
manuals, heavy	15+	10+

* adapted from [15, table 3.2]

The daily job diary (djd) is a necessary part of budgeting. If we keep track of time spent on a task then we can monitor and estimate whether we will be under or over budget.

Metrics for quality As explained in the introduction, an interested purchaser can calculate the percentage of pages of the index in a book. This is a rough estimate of the depth of the index: how much information is in the index and how easy is it to use the index to find information as compared to reading the table of contents.

There are two metrics of index quality: entries per page (E/pg) and locators per entry (L/E).

Entries per page (E/pg) This quantity is used as a forecasting aid; it is an indexing science term mentioned in [15] and is derived from the type of document. It is used to estimate the size of the index, the number of pages set aside in the document, as a percentage of the number of pages of text. In the section in which this table appears, studies are quoted as discovering that the average size of an index is three percent. Table 4 is my adaptation of the information.

Locators per Entry (L/E) This quantity is an integer, typically in the range 3–13, decided on before polishing the final draft of the index to subdivide entries and therefore increase the size of the index.

4 About goals, strategies and tactics

We have two goals. The first is to prepare an index for a reader with sufficient detail and depth for the type of document. The second is to be aware of the budget and enhance the production process by reducing time spent on indexing tasks.

We'll discuss some packages, other tools, and tactics to meet these goals.

4.1 Strategies: packages

Several of the problems noted earlier have solutions that are outside the set of indexing tasks. Three packages can help reduce the time spent in administration of the document (all have many features besides what's mentioned here):

`chappg` changes *locator* output formatting from *page* to *chapter*.*page*.

`fancyvrb` provides `\VerbatimInput{file.ext}` to typeset external files verbatim.

`indextools` provides `\makeindex[intoc]` to put an index in the table of contents (instead of using `\addtocontents`), and `\indexprologue`, to typeset introductory text before the index, e.g., to describe the formatting conventions used.

More information about these packages is available through CTAN, at `http://ctan.org/pkg/`*pkgname*. They are all included in the usual TEX distributions.

4.2 Strategies: tools

This section discusses methods for using (LA)TEX commands that can help in index (and document) development.

▪ **flags** A flag is a variable which takes only boolean values, true or false. These are the TEX commands to initialize a new boolean switch named `ReviewIndex`:

```
\newif\ifReviewIndex
  \ReviewIndexfalse % set to false
  \ReviewIndextrue  % set to true
```

Then, this is the template for conditionally executing statements with this flag:

```
\ifReviewIndex
  % statements if true
\else
  % statements if false
\fi %end:ReviewIndex
```

The `\else` part is optional.

As seen in the template document of Listing 3 (in the appendix), I use this flag to control formatting and display of various intermediate indexing files for review.

This set of statements may be implemented in LATEX with the `ifthen` package [12, pg. 440]. I use the plain TEX statements because I use it to choose the `\documentclass` options.

▪ `\IfFileExists` This is a LATEX programming command [12, pg. 436]. This command can be used to ensure that the document compiles before temporary files are created.

```
\IfFileExists{filename.ext}
  {statements if true}
  {statements if false}
```

▪ `\includeonly` This is a LATEX command used to process only one subpart of a document, such as a chapter. Its use requires two statements, the first in the preamble:

```
\includeonly{ch-A}%disable after unit test
```

and then in the body [12, pg. 206]:

```
% in text part, list all chapters
\include{ch-A}
...
\include{ch-Z}
```

4.3 Tactics during markup

Here are some minor habits I developed to help produce a cleaner document.

- Mark up ranges after outlining in a file, e.g., `ch-0-outline-index`
- One index command per line better supports later searching.
- End entries with percent sign (`%`) to avoid multiple spaces between words.
- Place all redirects (*see* and *seealso*) in one file to be input after `\begin{document}`, as in: `\input{ch-0-index-redirects}`

5 Suggestions for a workflow

Table 5 lists the major steps in index creation.

Table 5: Overview of workflow for indexing

Step	Idea	Task
1	setup	make files
2	for chapters	writing
3	for chapters	indexing, unit test
4	review index	integration test
5	finish	

We'll now discuss each of these in turn.

▪ **setup** The first task is setting up files for the project and getting the document working with the various chapter and other files that support review. (The listings are in the appendix to this article.)

File	Task	Listing
`a-book.tex`	copy	3
`ch-0-outline-index.tex`	outline	5
	draft index,	
	mark ranges	
`ch-0-index-redirects.tex`	redirects	4
`a-book.bat`	doc working	2

The most important task in the setup, right after outlining the text, is to begin to draft the index. Make an entry for every task mentioned in the section commands. This is the first draft of the database — `sorted.idx` — that we will use as a reference when marking up each chapter.

▪ **writing a chapter** Create a new chapter file and copy the draft outline and index entries from file `ch-0-outline-index.tex` into it. Record amounts of time worked in a daily job diary.

File	Task
`a-book.tex`	`\includeonly{ch-X}`
	`\include{ch-X}`
`ch-X.tex`	write, type, edit, etc.

▪ **indexing a chapter** The task of indexing requires a change of intellectual attitude. Remove your author's hat and put on your proofreader's hat; change mode from exposition to reader and reviewer; become objective. Part of indexing is comparing the draft index of the complete document to what has been written in the chapter under review. Each entry ought to be either *new* or *already existing* in the draft database, `sorted.idx`. This comparison is a step in the quality assurance of the index. Remember to record time-start and -end in the daily job diary.

File	Task
`a-book.tex`	disable this chapter:
	`%\includeonly{ch-X}`
`a-book.bat`	process whole document
`a-book.pdf`	print only `sorted.idx`
`a-book.tex`	`\includeonly{ch-X}`
`a-book.pdf`	print galley of ch-X
`ch-X.tex`	for each page:
	move draft entries to correct line
	compare markup to `sorted.idx`
	type index entry
`a-book.bat`	process chapter
`a-book.ilg`	confirm unit test correct
	add new entries to draft file
`a-book.tex`	`%\includeonly{ch-X}`

Refer to the `showidx` listing at the top of each page; count and enter the number of index entries per page in the Entries-per-Page (E/pg) log. Calculate the mean and standard deviation of E/pg. Examine pages with E/pg more than 2.5 standard deviations from mean. Review the daily job diary; compare time spent writing with time spent indexing.

▪ **reviewing index** Choose the maximum locators per entry (L/E), e.g., from the Fibonacci series: (3, 5, 8, 13). This choice indicates the depth of the index. A large L/E indicates a shallow or small index. A reader will have to look up that many entries in the text to find what they want. Entries with more than that are to be subdivided.

File	Task
`a-book.tex`	disable all:
	`%\includeonly{ch-X}`
`a-book.bat`	process whole document
`a-book.pdf`	print `sorted.idx`
	find entries with large L/E:
`ch-?.tex`	expand entries
	find entries with single sub-items:
`ch-?.tex`	contract entries
`a-book.bat`	process whole document
`a-book.ilg`	confirm integration test complete
`a-book.pdf`	print index

▪ **finish** With all unit tests of chapters completed and the integration test of the document complete, print the final document and review the budget.

File	Task
`a-book.tex`	disable flag:
	`%\ReviewIndextrue`
`a-book.bat`	process whole document
`a-book.pdf`	print

Review the daily job diary; compare time spent reviewing the complete index (integration test) with sum of chapter times (unit tests). Compare time spent on indexing with time spent writing.

6 Suggested reading

This section contains *inspiration*, *manuals*, *budgeting*, *macros*, *standards*, and *types of indexing*.

inspiration In 2001, Robert Horn received a Lifetime Achievement Award from the Association of Computing Machinery. His acceptance speech is titled *What Kinds of Writing Have a Future?* His discipline of presenting information is called *structured writing*. This article and [5] are examples of the author's work using structured writing.

manuals After a dictionary and thesaurus, an author perhaps most needs a style guide. For American English writers, the essential guide is *The Chicago Manual of Style* [7]. The University of Chicago Press also publishes the bible of professional indexers, *Indexing Books* [15]. Chapter 9, *Editing the Index*, is the basis for my recommendation of polishing the complete index — the integration test — as a separate activity.

As a technical writer I found Ament's *Indexing Guide* [1] to be just the reference book that I needed. While he does not use the term *task* his explanation of deconstructing index entries into verbs and nouns made sense to me. His basic discipline can be expressed as follows:

A task consists of a verb and a noun. Always mark verbs in gerund form and nouns as plurals,

i.e.: `\index{<verb>ing <noun>s}`
e.g.: `\index{copying files}`

He states that the task of indexing can be used as feedback to improve the quality of the writing, because text that is poorly written is difficult to easily and accurately index.

This book is small (97 pages) and concise, written in plain language (`centerforplainlanguage.org`). I recommend it.

The Indexing Companion [4] provides an overview of the ideas and sources of *controlled vocabulary*. That discussion led me to the concept of an index as a database of the document, the knowledge base.

The database concepts I used in the explanation of my definition of an index are from [8].

The American Society of Indexers publishes *Best Practices for Indexing* [13]; it has appendices which provide guidance for authors in various genres.

budgeting *How to Communicate Technical Information* [16] describes the issues of budgeting in large projects of software and hardware documentation.

macros Gregorio, in [6], provides guidance for finding extra spaces when writing macros.

standards *The Global English Style Guide* [11] lays out a discipline of writing standardized English in technical manuals, which supports machine translation.

types of indexing This article is a discussion of the issues of *closed-system indexing*, i.e., finding information in one document. *The Organization of Information* [18] is an exposition of the theory and practice of information science which is concerned with *open-system indexing*, the issues of retrieval from multiple documents.

closing quotation *The LaTeX Companion* [14] ends the discussion of index markup with this quote:

> While all of these tools help to get the correct page numbers in the index, the real difficulty persists: choosing useful index entries for your readers. This problem you still have to solve. . . . to produce a comprehensive index that helps you, the reader, find not only the names of things . . . but also the tasks, concepts, and ideas described in the book.

7 Future work

I see several ideas that can reduce or eliminate time spent on indexing tasks. These ideas are *external input*, *metrics* and *redirects*.

external input For the case when both the author and a professional indexer are employed to mark up the entries we need a description of a file format, such as `.csv`, for use by packages which write `*.idx` files. Vendors of professional indexing software must be able to supply this file format to the TeXnician.

metrics I have identified the metrics entries per page (E/pg) and locators per entry (L/E) as summaries that may be reviewed for quality assurance. The data for these sums are in file `\jobname.idx`. **E/pg:** This data can be seen when using the package `showidx`, but the count must be manually tallied and recorded. **L/E:** This data can be seen in the file `sorted.idx`, but the count must be manually tallied and recorded.

redirects Add an index command for redirects which uses the same syntax as `\index` but which writes its output with the *locator* set to one. This allows multiple redirects in many files, which eliminates the task of moving them to a central location.

8 Conclusion

Marking up index entries is simple. The hard part of index preparation is polishing, which is perhaps 80% of the effort in preparing the index. In this paper I have presented several ideas to make the polishing effort less complicated. The first is sorting and saving the entries in the `.idx` file. The second is adding a flag `ReviewIndex` to turn on additional features. Another is to change the definition of `\thepage` so that it contains more information — the chapter number — in the `sorted.idx` file so that problem entries can be more easily located in their respective files. The last is to use the `ReviewIndex` flag to display various index files during review. I believe this set of packages, tools and workflow can help significantly reduce the time spent polishing the index in your next project.

9 Appendix

We've seen Listing 1 (`demo-1-makeindex.tex`, a minimal document with indexing) earlier (pg. 30). This appendix contains the remaining program listings.

It is left as an exercise for the reader to create the file `ch-A.tex` in order to get the template `a-book.tex` (Listing 3) working.

Listing 2: `a-book.bat`, Windows batch file for processing a document

```
rem this is a Windows DOS batch file
rem this is file  a-book.bat
set      jobname=a-book
pdflatex %jobname%
sort     %jobname%.idx /o sorted.idx
pdflatex %jobname%
```

Listing 3: `a-book.tex`, template document with several index review features

```
% this is template document a-book.tex
\newif\ifReviewIndex\ReviewIndexfalse
  \ReviewIndextrue %disable for final
\ifReviewIndex
     \documentclass[12pt,oneside]{book}
\else\documentclass[10pt,twoside]{book}
\fi %end:ReviewIndex
%
%\includeonly{ch-A}%unit test
%
\title{Example of Complex Indexing with
  ChapterPage, IndexTools and ShowIdx}
\author{R.J. Fehd}
\date{2016-March}
\ifReviewIndex
  \usepackage{showidx}%in right margin
  \setlength\oddsidemargin{0pt}%oneside
  \usepackage{chappg}
\fi
\usepackage{fancyvrb}  %VerbatimInput
\usepackage{indextools}%after showidx
\makeindex[intoc]      %write \jobname.idx
%\usepackage{hyperref}
%\usepackage{glossaries}%after hyperref
%\input{glossary-entries}
%
\begin{document}
%\frontmatter         %pg: i--x
%\input{c-frontmatter}%title, toc, etc.
%\mainmatter          %Chapter 1--N, pg 1--N
\ifReviewIndex \input{ch-0-outline-index}\fi
\input{ch-0-index-redirects}
\include{ch-A}%{ch-B}...{ch-Z}
%\input{c-backmatter}%bib, gloss, etc.
\ifReviewIndex
  \newcommand\Echo[1]{%arg=filename.ext
    \chapter         {#1}
    \IfFileExists    {#1}
      {\VerbatimInput{#1}}  %fancyvrb
      {file missing:  #1 } }%end:Echo
  \Echo{\jobname.ilg} %index log
  \Echo{ch-0-index-redirects.tex}
  \Echo{sorted.idx}
\fi %end:ReviewIndex
\indexprologue %an indextools command
  {This text explains font conventions.}
\printindex
\end{document}
```

Listing 4: `ch-0-index-redirects.tex`, template for index redirects

```
% this is file ch-0-index-redirects.tex
\index{canines|see{dogs}}
\endinput
```

Listing 5: `ch-0-outline-index.tex`, template chapter file

```
% this is file ch-0-outline-index.tex
\chapter{Draft: Outline and Index}

\section{chapter: Canines}canine
\index{dogs|textbf}%definition in bold
\index{foxes|textbf}%definition in bold

\section{chapter: Dogs}dog
\index{dogs|(}%range begin
\index{dogs!breeding}%
\index{breeding dogs}%
\index{dogs|)}%range end

\section{chapter: Foxes}fox
\index{foxes!habitat}%
\index{foxes!hunting}%
\endinput
```

About the author R. J. Fehd is a member of the TeX Users Group, and has been writing programs in SAS®[1], a language for statistical analysis, since 1986. He first published in 1997 and began using LaTeX in 2004; in 2006 he hacked other conference classes to produce `sugconf`. After retiring in 2012, he began work on a compilation of his papers; he continues to present papers and seminars at SAS user group conferences and is Senior Statistical Programmer at Stakana Analytics. He has been recognized as a peer for his contributions to the *SAS-L listserv*, and is a contributor to the *sasCommunity.org wikipedia*.

References

[1] Kurt Ament. *Indexing, A Nuts-and-Bolts Guide for Technical Writers.* William Andrew Publishing, Norwich, NY, June 2001. 97 pp., 3 chap., index: 11 pp. (11%). `https://www.elsevier.com/books/indexing/ament/978-0-8155-1481-7`.

[2] David Bausum. *TeX Reference Manual.* Springer, 2002. 388 pp., 2 chap., 3 app., index: 6 pp. (1%).

[3] Frederick P. Brooks Jr. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition.* Addison-Wesley, 2nd edition, 1995. 322 pp., 19 chap., index: 14 pp. (4%); theory and story based on development of the IBM System/360 operating system. `https://en.wikipedia.org/wiki/The_Mythical_Man-Month`.

[1] SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. In the USA and other countries ® indicates USA registration.

[4] Glenda Browne and Jon Jeremey. *The Indexing Companion*. Cambridge University Press, New York, NY, April 2007. 249 pp., 9 chap., index: 23 pp. (9%). `http://www.cambridge.org/9780521689885`.

[5] Ronald J. Fehd. An autoexec companion, allocating location names during startup. In *MidWest SAS Users Group Annual Conference Proceedings*, 2015. `http://www.lexjansen.com/mwsug/2015/BB/MWSUG-2015-BB-10.pdf`.

[6] Enrico Gregorio. Recollections of a spurious space catcher. *TUGboat*, 36(2):149–161, 2015. `http://tug.org/TUGboat/tb36-2/tb113gregorio.pdf`.

[7] John Grossman, editor. *The Chicago Manual of Style*. University of Chicago Press, 14th edition, 1993 (15th ed., 2006). 921 pp., 19 chap., gloss., bib., index: 25 pp. (6%); locators are chap.section, not page numbers. `http://press.uchicago.edu/ucp/books/book/chicago/I/bo3625262.html`.

[8] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit, The Complete Guide to Dimensional Modeling, Second Edition*. John Wiley & Sons, Inc., New York, 2002. 387 pp., 17 chap., gloss.: 29 pp., index: 18 pp. (4%). `http://www.kimballgroup.com/html/booksDWT2.html`.

[9] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984. 483 pp., 17 chap., 8 app., index: 25 pp. (5%).

[10] Donald E. Knuth. *Literate Programming*. CSLI, Stanford, CA, 1992. 368 pp., 12 chap., index: 10 pp. (2%).

[11] John R. Kohl. *The Global English Style Guide: Writing Clear, Translatable Documentation for a Global Market*. SAS Press, 2008. 310 pp., 9 chap., 4 app., index: 14 pp. (4%). `http://www.globalenglishstyle.com`.

[12] Helmut Kopka and Patrick W. Daly. *Guide to LATEX*. Pearson Education, Inc., Boston, MA, 4th edition, February 2004. 597 pp., 18 chap., bib., index: 39 pp. (6%).

[13] Fred Leise and Devon Thomas. *Best Practices for Indexing*. American Society for Indexing, Tempe, AZ, 2015. 76 pp., 11 chap., index: 6 pp. (7%); 7 appendices for categories of books.

[14] Frank Mittelbach, Michel Goossens with Johannes Braams, David Carlisle, and Chris Rowley. *The LATEX Companion*. Addison-Wesley, 2nd edition, 2004. 961 pp., 14 chap., bib., index: 98 pp. (10%).

[15] Nancy C. Mulvany. *Indexing Books*. University of Chicago Press, 2nd edition, 2005. 315 pp., 10 chap., gloss., index: 25 pp. (8%). `http://press.uchicago.edu/ucp/books/book/chicago/I/bo3625262.html`.

[16] Jonathan Price and Henry Korman. *How to Communicate Technical Information: A handbook of software and hardware documentation*. Addison-Wesley, 1993. 402 pp., 21 chap., index: 22 pp. (5%).

[17] Maïeul Rouquette. `indextools` — producing multiple indices, 2015. `http://ctan.org/pkg/indextools`.

[18] Arlene G. Taylor. *The Organization of Information*. Libraries Unlimited, 1999. 280 pp., 10 chap., 3 app., gloss., 22 pp., index: 25 pp. (8%). `http://www.abc-clio.com/product.aspx?isbn=9781591585862`.

◇ Ronald J. Fehd
`ronald dot j dot fehd (at) gmail dot com`
`https://www.linkedin.com/in/ronald-fehd-5125991`

The compiling of an index is interesting work, though some authors are apt to find it tedious and delegate the work to others. The proofreader who undertakes it will find that it is splendid mental exercise and brings out his latent editorial capability.

Albert H. Highton,
Practical Proofreading, (1926)
quoted by Knuth, *The TEXbook*, pg. 481

TEXShop review

Frans Absil

Abstract

This paper is an introduction to and review of the TEXShop Mac OS X program for typesetting TEX document source files and previewing the output. Features of this tool and user experience will be presented. This information might be helpful to the novice user looking for a TEX typesetting environment on the Mac.

1 Introduction

For some decades now I have been using plain TEX and LATEX computer typesetting for scientific documents, lecture notes and other stuff. After starting typesetting on DEC VAX/VMS computer terminals I was pleased to see the integrated development environments for both Windows and MacOS.

Currently, I use TEXShop version 1.42, installed with the teTEX distribution on an old Mac PowerBook G4. (My office PC runs a MiKTEX distribution with TEXnicCenter.) My first impression was that TEXShop is fairly basic, but reliable.

The next section will describe a sampling of the features included in TEXShop, and report on my user experience.

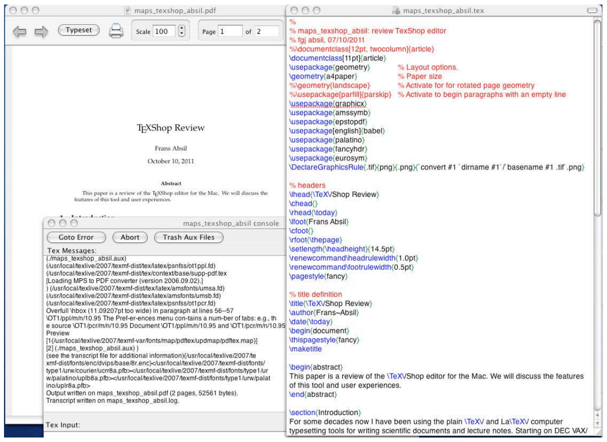
2 TEXShop features

This section lists essential features of TEXShop, with some user comments added. In a typical typesetting session the program will open three windows, as shown in Figure 1: the source editor, the PDF preview and the console window that reports the document compilation progress, warnings and errors.

2.1 Preferences and typesetting engine

The **Preferences** menu contains a number of tabs: e.g., the source `Document` editor font setting and window position, `Preview` window parameter settings, the typesetting `Engine` binary path and and options lists. The default typesetting engine is tickmarked in the `Typesetting` tab. A full range of engines, including TEX, LATEX, ConTEXt, and XeLATEX is available in the pulldown menu `Typeset`; there, also, workflow enhancement scripts or tools such as pdfTEX, TEX+Ghostscript, BibTEX and MakeIndex can be found.

As a LATEX user I select either the pdfTEX or TEX+Ghostscript typesetting engine; the latter is required for including scientific diagrams created with



TEXShop Review

Frans Absil

October 10, 2011

Abstract

This paper is a review of the TEXShop editor for the Mac. We will discuss the features of this tool and user experiences.

```
Tex Messages:
(./maps_texshop_absil.aux)
(/usr/local/texlive/2007/texmf-dist/tex/latex/psnfss/ot1ppl.fd)
(/usr/local/texlive/2007/texmf-dist/tex/context/base/supp-pdf.tex
[Loading MPS to PDF converter (version 2006.09.02).]
) (/usr/local/texlive/2007/texmf-dist/tex/latex/amsfonts/umsa.fd)
(/usr/local/texlive/2007/texmf-dist/tex/latex/amsfonts/umsb.fd)
(/usr/local/texlive/2007/texmf-dist/tex/latex/psnfss/ot1pcr.fd)
Overfull \hbox (11.09207pt too wide) in paragraph at lines 56--57
\OT1/ppl/m/n/10.95 The Pref-er-ences menu con-tains a num-ber of tabs: e.g., th
e source \OT1/pcr/m/n/10.95 Document \OT1/ppl/m/n/10.95 and \OT1/pcr/m/n/10.95
Preview
[1{/usr/local/texlive/2007/texmf-var/fonts/map/pdftex/updmap/pdftex.map}]
[2] (./maps_texshop_absil.aux) )
(see the transcript file for additional information){/usr/local/texlive/2007/te
xmf-dist/fonts/enc/dvips/base/8r.enc}</usr/local/texlive/2007/texmf-dist/fonts/
type1/urw/courier/ucrr8a.pfb></usr/local/texlive/2007/texmf-dist/fonts/type1/ur
w/palatino/uplb8a.pfb></usr/local/texlive/2007/texmf-dist/fonts/type1/urw/palat
ino/uplr8a.pfb>
Output written on maps_texshop_absil.pdf (2 pages, 52561 bytes).
Transcript written on maps_texshop_absil.log.

Tex Input:
```

```
%
% maps_texshop_absil: review TexShop editor
% fgj absil, 07/10/2011
%\documentclass[12pt, twocolumn]{article}
\documentclass[11pt]{article}
\usepackage{geometry}           % Layout options.
\geometry{a4paper}              % Paper size
%\geometry{landscape}           % Activate for for rotated page geometry
%\usepackage[parfill]{parskip}  % Activate to begin paragraphs with an empty line
\usepackage{graphicx}
\usepackage{amssymb}
\usepackage{epstopdf}
\usepackage[english]{babel}
\usepackage{palatino}
\usepackage{fancyhdr}
\usepackage{eurosym}
\DeclareGraphicsRule{.tif}{png}{.png}{`convert #1 `dirname #1`/`basename #1 .tif`.png}

% headers
\lhead{\TeXVShop Review}
\chead{}
\rhead{\today}
\lfoot{Frans Absil}
\cfoot{}
\rfoot{\thepage}
\setlength{\headheight}{14.5pt}
\renewcommand\headrulewidth{1.0pt}
\renewcommand\footrulewidth{0.5pt}
\pagestyle{fancy}

% title definition
\title{\TeXVShop Review}
\author{Frans~Absil}
\date{\today}
\begin{document}
\thispagestyle{fancy}
\maketitle

\begin{abstract}
This paper is a review of the \TeXVShop editor for the Mac. We will discuss the features
of this tool and user experiences.
\end{abstract}

\section{Introduction}
For some decades now I have been using the plain \TeXV and La\TeXV computer
typesetting tools for writing scientific documents and lecture notes. Starting on DEC VAX/
```

Figure 1: Overview of the three TEXShop windows: editor, previewer and console

the `PStricks` package. Naturally, in that case all figures have to be available as separate PostScript `.eps` files, whereas for the former option all figures are included as PDF files. And, by the way, if one drops a recognized figure file type on the editor window, it will generate the appropriate `\includegraphics` command.

2.2 Editor features

The source file editor uses colour coding, with LATEX commands shown in blue, comments in red, and parenthesis grouping in green, for easy code consistency checks. Although line numbers are not displayed, the **Edit** menu contains a `Line Number` and `Go to Error` command. This menu also contains items for running a spell checker and showing document statistics, such as word, line and character count excluding the LATEX commands.

The **Find** panel, under the **Window** menu, has a number of nifty options: it accepts regular expressions for advanced search, it applies find and replace to selected subregions in the source document (local scope) and it will list all appearances of the search string, a convenient option for navigation and as check before a global replace. It will also remember all previous search and replace strings for the current session.

2.3 Panels for the novice user

LATEX commands are mnemonic. However, at the entry level TEXShop contains menu items and special panels for entering source code. Figure 2 shows the menu for font selection. A separate LATEX panel (see Figure 3) shows the most frequently used symbols, environments and other document elements. Clicking an item on the panel will put the LATEX source code

First published in *MAPS* 46 (Najaar 2015), pp. 5–7. Reprinted with permission.

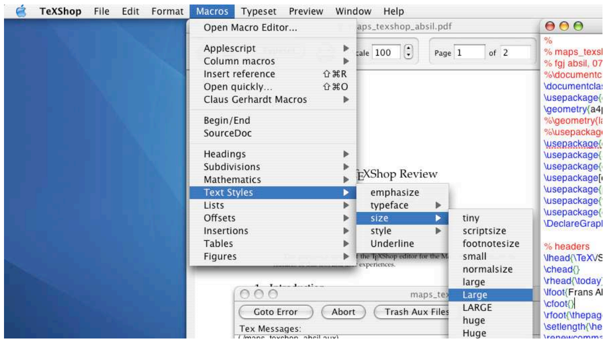


Figure 2: TEXShop's **Format** menu with font selection items

in the current document in the editor. TEXshop supports multiple editor windows.

In the **Preview** window there are the usual scrolling, paging and scaling functions. A nice feature is the magnifying glass; when selected, clicking anywhere in the previewed document will show that section in sufficient magnification to inspect typesetting details.

The novice user might also refer to the concise but adequate **Help** menu.

2.4 Miscellaneous

The **Format** menu has items for (un)commenting or indenting blocks of source code, a feature that is useful when debugging documents.

The **Macros** menu contains items for automating a workflow. My favourite from this set is the `Insert reference`: it opens a window with a list of `\label` commands in the current document for selecting the appropriate entry. Also convenient is the `Bibliography` Applescript, which does the multiple runs needed for creating bibliographical references in a document.

For larger documents, which consist of a set of smaller files (e.g., book chapters), there is the option of setting the project root, i.e., the path to the project `main.tex` file; this is available under the **File** menu. What is missing, however, is a window representing the structure of a large-scale project as a tree graph with directory paths to source files, figures etc. TEXnicCenter has this window, which is a great help during editing and debugging report or book class documents.

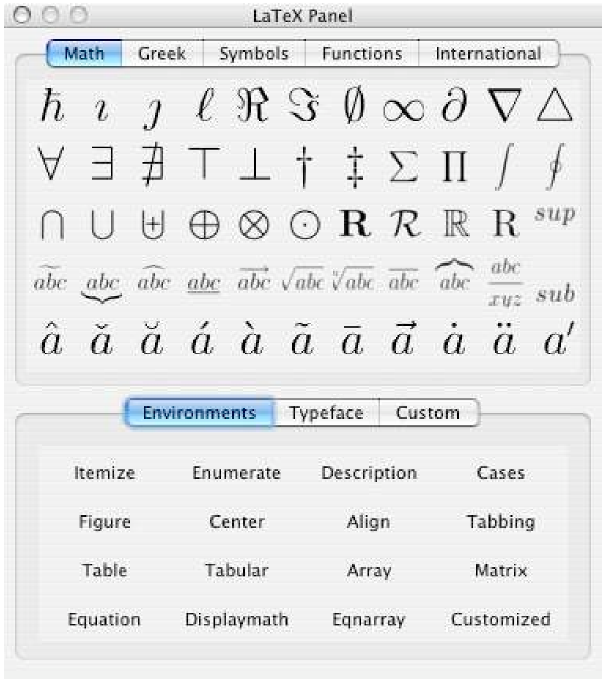


Figure 3: LATEX panel for selection of common symbols and document elements.

(Newer versions of TEXShop than mine can display a PDF outline, e.g., as made by `hyperref`, in the Preview Window. Clicking in the outline, then CMD-clicking in the PDF, will get to the corresponding source file and position. This is some of the same functionality, though quite a different approach.)

3 Conclusion

This old version of TEXShop, running on a Mac OS 10.3.8 operating system, is a reliable workhorse. Although there are differences I have no problem using this tool and TEXnicCenter on my Windows PC in parallel.

A much more recent version, at this writing TEXShop v. 3.59 for Mac OS X 10.7 (Lion) and later systems, is available at `http://www.uoregon.edu/~koch/texshop/`. It is also included on the TEX Live DVD. A comparison between many LATEX editors can be found at `https://en.wikipedia.org/wiki/Comparison_of_TeX_editors`; this has a table with the features of each program.

◇ Frans Absil
`frans dot absil (at) gmail dot com`

TEXworks: A simple GUI with advanced options

Sytse Knypstra

1 Abstract

Out of the many available TEX (LATEX, ConTEXt) editors, TEXworks is relatively new. It excels in simplicity of the user interface, the linking between the source text and the PDF output and in its substantial degree of flexibility. With the use of scripts the user can add nearly any desired option.

2 Introduction

As a small-scale user of ConTEXt I don't much care which editor I use, as long as it is easy to install, easy to type in the source text and transform this into a PDF file by simply pressing a button. Furthermore, a PDF viewer should be attached which doesn't protest if after a new run an old version has to be overwritten. My starting point is in fact: the simpler the better.

3 Simplicity

TEXworks meets these requirements. It is easy to install and use and a viewer is attached that shows the PDF result without any problems, even with a magnifying glass. This simplicity in use is not a consequence of the fact that the program is in its infancy. It is deliberate. The authors, originally Jonathan Kew and now Stefan Löffler, were inspired by the success of TEXShop which is available only for the Mac OS X platform. That program made TEX (LATEX, ConTEXt) more widely accessible to a large group of new users.

TEXworks is available for all common platforms, including Linux and Windows. It does not look intimidating for newbies: no screen populated with pop-up menus, no menu bars with mathematical symbols, only two plain windows that fill the screen (see figure 1). On the left hand side we see a window for the source text, with an output panel below which disappears when the TEX-engine does not report any errors, and on the right hand side a window with the resulting PDF. It is easy to jump between corresponding places: 'Ctrl+left mouse click' causes the matching paragraph to be highlighted. This linkage is provided by SyncTEX.

TEXworks does not focus on one macro package as some other editors do. This means that specific LATEX options are absent in the menu, which adds to the simplicity of the user interface. But it also has its downside. Some more advanced or specific LATEX options which in other editors are included in the menu have to be added by editing configuration files. This applies in particular to spellchecking and auto-completion. I have been satisfied with the standard options of TEXworks, but for the preparation of this paper I investigated some additional options.

First published in *MAPS* 46 (Najaar 2015), pp. 13–15. Translation by the author. Reprinted with permission.

4 Spellchecking and auto-completion

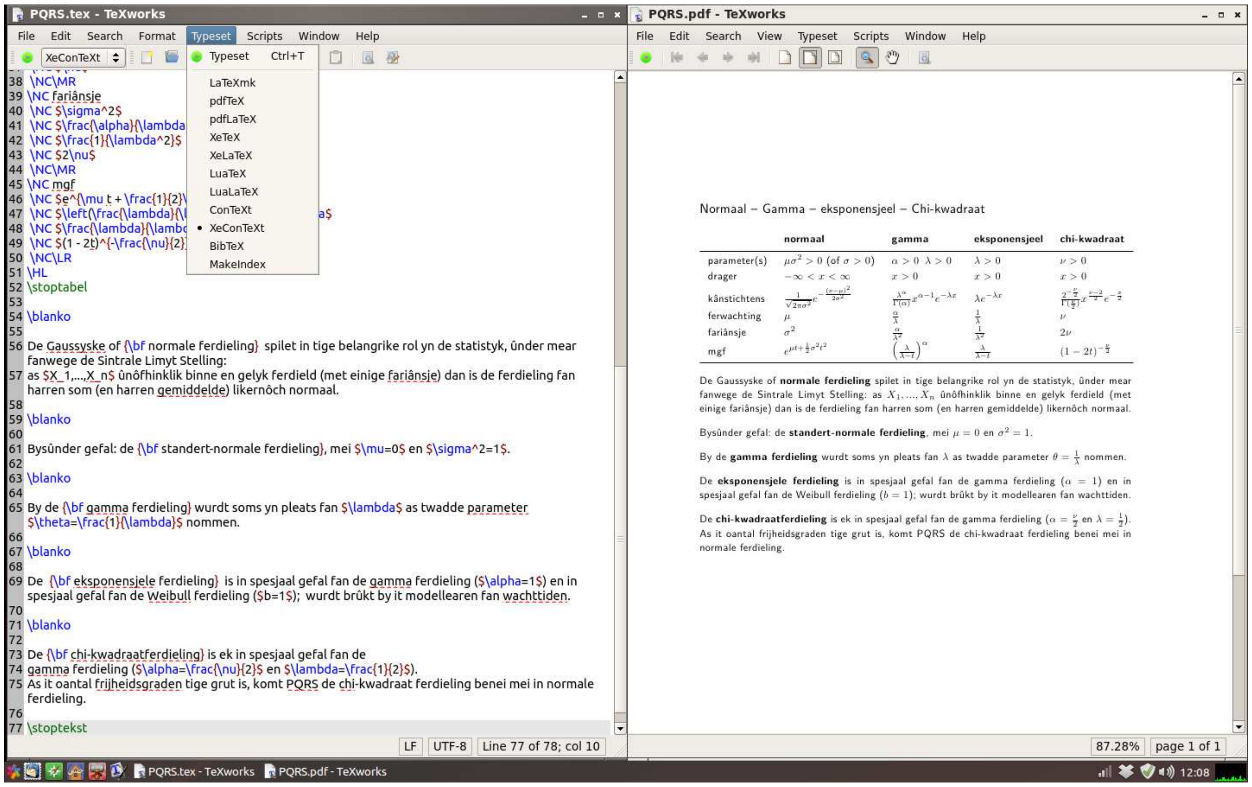
First, spellchecking. Normally I don't bother to install spellcheckers (I never make spelling mistakes :)), but of course it is possible to do so. In TEXworks it is rather simple: you need a `.aff` and a `.dic` file of the language to be used. I found them somewhere hidden in my Linux installation under `~/.mozilla/firefox`, and it seems they are also included in LibreOffice. Next you copy them to `/usr/share/myspell/dicts`, or you can create a link to that directory. If you then start TEXworks, you can tick the desired language in the menu under `Edit` → `Spelling`. The result, with `fy` (West-Frisian, my mother tongue), can be seen in figure 1. The spellchecker ignores all words beginning with \, regardless of whether they are TEX, LATEX, or ConTEXt commands.

Second, auto-completion. In the appropriate configuration file I inserted the line: `tw:=TeXworks`. Then, after typing '`tw`', followed by 'TAB', one of the two following words shows up: '`\textwidth`' or '`TeXworks`'. The first word is a member of the set of approximately 600 abbreviations that were entered in advance for LATEX; for this article I need the second word. If the wrong word appears first, you must press 'TAB' again. For ConTEXt a script `autocomplete.js` is available, made by Henrik Skov Midtiby (see under Sources). After copying this file to the correct folder, your ConTEXt commands are auto-completed if you press 'Ctrl+M' one or more times.

5 Scripts

This brings us to the subject of 'scripts'. The authors of TEXworks wanted to keep their program simple in order to attract beginners, but at the same time they did not want to put off experienced users who need more advanced options. Therefore it is possible to add scripts, in principle in one of the scripting languages QtScript, Lua or Python. In practice all existing scripts have been written in QtScript (very similar to JavaScript; the filenames also end in `.js`).

Scripts exist in two flavours: hook scripts and standalone scripts. Hook scripts are executed when certain situations arise, for example immediately after starting TEXworks, or following the compilation of the source text. Standalone scripts, on the other hand, are executed by selecting menu items or pressing corresponding key combinations. One


PQRS.tex - TeXworks
File Edit Search Format Typeset Scripts Window Help
XeConTeXt
Typeset Ctrl+T
LaTeXmk
pdfTeX
pdfLaTeX
XeTeX
XeLaTeX
LuaTeX
LuaLaTeX
ConTeXt
• XeConTeXt
BibTeX
MakeIndex
LF UTF-8 Line 77 of 78; col 10
PQRS.pdf - TeXworks
File Edit Search View Typeset Scripts Window Help
87.28% page 1 of 1
PQRS.tex - TeXworks PQRS.pdf - TeXworks 12:08

```
 \NC\MR
\NC fariânsje
 \NC $\sigma^2$
 \NC $\frac{\alpha}{\lambda
 \NC $\frac{1}{\lambda^2}$
 \NC $2\nu$
 \NC\MR
\NC mgf
 \NC $e^{\mu t + \frac{1}{2}\
 \NC $\left(\frac{\lambda}{\l                        a$
 \NC $\frac{\lambda}{\lambd
 \NC $(1 - 2t)^{-\frac{\nu}{2}}
 \NC\LR
\HL
\stoptabel

\blanko

De Gaussyske of {\bf normale ferdieling}  spilet in tige belangrike rol yn de statistyk, ûnder mear
fanwege de Sintrale Limyt Stelling:
as $X_1,...,X_n$ ûnôfhinklik binne en gelyk ferdield (met einige fariânsje) dan is de ferdieling fan
harren som (en harren gemiddelde) likernôch normaal.

\blanko

Bysûnder gefal: de {\bf standert-normale ferdieling}, mei $\mu=0$ en $\sigma^2=1$.

\blanko

By de {\bf gamma ferdieling} wurdt soms yn pleats fan $\lambda$ as twadde parameter
$\theta=\frac{1}{\lambda}$ nommen.

\blanko

De  {\bf eksponensjele ferdieling}  is in spesjaal gefal fan de gamma ferdieling ($\alpha=1$) en in
spesjaal gefal fan de Weibull ferdieling ($b=1$);  wurdt brûkt by it modellearen fan wachttiden.

\blanko

De {\bf chi-kwadraatferdieling} is ek in spesjaal gefal fan de
gamma ferdieling ($\alpha=\frac{\nu}{2}$ en $\lambda=\frac{1}{2}$).
As it oantal frijheidsgraden tige grut is, komt PQRS de chi-kwadraat ferdieling benei mei in normale
ferdieling.

\stoptekst
```

Normaal – Gamma – eksponensjeel – Chi-kwadraat

	normaal	**gamma**	**eksponensjeel**	**chi-kwadraat**
parameter(s)	$\mu\sigma^2 > 0$ (of $\sigma > 0$)	$\alpha > 0$ $\lambda > 0$	$\lambda > 0$	$\nu > 0$
drager	$-\infty < x < \infty$	$x > 0$	$x > 0$	$x > 0$
kânstichtens	$\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\frac{\lambda^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\lambda x}$	$\lambda e^{-\lambda x}$	$\frac{2^{-\frac{\nu}{2}}}{\Gamma(\frac{\nu}{2})}x^{\frac{\nu-2}{2}}e^{-\frac{x}{2}}$
ferwachting	μ	$\frac{\alpha}{\lambda}$	$\frac{1}{\lambda}$	ν
fariânsje	σ^2	$\frac{\alpha}{\lambda^2}$	$\frac{1}{\lambda^2}$	2ν
mgf	$e^{\mu t+\frac{1}{2}\sigma^2 t^2}$	$\left(\frac{\lambda}{\lambda-t}\right)^\alpha$	$\frac{\lambda}{\lambda-t}$	$(1-2t)^{-\frac{\nu}{2}}$

De Gaussyske of **normale ferdieling** spilet in tige belangrike rol yn de statistyk, ûnder mear fanwege de Sintrale Limyt Stelling: as $X_1, ..., X_n$ ûnôfhinklik binne en gelyk ferdield (met einige fariânsje) dan is de ferdieling fan harren som (en harren gemiddelde) likernôch normaal.

Bysûnder gefal: de **standert-normale ferdieling**, mei $\mu = 0$ en $\sigma^2 = 1$.

By de **gamma ferdieling** wurdt soms yn pleats fan λ as twadde parameter $\theta = \frac{1}{\lambda}$ nommen.

De **eksponensjele ferdieling** is in spesjaal gefal fan de gamma ferdieling ($\alpha = 1$) en in spesjaal gefal fan de Weibull ferdieling ($b = 1$); wurdt brûkt by it modellearen fan wachttiden.

De **chi-kwadraatferdieling** is ek in spesjaal gefal fan de gamma ferdieling ($\alpha = \frac{\nu}{2}$ en $\lambda = \frac{1}{2}$). As it oantal frijheidsgraden tige grut is, komt PQRS de chi-kwadraat ferdieling benei mei in normale ferdieling.

Figure 1: Screenshot of TEXworks with expanded *Typeset* menu.

supplied example makes a selected text bold. The script places '`\textbf{`' and '`}`' around the selected text. Using scripts offers many possibilities, but in order to create them, you have to know the scripting language. It's much easier to download published scripts from the Internet, place them in the correct folder and, if necessary, adapt them to your own needs. The boldface example can be adapted for ConTEXt by replacing '`\textbf{`' by '`{\bf`'.

A convenient script for LATEX users is an available hook script that is executed immediately after compiling (at the event `AfterTypeset`). If an error is detected during compilation, it opens a second tab page in the output panel with a list of errors found, which you can then trace and correct. The first tab page contains *all* output notifications.

6 Overview

Not (yet) implemented is a method of folding text, which would give you a better overview over long stretches of text. But there is an alternative: you can open a panel (next to the source text) in which the structure of chapters and (sub)sections is shown, along with possible bookmarks. A bookmark is created by putting a `%` at the start of a line. Hence it does not appear in the resulting PDF. By clicking the section title or bookmark you jump to the desired place in the source text.

In summary, TEXworks is a very suitable editor to start with in the TEX world. A convenient option is the possibility to easily jump between corresponding places in the source text and the PDF file. For more experienced users (mastering the QtScript language) adding scripts will extend the options available almost without limit.

References

[1] `http://www.tug.org/texworks`. TEXworks site.

[2] `http://github.com/texworks`. Development site for TEXworks.

[3] `http://www.youtube.com/watch?v=9-Z43CSPgM0`. Jonathan Kew's presentation for the TUG 2010 conference: *TEXworks for newcomers and what's new for old hands.*

[4] `http://twscript.paulanorman.info/index.html`. The scripting site of Paul A. Norman.

[5] `https://github.com/henrikmidtiby/TeXworks-scripts`. Contains the script `autocomplete.js` for ConTEXt by Henrik Skov Midtiby.

◇ Sytse Knypstra
`sytse dot knypstra (at) home dot nl`

TeXstudio: Especially for LaTeX newbies

Siep Kroonenberg

Abstract

TeXstudio is the default editor of the TeX Live installation at the Rijksuniversiteit Groningen. This article tries to show how TeXstudio can help new users come to grips with LaTeX and what makes it a good choice for a LaTeX introduction.

Introduction

The TeX installation at our university, the Rijksuniversiteit Groningen in the Netherlands, includes TeXstudio as one of three (LA)TeX editors. TeXstudio, the subject of this article, is the initial default editor. Its features make it especially suitable for first-time LaTeX users: there are many GUI elements for entering mathematics and LaTeX code, there are buttons for one-click compiling and previewing LaTeX documents, and the editor gives a lot of useful feedback.

TeXstudio is open source and cross-platform, and is actively being developed. This article refers to version 2.10.4.

The main window

In the TeXstudio window, as shown in figure 1, the editing area is surrounded by various panels and toolbars: on the left a ‘structure view’ of the document which can be used for navigation, below a message area and on the right a preview window. The previewer has been adopted from TeXworks and supports source–pdf synchronization.

For illustration purposes, a document of some complexity has been loaded. The section ‘Historical remarks’ is in view in both the editor and the previewer, and is highlighted in the structure view at the left. The message panel shows the command-line used for the latest compilation.

Help and documentation

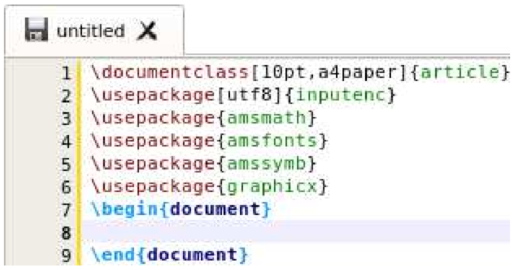
TeXstudio help consists of two HTML documents. The first is a manual for TeXstudio itself, with sections on configuration, editing, compiling and more. The other one is a copy of the third-party document ‘LaTeX 2_ε: An unofficial reference manual’, but linked to a custom stylesheet. This file is also used for popup help.

For an introduction to LaTeX you need to look elsewhere, such as ‘The Not So Short Introduction to LaTeX 2_ε’, or my own tutorial-style ‘RUG LaTeX Course’ which specifically refers to TeXstudio. Both introductions, and the original version of the reference mentioned above, are available from CTAN. Our local TeX installation has menu entries for all three documents.

Starting out

We have a few choices for starting a new document. The File menu has items ‘New’, which starts with a blank document, and ‘New from Template’, which creates a new document and adds templates for a title and abstract.

But in this article we start a new document with the ‘Quick Start’ entry from the Wizards menu; see figure 2. If we just click the OK button, we see the following in the edit panel:



```
\documentclass[10pt,a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{graphicx}
\begin{document}

\end{document}
```

Subsequently we can start entering text between `\begin{document}` and `\end{document}`.

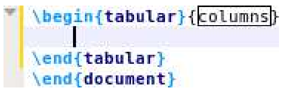
Previewing Now that we have some text, it is a good moment to save the file and get acquainted with the edit–compile–preview cycle. After saving the file, we click first the Compile button (▶) and then the View button (🔍) or, more efficiently, just the The Build & View button (⏩) which does everything in one go. Either way, the on-screen result is a much emptier version of figure 1.

LaTeX markup

Our next undertaking is adding text with markup.

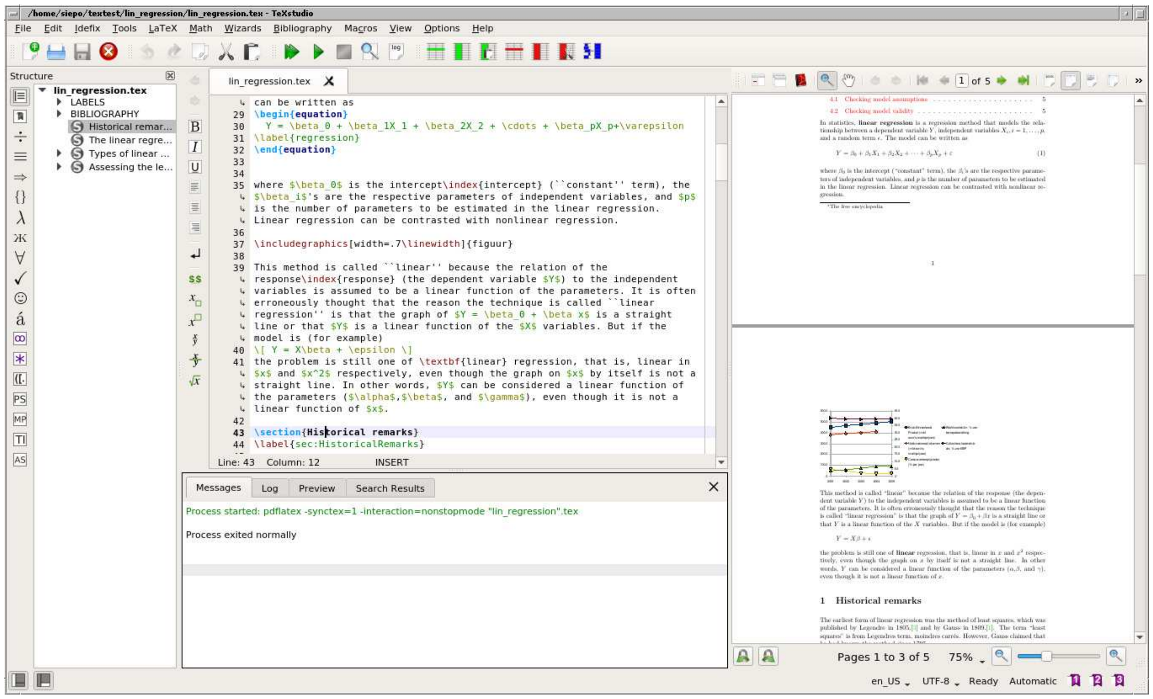
Bold and italic. This is to assure new users that they are not on completely alien soil. The inner vertical toolbar, *i.e.* at the left of the editing area, contains buttons for these: **B** and *I*. Of course, these text styles will be applied to any text which happens to be selected at the time: `\textit{Hel}lo, \textbf{world}`.

The LaTeX menu. These and other styles are also available via the LaTeX / Font Styles menu, but this menu contains many more items and submenus, *e.g.* clicking LaTeX / Tabular Environment / `\begin{tabular}` gets us:



```
\begin{tabular}{columns}
	|
\end{tabular}
\end{document}
```

Autocompletion and tooltip help. To see these features at work, we start typing: `\tabl...`

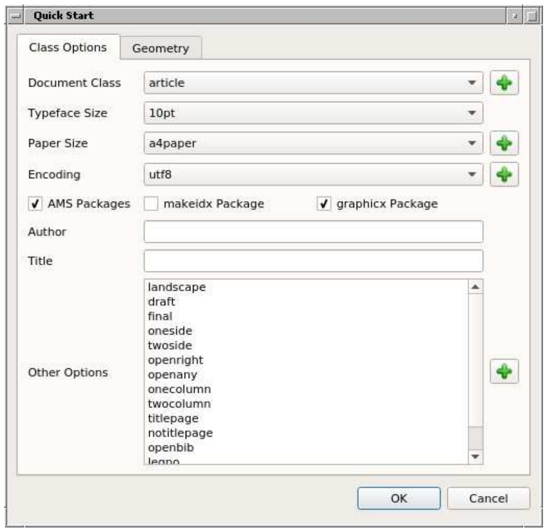


```
can be written as
\begin{equation}
  Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \cdots + \beta_pX_p+\varepsilon
\label{regression}
\end{equation}


where $\beta_0$ is the intercept\index{intercept} (``constant'' term), the
$\beta_i$'s are the respective parameters of independent variables, and $p$
is the number of parameters to be estimated in the linear regression.
Linear regression can be contrasted with nonlinear regression.

\includegraphics[width=.7\linewidth]{figuur}

This method is called ``linear'' because the relation of the
response\index{response} (the dependent variable $Y$) to the independent
variables is assumed to be a linear function of the parameters. It is often
erroneously thought that the reason the technique is called ``linear
regression'' is that the graph of $Y = \beta_0 + \beta x$ is a straight
line or that $Y$ is a linear function of the $X$ variables. But if the
model is (for example)
\[ Y = X\beta + \epsilon \]
the problem is still one of \textbf{linear} regression, that is, linear in
$x$ and $x^2$ respectively, even though the graph on $x$ by itself is not a
straight line. In other words, $Y$ can be considered a linear function of
the parameters ($\alpha$,$\beta$, and $\gamma$), even though it is not a
linear function of $x$.

\section{Historical remarks}
\label{sec:HistoricalRemarks}
```

```
Process started: pdflatex -synctex=1 -interaction=nonstopmode "lin_regression".tex

Process exited normally
```

/home/siepo/textest/lin_regression/lin_regression.tex - TeXstudio
File Edit Idefix Tools LaTeX Math Wizards Bibliography Macros View Options Help
Structure
lin_regression.tex
LABELS
BIBLIOGRAPHY
Historical remar...
The linear regre...
Types of linear ...
Assessing the le...
lin_regression.tex
Line: 43 Column: 12 INSERT
Messages Log Preview Search Results
1 of 5
Pages 1 to 3 of 5 75%
en_US UTF-8 Ready Automatic

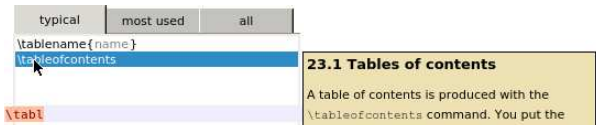
Figure 1: Main TeXstudio window



Document Class	article
Typeface Size	10pt
Paper Size	a4paper
Encoding	utf8
✓ AMS Packages	makeidx Package; ✓ graphicx Package
Author	
Title	
Other Options	landscape, draft, final, oneside, twoside, openright, openany, onecolumn, twocolumn, titlepage, notitlepage, openbib, leqno

Figure 2: The Quick Start wizard

TeXstudio offers a list of completions, including `\tableofcontents`, and adds a tooltip with information about the highlighted command.



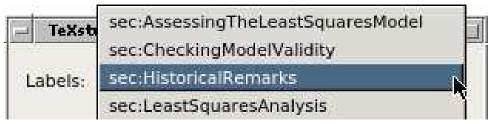
```
\tablename{name}
\tableofcontents
```

23.1 Tables of contents

A table of contents is produced with the `\tableofcontents` command. You put the

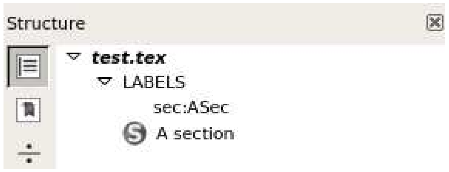
The tooltip text is taken from the LaTeX reference mentioned earlier. However, in my tests TeXstudio occasionally did not manage to extract useful information. The control sequence remains pink/orange until it has become a valid LaTeX command.

Cross-references. Also of note is the handling of cross-references. If we enter `\ref`, we are presented with a list of existing labels. This is how such a list looks when we entered `\ref` via the GUI:



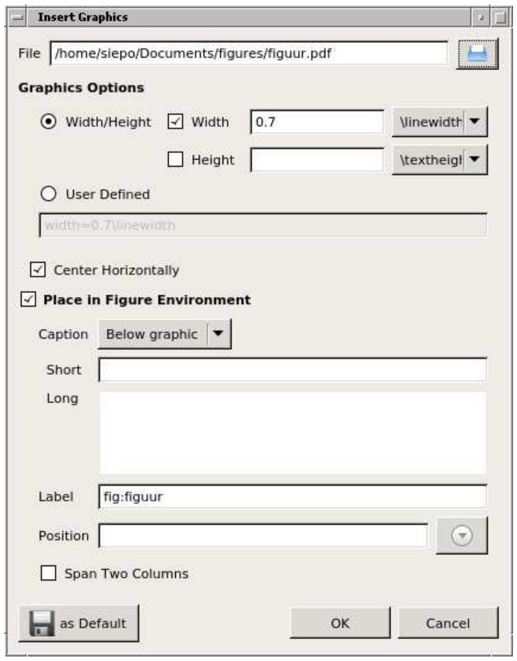
```
sec:AssessingTheLeastSquaresModel
sec:CheckingModelValidity
sec:HistoricalRemarks
sec:LeastSquaresAnalysis
```

With one section and one cross-reference label in the document, the Structure panel might look as follows:



Math. TeXstudio has an extensive Math menu, including inline and display math and constructs such as `\frac` and `\begin{array}`. Selecting *e.g.* the `\frac` item results in the following: `\frac{num}{den}`. `\frac` and various other common constructs are also represented in the inner vertical toolbar ($\frac{x}{y}$).

The Structure panel can turn into one of several palettes of mathematical symbols. These can be invoked with the buttons of the outer vertical toolbar,



Insert Graphics

File /home/siepo/Documents/figures/figuur.pdf

Graphics Options

[✓ Width/Height] [✓ Width] 0.7 \linewidth

[ ] Height \textheight

( ) User Defined

width=0.7\linewidth

[✓ Center Horizontally]

[✓ Place in Figure Environment]

Caption: Below graphic

Short:

Long:

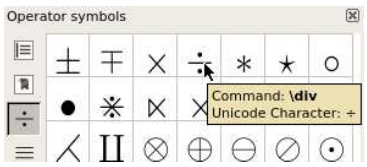
Label: fig:figuur

Position:

[ ] Span Two Columns

Figure 3: Include Graphic wizard. The graphic path will show up in the source as '`figures/figuur`'.

at the far left of the TeXstudio window. Here is a fragment of the Operators palette:

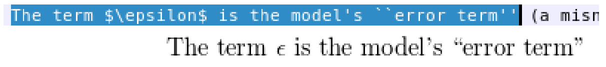


Command: \div
Unicode Character: ÷

Arrays and tabulars. There are fairly basic array- and tabular wizards, but the LaTeX menu also contains a 'Manipulate Tables' submenu, with items for removing and adding rows and columns, and for pasting columns (it is best first to apply the item 'Align Columns'). This submenu handles both math mode arrays and text mode tabulars. These operations are also accessible via a section of the toolbar at the top:

Insert Graphic wizard. Another wizard to mention here is the Insert Graphic wizard; see figure 3. It uses a file browser, and creates a relative path for the graphic file if at all possible.

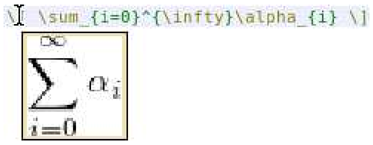
Previewing a selection. When we right-click with some text selected we can choose an option Preview Selection/Parentheses. Depending on the configuration, this produces a preview of the selection either in the Preview tab of the message area, or inline in the text, like this:



```
The term $\epsilon$ is the model's ``error term'' (a misn
```

The term ϵ is the model's "error term"

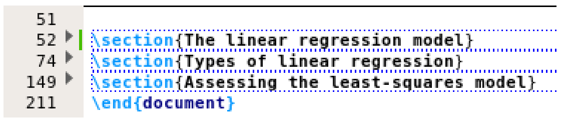
In addition, if the mouse cursor hovers for a while inside a display math environment, a preview of the equation will pop up:



```
\[ \sum_{i=0}^{\infty}\alpha_{i} \]
```

$$\sum_{i=0}^{\infty}\alpha_{i}$$

Outline mode. TeXstudio has an outline mode, which can be accessed via the Collapse- and Expand submenus of the View menu:



```
\section{The linear regression model}
\section{Types of linear regression}
\section{Assessing the least-squares model}
\end{document}
```

Bibliography support

During compilation, TeXstudio automatically runs BibTeX if necessary, or biber if that has been configured as the default.

In addition, TeXstudio has some support for editing BibTeX databases, although it does not pretend to be a full-blown bibliography manager.

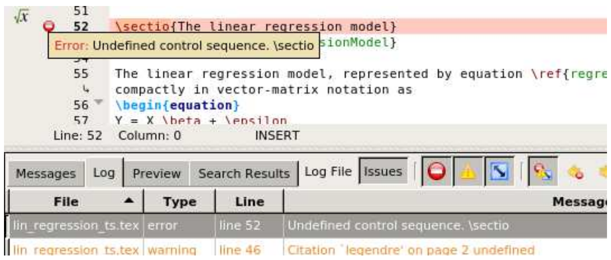
The Bibliography menu contains a long list of publication types. If we select *e.g.* 'Article in Conference Proceedings' then the following code is generated:

```
@InProceedings{ID,
author = {author},
title = {title},
booktitle = {booktitle},
OPTcrossref = {crossref},
...
OPTannote = {annote},
}
```

The idea is to remove `OPT` from those fields which we actually use. When we are done with the entry, the menu item Bibliography / Clean will remove all remaining `OPT` fields.

Error handling

TeXstudio is rather emphatic about error reporting, *e.g.*



```
\sectio{The linear regression model}
Error: Undefined control sequence. \sectio   sionModel}

The linear regression model, represented by equation \ref{regre
compactly in vector-matrix notation as
\begin{equation}
Y = X \beta + \epsilon
Line: 52   Column: 0          INSERT
```

File	Type	Line	Message
lin_regression_ts.tex	error	line 52	Undefined control sequence. \sectio
lin_regression_ts.tex	warning	line 46	Citation `legendre' on page 2 undefined

In the editing area, the cursor has jumped to the error, and the message area now shows the Log tab, with the first error highlighted. There are also previous/next error buttons:

Configuration

TeXstudio has extensive options for configuration under Options / Configure TeXstudio; here are a few highlights.

For finding LaTeX and friends (Options / Commands), it relies on the search path and, by default, does not explicitly store their locations, but on Windows there are exceptions, particularly the various viewers.

On the Build tab, there are options for setting the default engine (Compiler) and bibliography processor.

The editor is also very configurable: font and font size, various aspects of spell-checking, grammar checking, encoding detection, tooltips, syntax highlighting, *etc.* Turning off some of these options can help to make the editor less noisy.

More advanced configuration includes customization of the menus, of keyboard shortcuts, and of the set of autocompletion files.

Users can add templates via File / Make Template, and macros via Macros / Edit Macros. Users can add options to the Quick Start wizard; see the ✚ symbols in the Quick Start screenshot (figure 2).

Under Windows, TeXstudio does not use the registry, but stores configuration information in a series of text files in a subdirectory `texstudio` of `%appdata%`, analogous to what it does under Linux.

TeXstudio, Texmaker and Kile

TeXstudio is a fork of Texmaker, which is also cross-platform. If you find TeXstudio a bit over the top you may want to try Texmaker. Pascal Brachet, the author of Texmaker, is also the original author of Kile, a LaTeX editor running under KDE. Not unexpectedly, there is a family resemblance between the three editors.

Summary

I have given some examples of how the TeXstudio interface helps new users on their way and saves the instructor a lot of explaining. Many users seem to like it well enough, even for the long haul.

Notes

This article is based on a Dutch-language article 'TeXstudio, speciaal voor LaTeX starters', pp. 16–22, MAPS 46, 2015 (`http://ntg.nl/maps`). It describes a slightly earlier version of TeXstudio.

The document loaded in figure 1 consists of a Wikipedia page on linear regression manually converted to LaTeX.

Screenshots for this article have been newly created from TeXstudio 2.10.*x*, running on Ubuntu 15.10.

URLs

Kile home page:
`http://kile.sourceforge.net/`

LaTeX 2_ε: An unofficial reference manual:
`http://ctan.org/pkg/latex2e-help-texinfo`

Texmaker home page:
`http://www.xm1math.net/texmaker/`

TeXstudio home page:
`http://texstudio.org/`

TeXstudio repository:
`http://sourceforge.net/p/texstudio`
`/hg/ci/default/tree/`

The Not So Short Introduction to LaTeX 2_ε:
`http://ctan.org/pkg/lshort`

RUG LaTeX Course:
`http://ctan.org/pkg/latexcourse-rug`

◇ Siep Kroonenberg
Groningen
The Netherlands
`siepo (at) cybercomm dot nl`

10 years of TEX Live in Debian

Norbert Preining

Abstract

TEX Live has turned into the most widely used TEX distribution since support ended for teTEX. Debian has carried a packaged version of TEX Live for 10 years now. We review the history of TEX packages in Debian, and in particular the history of TEX Live packaging.

1 Introduction

Getting older, people usually start looking back at things that happened in the past, and I am no different. So I recently realized that this year (2016) there are several *anniversaries* of my involvement in the TEX world: 14 years ago I started building binaries for TEX Live, 11 years ago I proposed packaging TEX Live for Debian, 10 years ago the TEX Live packages entered Debian. There are other things to celebrate next year (2017), namely the 10 year anniversary of the (no longer new) infrastructure (esp. `tlmgr`) of TEX Live packaging, but this will come later. In this article I want to concentrate on my involvement with TEX Live in Debian.

2 Debian releases and TEX systems

The TEX system of choice in Debian was for many years teTEX [8], curated by Thomas Esser. Digging through the Debian archive and combining this with ChangeLog entries as well as personal experiences since I joined Debian, the timeline of TEX in Debian to the best of my knowledge can be found in Table 1.

The history of TEX in Debian is thus split more or less into 10 years of teTEX, and 10 years of TEX Live. While I cannot check back to the ultimate origin, my guess is that already in the very first Debian releases (te)TEX was included. The first release I can confirm (via the Debian archive) shipping teTEX is the release Bo (June 1997). Maintainership during the first 10 years showed some fluctuation: The first years/releases, till about 2002, were dominated by Christoph Martin with Adrian Bunk and few others, who did most of the packaging work on teTEX version 1. After this Atsuhito Kohda with help from Hilmar Preusse and others brought teTEX up to version 2, and from 2004 to 2007 Frank Küster, again with the help of Hilmar Preusse and others, took over most of the work on teTEX. Other names commonly appearing throughout the ChangeLog are Julian Gilbey, Ralf Stubner, LaMont Jones, and C.M. Connelly — and there were many more bug reporters and fixers.

Looking at table I have to mention the incredible amount of work that both Atsuhito Kohda and Frank Küster have put into the teTEX packages, and many of their contributions have been carried over into the TEX Live packages. While there weren't many releases during their maintainership, their work has inspired and supported the packaging of TEX Live to a huge extent.

3 Start of TEX Live

I got involved in TEX Live back in 2002 when I started building binaries for the alpha-linux architecture. I can't remember when I first had the idea to package TEX Live for Debian, but here is a timeline from my first email to the Debian Developers mailing list concerning TEX Live to the first accepted upload:

2005-01-11: *binaries for different architectures in debian packages [1]*
This is my first email to the Debian community about packaging TEX Live. It is easy to see that I didn't have much of a clue about Debian packaging at that time, as I proposed to simply reuse the binaries that are included in TEX Live, instead of properly building them for Debian.

2005-01-25: *Debian-TeXlive Proposal II [2]*
After the initial round of feedback (and flames) I proposed a new layout with adaptations, but still continued to try to avoid rebuilding the binaries.

2005-05-17: *Proposal for a tex-base package [3]*
As we were planning to have two distinct (and overlapping) TEX systems in Debian, together with Frank Küster we proposed a package `tex-base`, later to be named `tex-common`, as basis for both the teTEX and TEX Live packages, providing common basic infrastructure.

2015-06-10: *Bug#312897: ITP: texlive [4]*
The first *official* step in packaging a new 'program' for Debian is the ITP bug — *Intend to package.*

2005-09-17: *Re: Take over of texinfo/info packages [5]*
In the course of preparing TEX Live package I needed to put my hands on several other TEX-related packages, the first being `texinfo`, which was orphaned (without a Debian maintainer) at that time. It was also based on this package that I became a Debian Developer.

2005-11-28: *Re: texlive-basic_2005-1_i386.changes REJECTED [6]*
When a new package is the first time uploaded to Debian, it cannot enter immediately but has to go through a severe scrutiny by the so-called 'ftp-masters'. They check for license compliance, Debian

Date	Version	Name	teTEX/TEX Live	Maintainers
1993–96	<1	?	?	Christoph Martin
6/1996	1.1	Buzz	?	
12/1996	1.2	Rec	?	
6/1997	1.3	Bo	teTEX 0.4	
7/1998	2.0	Ham	teTEX 0.9	
3/1999	2.1	Slink	teTEX 0.9.9N	
8/2000	2.2	Potato	teTEX 1.0	
7/2002	3.0	Woody	teTEX 1.0	
6/2005	3.1	Sarge	teTEX 2.0	Atsuhito Kohda
4/2007	4.0	Etch	teTEX 3.0,	Frank Küster
			TEX Live 2005	NP
2/2009	5.0	Lenny	TEX Live 2007	NP
2/2011	6.0	Squeeze	TEX Live 2009	
5/2013	7.0	Wheezy	TEX Live 2012	
4/2015	8.0	Jessie	TEX Live 2014	
?	?	Stretch	TEX Live ≥2015	

Table 1: History of TEX systems in Debian

policy compliance, and some say their daily level of comfort, before allowing a new package to enter Debian. After my first upload I got extremely negative feedback, including statements like ‘Why do we need another TEX system.’ Together with Frank Küster we drafted a response, which sparked a long discussion about packaging and helped improve the naming of packages (but not especially the packaging itself).

2006-01-12: *Upload of TEX Live 2005-1 to Debian*
The first upload that successfully passed the scrutiny of the ftp-masters.

2006-01-22: *Accepted texlive-base 2005-1 (source all) [7]*
TEX Live packages accepted to Debian/experimental.

One can see from the first emails that at that time I had no idea of correct Debian packaging and proposed to ship the binaries built within the TEX Live system on Debian. What followed was first a long discussion about whether there is any need for “just another” TEX system. The then maintainer Frank Küster took a clear stance in favor of including TEX Live, and after several rounds of proposals, tests, rejections and improvements, the first successful upload of TEX Live packages to Debian/experimental happened on 12 January 2006, so exactly 10 years ago.

4 Packaging

From the beginning, Debian has used a meta-packaging approach. That is, instead of working directly with the TEX Live sources, (Perl) scripts generate Debian source packages from a set of directives. We introduced this extra layer for several reasons:

- The original format of the TEX Live packaging information (`tpm`) was XML files that Debian parsed with an XML parser (`libxml`). I surmise (from what I have seen over the years) that only the Debian packages did proper parsing of these `.tpm` files for packaging.
- TEX Live packages were often reshuffled, and Debian package names changed, which would have otherwise caused a certain level of pain during the creation of original tar files and packaging.
- General flexibility in creating additional packages and arbitrary dependencies.

Although I have never been 100% sure that it was the best idea, the scripts nevertheless remain in place to the present day, only adapted to the new packaging paradigm in TEX Live (without XML) and adding new functionality. This allows me to just kick off one script that does all the work, including building `.orig.tar.gz`, source packages, and binary packages.

For those interested in following the frantic activity during the first few years, there is a file `CHANGES.packaging` [9] which extensively documents the changes made for the years from 2005 to 2011. I don’t want to count the hours that went into this.

5 Development over the years

TEX Live 2005 was just another TEX system but not the preferred one in Debian Etch and earlier. But in May 2006, Thomas Esser announced the end of development for teTEX, which cleared the path for TEX Live as the main TEX system in Debian (and the world!). The next release of Debian, Lenny (1/2009), already carried only TEX Live. Unfortunately it was only TEX Live 2007 and not 2008, mostly due to my having been involved in rewriting the upstream infrastructure based on plain text package descriptions instead of the notorious XML files. This took quite a lot of attention and time from Debian away to upstream development, but this will be discussed in a different post.

Similarly, the release of TEX Live included in Debian Squeeze (released 2/2011) was only TEX Live 2009 (instead of 2010), but in the releases since then (Wheezy and Jessie), the versions of TEX Live in Debian have been the latest releases.

6 Current status

Since about 2013 I am trying to keep a regular schedule of new TEX Live packages every month. These helps me to keep up with the changes in upstream packaging and reduces the load of packaging a new release of TEX Live. It also brings to users of unstable and testing a very up-to-date TEX system, where packages at most lag one month behind the TEX Live network updates.

7 Future

As most of the readers here know, besides caring for TEX (Live) and related packages in Debian, I am also responsible for the TEX Live Manager (`tlmgr`) and most of upstream's infrastructure including network distribution. Thus, my (spare, outside work) time needs to be distributed between all these projects (among others) which leaves less and less time for Debian packaging. Fortunately the packaging is in a state that makes regular updates once a month a light enough burden to accomplish, since most steps are automated. What remains a bit of a struggle is adapting the binary package (`src:texlive-bin` [10]) to new releases. But also this has become simpler due to less invasive changes over the years.

All in all, I don't have many plans for TEX Live in Debian besides keeping the current system running as it is. And this is in itself already a good reason to search for new contributors and maintainers!

8 Search for and advice to future maintainers and collaborators

I would be more than happy if new collaborators appear, with fresh ideas and some spare time. Unfortunately, my experience over these 10 years with people showing up and proposing changes (anyone remember the fellow proposing a complete rewrite in ML?) has been that nobody wants to invest serious time and energy, but merely searches for quick solutions. This is not something that will work with a package like TEX Live, with a size of several gigabytes (the biggest in the Debian archive), and complicated inner workings.

I advise everyone interested in helping to package TEX Live for Debian (or for that matter any other operating system distribution), to first install normal TEX Live from TUG, get used to what actions happen during updates (format rebuilds, hyphenation patterns, map file updates). One does not need to have a perfect understanding of what exactly happens down there in the guts (I didn't have in the beginning, either), but if you want to help with packaging but have never heard of format dumps or map files, this just might be a small obstacle.

9 Conclusion

TEX Live is the only TEX system in wide use across many hardware architectures and operating systems. The only comparable system, MiKTEX, is Windows-specific (although it contains some traces of ports to Unix). Backed by all the big user groups of TEX, TEX Live will remain the prime choice for the foreseeable future, and thus also TEX Live in Debian.

References

[1] `https://goo.gl/3EkZul`.

[2] `https://goo.gl/GeY5e7`.

[3] `https://goo.gl/oGb61o`.

[4] `https://goo.gl/6rR5bs`.

[5] `https://goo.gl/Hs4UkJ`.

[6] `https://goo.gl/SrKtkI`.

[7] `https://goo.gl/sz5BNj`.

[8] The teTEX home page. `http://tug.org/tetex/`.

[9] `CHANGES.packaging`. `http://goo.gl/ukVYCk`.

[10] `texlive-bin` source package on Debian QA. `https://goo.gl/MGmRd3`.

◇ Norbert Preining
Japan Advanced Institute of Science and Technology
Nomi, Ishikawa, Japan
`norbert (at) preining dot info`
`http://www.preining.info`

Paragraph designer with galley approach

Oleg Parashchenko

Abstract

The LaTeX package `paravesp.sty` controls the space above and below paragraphs.

The Python script `parades.py` generates paragraph styles with support of space above, space below and tabulators.

The system imposes the galley approach on the document.

1 Introduction

The goal was to support one layout specification defining the space above and below paragraphs. This is not how TeX works. To satisfy the requirement, the package `paravesp` (PARAgraph VErtical SPace) was developed.

The solution imposes the galley approach on the document. Paragraphs need to be wrapped by a tracking code, which controls how the material is added into the TeX vertical list.

The paragraph designer appeared as a generalization of the tracking code to other paragraph properties. The user describes the formatting options in a Python file. The program `parades.py` converts the definitions into TeX code.

The system works successfully in production, but so far is limited to my needs. A complete set of paragraph properties is not an immediate goal. Switching to the package `xgalley` from the LaTeX3 project might be a step in future development.

This article starts with the definition of the space between paragraphs and how it is implemented. The example demonstrates the use of the commands, which are then described using pseudocode.

The paragraph designer is first illustrated by a sample LaTeX fragment, which uses the paragraph styles. For each of the three types of styles, we give a sample definition in Python and the result of translating to TeX code, with explanations. Finally, a reference section lists all the supported paragraph properties and the commands of the Python `parades.py` tool.

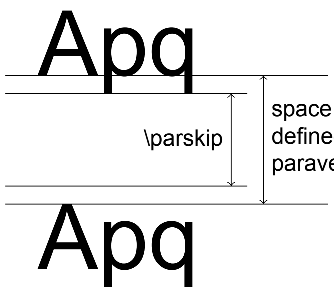
The article concludes with information on how to get the code and run it.

2 Space between paragraphs

The notion of "space between paragraphs" can be defined in various ways.

In one definition, the space between paragraphs is the amount of additional space relative to what happens inside a paragraph. This is what most typesetting engines implement, and what is named `parskip` in TeX.

The definition for `paravesp.sty` is: the space between paragraphs is the distance between the baseline of the preceding paragraph and the top of the next paragraph. The code ensures that this distance is larger than `prevdepth`.



2.1 Usage

The package `paravesp` imposes restrictions on how to construct a document. Otherwise it can't guarantee the desired space above or below paragraphs.

- Switches between the vertical and horizontal modes must be controlled. TeX's automatic switching is partially forbidden.
- The register `\parskip` belongs to the controlling code.
- The commands rely on the automatic insertion of `parskip` glue by TeX.

The guidelines for the controlling code are:

- At the end of a paragraph (after `\par`) use the command `\ParaSpaceBelow`.
- At the beginning of a paragraph, while still in the vertical mode, use `\ParaSpaceAbove`.
- At the beginning of block content, for which TeX will not insert `\parskip` automatically, use both `\ParaSpaceAbove` and `\IssueParaSpace`.

An example:

```
\ParaSpaceAbove{20pt}%
{\HeadingStyle Heading}\par
\ParaSpaceBelow{20pt}%
%
\ParaSpaceAbove{10pt}%
A paragraph of normal text...\par
\ParaSpaceBelow{10pt}%
%
\ParaSpaceAbove{10pt}%
Another paragraph of normal text...\par
\ParaSpaceBelow{10pt}%
%
\ParaSpaceAbove{20pt}\IssueParaSpace
\vbox{\fbox{Some info in a box}}%
\ParaSpaceBelow{20pt}%
```

2.2 Technical details

Below is a simplified version of what happens. Special cases are not shown.

After `\ParaSpaceBelow{`*length*`}`:

- vertical list is not changed
- `parskip` := *length* − `prevdepth`
- `prevdepth` is not changed

The command `\ParaSpaceBelow` splits its argument between two lengths, `prevdepth` and `parskip`. This is a precaution for the case when the next element in the vertical list is not controlled by the galley. Thanks to the retained `prevdepth`, a possible layout corruption is avoided.

After `\ParaSpaceAbove{`*length*`}`:

- vertical list: `vskip` −`prevdepth`,
  penalty as before `vskip`
- `parskip` := max(*length*, *old_ length*)
- `prevdepth` := −1000pt

The command `\ParaSpaceAbove`, which precedes a paragraph, can't know how much interline glue induced by `baselineskip` will be added. As a solution, the command disables this glue completely by setting `prevdepth` to minus infinity.

After `\IssueParaSpace`:

- vertical list: `vskip parskip`,
  penalty as before `vskip`
- `parskip` := 0pt
- `prevdepth` := −1000pt

You need the command `\IssueParaSpace` when TEX does not insert `\parskip` automatically, for example, before a box.

The command expects that it is called after `\ParaSpaceAbove`.

After `\IgnoreSpaceAboveNextPara`:

- vertical list is not changed
- `parskip` := −0.01pt
- `prevdepth` is not changed

The special case is `parskip` less than 0pt, which cancels the vertical spacing. It is useful when display content (image, list, etc.) is the first element inside a table cell.

3 Paragraph designer

The paragraph designer transforms Python objects with desired paragraph properties into TEX code which implements these properties.

The main benefit is that the paragraphs definitions can be constructed in such way that the repetitions (for example, font names) can be extracted into common settings.

The system proposes that every block-level element of a document should be wrapped into a command or an environment, which support the galley approach. The suggested sorts of the paragraphs:

- long body text paragraphs, wrapped by an environment,
- short paragraphs, wrapped by a command, and
- short paragraphs with tab stops, also wrapped by a command.

A document made using this approach looks structured. Here is an example.

```
\HeadI{Universal Declaration of Human Rights}
\HeadII{Preamble}
\begin{para}Whereas recognition...\end{para}
\begin{para}Whereas disregard
    and contempt...\end{para}
...
\HeadII{Article 14}
\begin{udhrlist}
\listitem{1}{Everyone has the right ...}
\listitem{2}{This right may not be invoked ...}
\end{udhrlist}
```

The sample is generated automatically from the XML source. The generation script, the paragraph styles as Python definition, the `.sty` code, and the PDF result are included in the package in the directory `example`.

3.1 Example: the command `\HeadI`

Commands are recommended for small paragraphs, such as headings and captions.

```
\HeadI{Universal Declaration of Human Rights}
```

A sample definition in Python:

```
add_style(ParagraphOptions(cmd='HeadI',
    space_above='20pt',
    space_below='20pt',
    fontsize='12pt', baseline='14pt',
    fontcmd=r'\fontseries{b}\selectfont',
    afterpar=r'\nobreak',
    ))
```

The properties of the paragraph are stored inside the object `ParagraphOptions`. As in many other programming languages, the backslash (\) is normally an escape character (not in the TEX sense!), and must be doubled inside strings (\\). An alternative in Python, as seen in the example here, is to prefix the string with `r`, which disables the escape.

The function `add_style` remembers the object in the global styles list. At the end of the Python script, the objects in the list are converted to TEX code.

The result of the conversion:

```
\newcommand{\HeadI}[1]{{%
\fontsize{12pt}{14pt}\fontseries{b}\selectfont%
\ParaSpaceAbove{20pt}%
\noindent #1\par}%
\nobreak\ParaSpaceBelow{20pt}}
```

The peculiarities are:

- The paragraph is created explicitly with `\noindent #1\par`.
- The text and the pre-paragraph settings are in a group. This way settings such as font changes affect only the given paragraph and not the rest of the document.

3.2 Example: the environment `para`

Environments are recommended for wrapping paragraphs in the text body.

```
\begin{para}Whereas recognition...\end{para}
\begin{para}Whereas disregard
    and contempt...\end{para}
```

A sample definition in Python:

```
add_style(ParagraphOptions(cmd='paracmd',
    env='para',
    space_above='10pt plus1pt minus1pt',
    ))
```

The result of the conversion, in a `.sty` file:

```
\newenvironment{para}{%
  \ParaSpaceAbove{10pt plus1pt minus1pt}%
  \noindent \ignorespaces}
{\par\global\def\pd@after@para{%
  \ParaSpaceBelow{0pt}}%
  \aftergroup\pd@after@para}
```

The paragraph is started explicitly with the command `\noindent`, followed by `\ignorespaces`, and finished, also explicitly, with `\par`.

The changes inside an environment, including post-paragraph settings, are again local and thus automatically discarded when the environment's group is finished. Therefore, using `\aftergroup`, the post-paragraph settings are applied after the end of the environment.

3.3 Example: tab stops in `listitem`

Paragraphs with tab stops are used to implement list items, captions, table of content entries and similar elements. The list paragraphs in the following example have one tab stop to store the list numbering.

```
\listitem{1}{Everyone has the right ...}
\listitem{2}{This right
    may not be invoked ...}
```

A sample definition in Python:

```
add_style(ParagraphOptions(cmd='listitem',
    moresetup='\\interlinepenalty=150\\relax',
    space_above='8pt',
    boxes=(('0cm', '0.5cm'),),
    leftskip='0.5cm'))
```

The argument `boxes` is a list of pairs. Each pair gives the offset of the tab stop from left and the width of the box. Due to peculiarities of Python, one-element lists of pairs need an extra comma inside.

The position of the paragraph text should be tuned manually to avoid overlapping with the tab stop boxes. In the example above, the left margin is set to `0.5cm` using `\leftskip`.

The result of the conversion, in a `.sty` file, is complicated:

```
\newcommand{\listitem}[2]{{%
  \ParaSpaceAbove{8pt}%
  \interlinepenalty=150\relax%
  \noindent \advance\pd@leftskip by 0.5cm %
  \hbox to 0pt{\hss\hbox to 0.5cm{#1\hss}%
    \dimen0=0.5cm %
    \advance\dimen0 by -0cm %
    \advance\dimen0 by -0.5cm \hskip\dimen0}%
    \the\everypar #2\par}%
  \ParaSpaceBelow{0pt}}
```

The skeleton of the list paragraph has these elements:

`\noindent` *tab stops* `\everypar` *text* `\par`

The use of `\noindent` and `\par` is clear. The paragraph starts with the tab stop boxes, therefore TeX does not insert `\everypar` automatically, therefore the code does it.

The token `\pd@leftskip` is a `\let`-synonym for `\leftskip`. In a right-to-left document one would set the token to `\rightskip`.

A tab stop is constructed from two nested boxes. The inner box gives the width of the tab stop and aligns the content to the left:

`\hbox to` *width*`{`*content* `\hss}`

The outer box puts the inner box at the specified offset.

```
\hbox to 0pt{\hss inner_box%
\dimen0=leftskip
\advance\dimen0 by -offset
\advance\dimen0 by -width
\hskip\dimen0}%
```

The calculation is not obvious. The illustration in figure 1 provides the source for it.

The image reflects how the boxes, glues and lengths are related. We see that `offset`+`width`+`x` is `leftskip`, therefore `x` (`\dimen0`) is `leftskip` minus `offset` minus `width`.

4 Paragraph designer reference

Denomination: `cmd`, `env`, `stylecmd`. These are the names for the generated commands and environments.

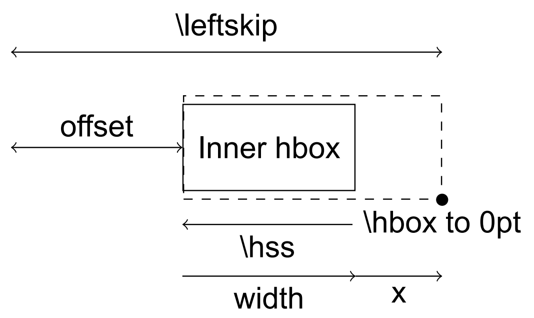


Figure 1: Calculation for tab stops.

Examples of `cmd` and `env` have already been given. The command for `stylecmd` makes a character style, which affects the font and does not set the paragraph properties (vertical spacing, tabulars, etc.).

A sample paragraph definition:

```
ParagraphOptions(cmd="Caption,
  stylecmd="UseCaption", ...)
```

In a LaTeX document you could then write:

```
{\UseCaption Article 1.} All human beings
are born free and equal in dignity ...
```

All the three denominators can be mixed together at once. You must specify `cmd` even if you don't need it.

Fonts: `fontsize`, `baseline`, `fontcmd`.

The only supported font properties are its size and line spacing. The other properties, such as width and series, need to be manually defined in `fontcmd`:

```
ParagraphOptions(...,
    fontcmd=r'\fontseries{b}\selectfont',
    ...)
```

Dimensions: `leftskip`, `hsize`, `space_above` and `space_below`.

The names are self-explanatory.

The default value for both `space_above` and `space_below` is `0pt`. This means that if you haven't given a value, then two consecutive paragraphs will touch each other, as if `\nointerlineskip` were given between them.

Use the special value `#natural` to disable the use of `\ParaSpaceAbove` or `\ParaSpaceBelow` and instead restore the default TeX behaviour.

```
ParagraphOptions(...,
  space_above='#natural',
  space_below='#natural', ...)
```

Tuning: `moresetup`, `afterpar`, `preamble_arg1`, `preamble_arg2`, `preamble_arg3`, `preamble_arg4`.

The content of `moresetup` is literally copied into the style definition at the end of the paragraph setup, just before `\noindent`. A few ideas what can be set in `moresetup`:

- A color for the paragraph text,
- `\penalty` to suggest a page break,
- `\interlinepenalty` for list item paragraphs, to avoid a page breaks inside.

The content of `afterpar` is literally copied into the style definition directly after `{...\par}`. This is a good place to put `\nobreak` or some other penalty.

The content of `preamble_argN` is copied literally into the style definition directly before `#N`. Possible applications:

- Add `\ignorespaces` if the text might contain spurious spaces at the beginning.
- For list item paragraphs, `\hfil` centers the tab box content, `\hfill` aligns to the right.

Tab stops. Tab stops are hboxes of a given width at given offset. All the offsets are relative to the left border of the text flow.

```
ParagraphOptions(...,
  boxes=(
    (offset1,width1),
    (offset2,width2),
    ...,
    (offset_n,width_n)),
  ...)
```

Due to Python peculiarities, a one-element list of lists needs an additional comma, otherwise Python unwraps one level of parentheses. Thus, the correct way is:

```
ParagraphOptions(...,
  boxes=((offset,width),), # comma inside
  ...)
```

The content of the boxes is left-aligned. To center or right-align the content, add `\hfil` or `\hfill` through the parameter `preamble_argN`.

Inheritance. The parameter `parent` uses an existing paragraph object as the starting point for the paragraph being defined. Properties not specified in the new paragraph definition are taken from the parent.

```
head_i = ParagraphOptions(
    cmd='HeadI',
    fontsize='12pt', baseline='14pt',
    fontcmd=r'\fontseries{b}\selectfont',
    ... )

ParagraphOptions(cmd='HeadII',
    parent=head_i, # Inheritance
    fontsize='11pt', baseline='13pt',
    ... )
```

In the example, the paragraph `HeadII` inherits `fontcmd` from `HeadI`, but uses the custom `fontsize` and `baseline` settings.

The infrastructure. A Python file with definitions: (1) starts by importing the support code; (2) continues with collecting the definitions; and (3) finishes with the command to dump the TEX result.

```
from parades import * # (1)

add_style(ParagraphOptions(...)) # (2)
add_style(ParagraphOptions(...))
...
add_style(ParagraphOptions(...))

main('paras') # (3)
```

The parameter of the function `main` (in this example `paras`) is the name of the generated sty-package as given by `\ProvidesPackage`.

5 Getting and running the code

All the files, including the example, are contained in the CTAN package `parades` (`http://ctan.org/pkg/parades`). Alternatively, you can get the source code from `github` in the repository `http://github.com/olpa/tex`, in the folder `paragraph_designer`.

Put the file `paravesp.sty` into a directory in which TEX will find it. Put the file `parades.py` into a directory in which Python will find it.

The paragraph generator runs from the command line.

```
$ python input-defs.py [output-defs.sty]
```

The script `input-defs.py` is the file with the Python definitions of the paragraphs. The optional argument is the name of a `.sty` file with the generated TEX definitions. If the output file is not specified, the code is dumped to the standard output.

The directory `example` contains a sample project. Refer to the file `README` in this directory for details how to use it.

6 Conclusion

The paragraph designer helps both on the technical and organization levels. On the technical level, it helps generating code for paragraph styles. It would be an unpleasant and error-prone task to write this code manually:

- Space above and below a paragraph.
- Paragraphs with tab stops such as list items, table of content entries, headers.

On the organizational level, the Python scripts allow one to have a common code base and adapt it to the needs of specific layouts.

The LATEX package `paravesp` can be used independently of the paragraph designer to implement vertical spacing.

There are problems with the package `paravesp` and the paragraph designer:

- Many features are not implemented and some need rework.
- The LATEX code written in the galley style is too verbose to be typeset manually.

The paragraph designer has been used in a production system for years. Thus the benefits can outweigh the problems.

◇ Oleg Parashchenko
bitplant.de GmbH
Fabrikstr. 15
89520 Heidenheim, Germany
`olpa (at) uucode dot com`
`http://uucode.com/`

LuaTeX 0.90 backend changes for PDF and more

Hans Hagen

Abstract

LuaTeX 0.90 brings a sweeping reorganization of the backend, especially PDF-related features. Many fundamental primitives, both commands and variables, have been renamed and generalized. Some primitives have been removed from the TeX interface, but can be made available through Lua. Internally, some uses of whatsits have been converted to regular nodes.

1 Introduction

The original design of TeX has a clear separation between the frontend and backend code. In principle, shipping out a page boils down to traversing the to-be-shipped-out box and translating the glyph, rule, glue, kern and list nodes into positioning just glyphs and rules on a canvas. The DVI backend is therefore relatively simple, as the DVI output format delegates to other programs the details of font inclusion and such into the final format; it just describes the pages.

Because we eventually want color and images as well, there is a mechanism to pass additional information to post-processing programs. One can insert `\special`s with directives like `insert image named foo.jpg`. The frontend as well as the backend are not concerned with what goes into a special; the DVI post-processor of course is.

The PDF backend, on the other hand, is more complex as it immediately produces the final typeset result and, as such, offers possibilities to insert verbatim code (`\pdfliteral`), images (`\pdfximage` cum suis), annotations, destinations, threads and all kinds of objects, reuse typeset content (`\pdfxform` cum suis); in the end, there are all kinds of `\pdf...` commands. The way these were implemented in LuaTeX prior to 0.82 violates the separation between frontend and backend, an inheritance from pdfTeX. Additional features such as protrusion and expansion add to that entanglement. However, because PDF is an evolving standard, occasionally we need to adapt the related code. A separation of code makes sure that the frontend can become stable (and hopefully frozen) at some point.[1]

In LuaTeX we had already started making this separation of specialized code, such as a cleaner implementation of font expansion, but all these `\pdf...` commands were still pervasive, leading to fuzzy dependencies, checks for backend modes, etc. so a logical step was to straighten all this out. That way we give LuaTeX a cleaner core constructed from traditional TeX, extended with ε-TeX, Aleph/Omega, and LuaTeX functionality.

Author's note: Thanks to Alan Braslau and Karl Berry for text corrections.

[1] In practice nowadays, the backend code changes little, because the PDF produced by LuaTeX is rather simple and is easily adapted to the changing standard.

2 Extensions

A first step, then, was to transform generic (i.e. independent from the backend) functionality which was still (sort of) bound to Aleph and pdfTeX, into core functionality. A second step was to reorganize the backend specific PDF code, i.e. move it out of the core and into the group of extension commands. This extension group is somewhat special and originates in traditional TeX; it is the way to add your own functionality to TeX, the program.

As an example for future programmers, Don Knuth added four (connected) primitives as extensions: `\openout`, `\closeout`, `\write` and `\special`. The Aleph and pdfTeX engines, on the other hand, put some functionality in extensions and some in the core. This arose from the fact that dealing with variables in extensions is often inconvenient, as they are then seen as (unexpandable) commands instead of integers, token lists, etc. That the write-related commands are there is almost entirely due to being the demonstration of the mechanism; everything related to *reading* files is in the core. There is one property that perhaps forces us to keep the writers there, and that's the `\immediate` prefix.[2]

In the process of separating, we reshuffled the code base a bit; the current use of the extensions mechanism still suits as an example and also gives us backward compatibility. However, new backend primitives will not be added there but rather in specific plugins (if needed at all).

3 From whatsits to nodes: images, forms, directions

The PDF backend introduced two new concepts into the core: (reusable) images and (reusable) content (wrapped in boxes). In keeping with good TeX practice, these were implemented as whatsits (a node type for extensions); but this created, as a side effect, an anomaly in the handling of such nodes. Consider looping over a node list where we need to check dimensions of nodes; in Lua, we can write something like this:

```
while n do
    if n.id == glyph then
```

[2] Unfortunately we're stuck with `\immediate` in the backend; a `deferred` keyword would have been handier, especially since other backend-related commands can also be immediate.

```
        -- wd ht dp
    elseif n.id == rule then
        -- wd ht dp
    elseif n.id == kern then
        -- wd
    elseif n.id == glue then
        -- size stretch shrink
    elseif n.id == whatsits then
        if n.subtype == pdfxform then
            -- wd ht dp
        elseif n.subtype == pdfximage then
            -- wd ht dp
        end
    end
    n = n.next
end
```

So for each node in the list, we need to check these two whatsit subtypes. But as these two concepts are rather generic, there is no evident need to implement it this way. Of course the backend has to provide the inclusion and reuse, but the frontend can be agnostic about this. That is, at the input end, in specifying these two injects, we only have to make sure we pass the right information (so the scanner might differentiate between backends).

Thus, in LuaTEX these two concepts have been promoted to core features:

```
\pdfxform           \saveboxresource
\pdfximage          \saveimageresource
\pdfrefxform        \useboxresource
\pdfrefximage       \useimageresource
\pdflastxform       \lastsavedboxresourceindex
\pdflastximage      \lastsavedimageresourceindex
\pdflastximagepages \lastsavedimageresourcepages
```

The index should be considered an arbitrary number set to whatever the backend plugin decides to use as an identifier. These are no longer whatsits, but a special type of rule; after all, TEX is only interested in dimensions. Given this change, the previous code can be simplified to:

```
while n do
    if n.id == glyph then
        -- wd ht dp
    elseif n.id == rule then
        -- wd ht dp
    elseif n.id == kern then
        -- wd
    elseif n.id == glue then
        -- size stretch shrink
    end
    n = n.next
end
```

The only consequence for the previously existing rule type (which, in fact, is also something that needs to be dealt with in the backend, depending on the target format) is that a normal rule now has subtype 0 while the box resource has subtype 1 and the image subtype 2.

If a package writer wants to retain the pdfTEX names, the previous table can be used; just prefix `\let`. For example, the first line would be (spaces optional, of course):

```
\let \pdfxform \saveboxresource
```

3.1 Direction nodes

A similar change has been made for "direction" nodes, which were also previously whatsits. These are now normal nodes so again, instead of consulting whatsit subtypes, we can now just check the id of a node.

It should be apparent that all of these changes from whatsits to normal nodes already greatly simplify the code base.

4 Commands promoted to the core

Many more commands have been promoted to the core. Here is an additional list of original pdfTEX commands and their new counterparts (this time with the `\let` included):

```
\let\pdfpagewidth        \pagewidth
\let\pdfpageheight       \pageheight

\let\pdfadjustspacing    \adjustspacing
\let\pdfprotrudechars    \protrudechars
\let\pdfnoligatures      \ignoreligaturesinfont
\let\pdffontexpand       \expandglyphsinfont
\let\pdfcopyfont         \copyfont

\let\pdfnormaldeviate  \normaldeviate
\let\pdfuniformdeviate \uniformdeviate
\let\pdfsetrandomseed  \setrandomseed
\let\pdfrandomseed     \randomseed

\let\ifpdfabsnum       \ifabsnum
\let\ifpdfabsdim       \ifabsdim
\let\ifpdfprimitive    \ifprimitive

\let\pdfprimitive      \primitive

\let\pdfsavepos        \savepos
\let\pdflastxpos       \lastxpos
\let\pdflastypos       \lastypos

\let\pdftexversion     \luatexversion
\let\pdftexrevision    \luatexrevision
\let\pdftexbanner      \luatexbanner

\let\pdfoutput         \outputmode
\let\pdfdraftmode      \draftmode

\let\pdfpxdimen        \pxdimen

\let\pdfinsertht       \insertht
```

```
\protected\def\pdfliteral        {\pdfextension literal }
\protected\def\pdfcolorstack     {\pdfextension colorstack }
\protected\def\pdfsetmatrix      {\pdfextension setmatrix }
\protected\def\pdfsave           {\pdfextension save\relax}
\protected\def\pdfrestore        {\pdfextension restore\relax}
\protected\def\pdfobj            {\pdfextension obj }
\protected\def\pdfrefobj         {\pdfextension refobj }
\protected\def\pdfannot          {\pdfextension annot }

\protected\def\pdfstartlink      {\pdfextension startlink }
\protected\def\pdfendlink        {\pdfextension endlink\relax}
\protected\def\pdfoutline        {\pdfextension outline }
\protected\def\pdfdest           {\pdfextension dest }
\protected\def\pdfthread         {\pdfextension thread }
\protected\def\pdfstartthread    {\pdfextension startthread }
\protected\def\pdfendthread      {\pdfextension endthread\relax}

\protected\def\pdfinfo           {\pdfextension info }
\protected\def\pdfcatalog        {\pdfextension catalog }
\protected\def\pdfnames          {\pdfextension names }
\protected\def\pdfincludechars   {\pdfextension includechars }
\protected\def\pdffontattr       {\pdfextension fontattr }
\protected\def\pdfmapfile        {\pdfextension mapfile }
\protected\def\pdfmapline        {\pdfextension mapline }
\protected\def\pdftrailer        {\pdfextension trailer }
\protected\def\pdfglyphtounicode {\pdfextension glyphtounicode }
```

Table 1: List of pdfTEX commands and their new `\pdfextension` equivalents in LuaTEX.

5 Commands: from `\pdf...` to `\pdfextension`

There are many commands that start with `\pdf` and, over the history of development of pdfTEX and LuaTEX, some have been added, some have been renamed, others removed. Instead of the many, we now have just one: `\pdfextension`. A couple of usage examples:

```
\pdfextension literal {1 0 0 2 0 0 cm}
\pdfextension obj     {/foo (bar)}
```

Here, we pass a keyword that tells for what to scan and what to do with it. A backward-compatible interface is easy to write. Although it delegates a bit more management of these `\pdf` commands to the macro package, the responsibility for dealing with such low-level, error-prone calls is there anyway. The full list of `\pdfextension`s is given in table 1. The scanning after the keyword is the same as for pdfTEX.

6 Variables: from `\pdf...` to `\pdfvariable`

As with commands, there are many variables that can influence the PDF backend. The most important one was, of course, that which set the output mode (`\pdfoutput`). Well, that one is gone and has been replaced by `\outputmode`. A value of 1 means that we produce PDF.

One complication of variables is that (if we want to be compatible), we need to have them as real TEX registers. However, as most of them are optional, an easy way out is simply not to define them in the engine. In order to be able to still deal with them as registers (which is backward compatible), we define them as shown in table 2.

You can set them as follows (the values shown here are the initial values):

`\pdfcompresslevel`	9
`\pdfobjcompresslevel`	1
`\pdfdecimaldigits`	3
`\pdfgamma`	1000
`\pdfimageresolution`	71
`\pdfimageapplygamma`	0
`\pdfimagegamma`	2200
`\pdfimagehicolor`	1
`\pdfimageaddfilename`	1
`\pdfpkresolution`	72
`\pdfinclusioncopyfonts`	0
`\pdfinclusionerrorlevel`	0
`\pdfignoreunknownimages`	0
`\pdfreplacefont`	0
`\pdfgentounicode`	0
`\pdfpagebox`	0
`\pdfminorversion`	4
`\pdfuniqueresname`	0

```
\edef\pdfminorversion        {\pdfvariable minorversion}
\edef\pdfcompresslevel       {\pdfvariable compresslevel}
\edef\pdfobjcompresslevel    {\pdfvariable objcompresslevel}
\edef\pdfdecimaldigits       {\pdfvariable decimaldigits}

\edef\pdfhorigin             {\pdfvariable horigin}
\edef\pdfvorigin             {\pdfvariable vorigin}

\edef\pdfgamma               {\pdfvariable gamma}
\edef\pdfimageresolution     {\pdfvariable imageresolution}
\edef\pdfimageapplygamma     {\pdfvariable imageapplygamma}
\edef\pdfimagegamma          {\pdfvariable imagegamma}
\edef\pdfimagehicolor        {\pdfvariable imagehicolor}
\edef\pdfimageaddfilename    {\pdfvariable imageaddfilename}
\edef\pdfignoreunknownimages {\pdfvariable ignoreunknownimages}

\edef\pdfinclusioncopyfonts  {\pdfvariable inclusioncopyfonts}
\edef\pdfinclusionerrorlevel {\pdfvariable inclusionerrorlevel}
\edef\pdfpkmode              {\pdfvariable pkmode}
\edef\pdfpkresolution        {\pdfvariable pkresolution}
\edef\pdfgentounicode        {\pdfvariable gentounicode}

\edef\pdflinkmargin          {\pdfvariable linkmargin}
\edef\pdfdestmargin          {\pdfvariable destmargin}
\edef\pdfthreadmargin        {\pdfvariable threadmargin}
\edef\pdfformmargin          {\pdfvariable formmargin}

\edef\pdfuniqueresname       {\pdfvariable uniqueresname}
\edef\pdfpagebox             {\pdfvariable pagebox}
\edef\pdfpagesattr           {\pdfvariable pagesattr}
\edef\pdfpageattr            {\pdfvariable pageattr}
\edef\pdfpageresources       {\pdfvariable pageresources}
\edef\pdfxformattr           {\pdfvariable xformattr}
\edef\pdfxformresources      {\pdfvariable xformresources}
```

Table 2: List of pdfTEX variables and their new `\pdfvariable` equivalents in LuaTEX.

```
\pdfhorigin          1in
\pdfvorigin          1in
\pdflinkmargin       0pt
\pdfdestmargin       0pt
\pdfthreadmargin     0pt
```

Their removal from the frontend has helped again to clean up the code and, by making them registers, their use is still compatible. A call to `\pdfvariable` defines an internal register that keeps the value (of course this value can also be influenced by the backend itself). Although they are real registers, they live in a protected namespace:

```
\meaning\pdfcompresslevel
```

which gives:

```
macro:->[internal backend integer]
```

It's perhaps unfortunate that we have to remain compatible because a setter and getter would be much nicer. I am still considering writing the extension primitive in Lua using the token scanner, but it might not be possible to remain compatible then. This is not so much an issue for ConTEXt that always has had backend drivers, but, rather, for other macro packages that have users expecting the primitives (or counterparts) to be available.

7 Read-only variables: from `\pdf...` to `\pdffeedback`

The backend can report on some properties that were also accessible via `\pdf...` primitives. Because these are read-only variables, another primitive now handles them: `\pdffeedback`. This primitive can be used to define compatible alternatives, as shown in table 3.

The variables are internal, so they are anonymous. When we ask for the meaning of some that were previously defined:

```
\meaning\pdfhorigin
\meaning\pdfcompresslevel
\meaning\pdfpageattr
```

```
\def\pdfcolorstackinit {\pdffeedback colorstackinit}
\def\pdfcreationdate   {\pdffeedback creationdate}
\def\pdffontname       {\numexpr\pdffeedback fontname\relax}
\def\pdffontobjnum     {\numexpr\pdffeedback fontobjnum\relax}
\def\pdffontsize       {\dimexpr\pdffeedback fontsize\relax}
\def\pdflastannot      {\numexpr\pdffeedback lastannot\relax}
\def\pdflastlink       {\numexpr\pdffeedback lastlink\relax}
\def\pdflastobj        {\numexpr\pdffeedback lastobj\relax}
\def\pdfpageref        {\numexpr\pdffeedback pageref\relax}
\def\pdfretval         {\numexpr\pdffeedback retval\relax}
\def\pdfxformname      {\numexpr\pdffeedback xformname\relax}
```

Table 3: List of read-only pdfTEX variables and their new `\pdffeedback` equivalents in LuaTEX.

we will get, similar to the above:

```
macro:->[internal backend dimension]
macro:->[internal backend integer]
macro:->[internal backend tokenlist]
```

8 \pdf... primitives removed

Finally, here is the list of primitives that have been removed, with no TEX-level equivalent available. Many were experimental, and they can be easily be provided to TEX using Lua.

```
\knaccode
\knbccode
\knbscode
\pdfadjustinterwordglue
\pdfappendkern
\pdfeachlinedepth
\pdfeachlineheight
\pdfelapsedtime
\pdfescapehex
\pdfescapename
\pdfescapestring
\pdffiledump
\pdffilemoddate
\pdffilesize
\pdffirstlineheight
\pdfforcepagebox
\pdfignoreddimen
\pdflastlinedepth
\pdflastmatch
\pdflastximagecolordepth
\pdfmatch
\pdfmdfivesum
\pdfmovechars
\pdfoptionalwaysusepdfpagebox
\pdfoptionpdfinclusionerrorlevel
\pdfprependkern
\pdfresettimer
\pdfshellescape
\pdfsnaprefpoint
\pdfsnapy
\pdfsnapycomp
\pdfstrcmp
\pdfunescapehex
\pdfximagebbox
\shbscode
\stbscode
```

9 Conclusion

The advantage of a clean backend separation, supported by just the three primitives `\pdfextension`, `\pdfvariable` and `\pdffeedback`, as well as a collection of registers, is that we can now further clean the code base, which remains a curious mix of combined engine code, sometimes and sometimes not converted to C from Pascal. A clean separation also means that if someone wants to tune the backend for a special purpose, the frontend can be left untouched. We will get there eventually.

All the definitions shown here are available in the file `luatex-pdf.tex`, which is part of the ConTEXt distribution.

◇ Hans Hagen
Pragma ADE
http://pragma-ade.com

Still expanding LuaTEX: Possibly useful extensions

Hans Hagen

Abstract

New LuaTEX programming features in a variety of areas: rules, spaces, token lists, active characters, `\csname`, packing of lists, and error handling.

1 Introduction

While working on LuaTEX, it is tempting to introduce all kinds of new fancy programming features. Arguments for doing this can be characterized by descriptions like 'handy', 'speedup', 'less code', 'necessity'. It must be stated that traditional TEX is rather complete, and one can do quite a lot of macro magic to achieve many goals. So let us look a bit more at the validity of these arguments.

The 'handy' argument is in fact a valid one. Of course, one can always wrap clumsy code in a macro to hide the dirty tricks, but, still, it would be nicer to avoid needing to employ extremely dirty tricks. I found myself looking at old code wondering why something has to be done in such a complex way, only to realize, after a while, that it comes with the concept; one can get accustomed to it. After all, every programming language has its stronger and weaker aspects.

The 'speedup' argument is theoretically a good one too, but, in practice, it's hard to prove that a speedup really occurs. Say we save 5% on a job. This is nice for multipass on a server where many jobs run at the same time or after each other, but a little bit of clever macro coding will easily gain much more. Or, as we often see: sloppy macro or style writing will easily negate those gains. Another pitfall is that you can measure (say) half a million calls to a macro can indeed be brought down to a fraction of its runtime thanks to some helper, but, in practice, you will not see that gain because saving 0.1 seconds on a 10 second run can be neglected. Furthermore, adding a single page to the document will already make such a gain invisible to the user as that will itself increase the runtime. Of course, many small speedups can eventually accumulate to yield a significant overall gain, but, if the macro package is already quite optimized, it might not be easy to squeeze out much more. At least in ConTEXt, I find it hard to locate bottlenecks that could benefit from extensions, unless one adds very specific features, which is not what we want.

Of course one can create 'less' code by using more wrappers. But this can definitely have a speed penalty, so this argument should be used with care. An appropriate extra helper can make wrappers fast and the fewer helpers the better. The danger is in choosing what helpers. A good criterion is that it should be hard otherwise in TEX. Adding more primitives (and overhead) merely because some macro package would like it would be bad practice. I'm confident that helpers for ConTEXt would not be that useful for plain TEX, LATEX, etc., and vice versa.

The 'necessity' argument is a strong one. Many already present extensions from ε-TEX fall into this category: fully expandable expressions (although the implementation is somewhat restricted), better macro protection, expansion control, and the ability to test for a so-called csname (control sequence name) are examples.

In the end, the only valid argument is 'it can't be done otherwise', which is a combination of all these arguments with 'necessity' being dominant. This is why in LuaTEX there are not that many extensions to the language (nor will there be). I must admit that even after years of working with TEX, the number of wishes for more facilities is not that large.

The extensions in LuaTEX, compared to traditional TEX, can be summarized as follows:

- Of course we have the ε-TEX extensions, and these already have a long tradition of proven usage. We did remove the limited directional support.
- From Aleph (follow-up on Omega), part of the directional support and some font support was inherited.
- From pdfTEX, we took most of the backend code, but it has been improved in the meantime. We also took the protrusion and expansion code, but especially the latter has been implemented a bit differently (in the frontend as well as in the backend).
- Some handy extensions from pdfTEX have been generalized; other obscure or specialized ones have been removed. So we now have frontend support for position tracking, resources (images) and reusable content in the core. The backend code has been separated a bit better and only a few backend-related primitives remain.
- The input encoding is now UTF-8, exclusively, but one can easily hook in code to preprocess data that enters TEX's parser using Lua. The characteristic catcode settings for TEX can be grouped and switched efficiently.
- The font machinery has been opened wide so that we can use the embedded Lua interpreter to implement any technology that we might

want, with the usual control that TEXies like. Some further limitations have been lifted. One interesting point is that one can now construct virtual fonts at runtime.

- Ligature construction, kerning and paragraph building have been separated as a side effect of Lua control. There are some extensions in that area. For instance, we store the language and min/max values in the glyph nodes, and we also store penalties with discretionaries. Patterns can be loaded at runtime, and character codes that influence hyphenation can be manipulated.
- The math renderer has been upgraded to support OpenType math. This has resulted in many new primitives and extensions, not only to define characters and spacing, but also to control placement of superscripts and subscripts and generally to influence the way things are constructed. A couple of mechanisms have gained control options.
- Several Lua interfaces are available making it possible to manipulate the (intermediate) results. One can pipe text to TEX, write parsers, mess with node lists, inspect attributes assigned at the TEX end, etc.

Some of the features mentioned above are rather LuaTEX specific, such as catcode tables and attributes. They are present as they permit more advanced Lua interfacing. Other features, such as UTF-8 and OpenType math, are a side effect of more modern techniques. Bidirectional support is there because it was one of the original reasons for going forward with LuaTEX. The removal of backend primitives and thereby separating the code in a better way (see companion article) comes from the desire to get closer to the traditional core, so that most documentation by Don Knuth still applies. It's also the reason why we still speak of 'tokens', 'nodes' and 'noads'.

In the following sections I will discuss a few new low-level primitives. This is not a complete description (after all, we have reported on much already), and one can consult the LuaTEX manual to get the complete picture. The extensions described below are also relatively new and date from around version 0.85, the prelude to the stable version 1 release.

2 Rules

For insiders, it is no secret that TEX has no graphic capabilities, apart from the ability to draw rules. But with rules you can do quite a lot already. Add to that the possibility to insert arbitrary graphics or even backend drawing directives, and the average user won't notice that it's not true core functionality.

When we started with LuaTEX, we used code from pdfTEX and Omega (Aleph), and, as a consequence, we ended up with many whatsits. Normal running text has characters, kerns, some glue, maybe boxes, all represented by a limited set of so-called nodes. A whatsit is a kind of escape as it can be anything an extension to TEX needs to wrap up and put in the current list. Examples are (in traditional TEX already) whatsits that write to file (using `\write`) and whatsits that inject code into the backend (using `\special`). The directional mechanism of Omega uses whatsits to indicate direction changes.

For a long time images were also included using whatsits, and basically one had to reserve the right amount of space and inject a whatsit with a directive for the backend to inject something there with given dimensions or scale. Of course, one then needs methods to figure out the image properties, but, in the end, all of this could be done rather easily.

In pdfTEX, two new whatsits were introduced: images and reusable so-called forms, and, contrary to other whatsits, these do have dimensions. As a result, suddenly the TEX code base could no longer just ignore whatsits, but it had to check for these two when dimensions were important, for instance in the paragraph builder, packager, and backend.

So what has this to do with rules? Well, in LuaTEX all the whatsits are now back to where they belong, in the backend extension code. Directions are now first-class nodes, and we have native resources and reusable boxes. These resources and boxes are an abstraction of the pdfTEX images and forms, and, internally, they are a special kind of rule (i.e. a blob with dimensions). Because checking for rules is part of the (traditional) TEX kernel, we could simply remove the special whatsit code and let existing rule-related code do the job. This simplified the code a lot.

Because we suddenly had two more types of rules, we took the opportunity to add a few more.

```
\nohrule width 10cm height 2cm depth 0cm
\novrule width 10cm height 2cm depth 0cm
```

This is a way to reserve space, and it's nearly equivalent to the following (respectively):

```
{\setbox0\hbox{}%
 \wd0=10cm\ht0=2cm\dp0=0cm\box0\relax}
{\setbox0\vbox{}%
 \wd0=10cm\ht0=2cm\dp0=0cm\box0\relax}
```

There is no real gain in efficiency because keywords also take time to parse, but the advantage is that no Lua callbacks are triggered.[1] Of course, this variant would not have been introduced had we still had just rules and no further subtypes; it was just a rather trivial extension that fit in the repertoire.[2]

So, while we were at it, yet another rule type was introduced, but this one has been made available only in Lua. As this text is about LuaTEX, a bit of Lua code does fit into the discussion, so here we go. The code shown here is rather generic and looks somewhat different in ConTEXt, but it does the job.

First, let's create a straightforward rectangle drawing routine. We initialize some variables first, then scan properties using the token scanner, and, finally, we construct the rectangle using four rules. The packaged (so-called) hlist is written to TEX.

```
\startluacode
function FramedRule()
    local width     = 0
    local height    = 0
    local depth     = 0
    local linewidth = 0
    --
    while true do
        if token.scan_keyword("width") then
            width = token.scan_dimen()
        elseif token.scan_keyword("height") then
            height = token.scan_dimen()
        elseif token.scan_keyword("depth") then
            depth = token.scan_dimen()
        elseif token.scan_keyword("line") then
            linewidth = token.scan_dimen()
        else
            break
        end
    end
    local doublelinewidth = 2*linewidth
    --
    local left    = node.new("rule")
    local bottom  = node.new("rule")
    local right   = node.new("rule")
    local top     = node.new("rule")
    local back    = node.new("kern")
    local list    = node.new("hlist")
    --
    left.width    = linewidth
    bottom.width  = width - doublelinewidth
    bottom.height = -depth + linewidth
    bottom.depth  = depth
    right.width   = linewidth
    top.width     = width - doublelinewidth
    top.height    = height
    top.depth     = -height + linewidth
    back.kern     = -width + linewidth
    list.list     = left
    list.width    = width
    list.height   = height
    list.depth    = depth
    list.dir      = "TLT"
    --
    node.insert_after(left,left,bottom)
    node.insert_after(left,bottom,right)
    node.insert_after(left,right,back)
    node.insert_after(left,back,top)
    --
    node.write(list)
 end
\stopluacode
```

This function can be wrapped in a macro:

```
\def\FrameRule{\directlua{FramedRule()}}
```

and the macro can be used as follows:

```
\FrameRule width2cm height.5cm depth.5cm line2pt
```

The result is:

A different approach follows. Again, we define a rule, but, this time we only set dimensions and assign some attributes to it. Normally, one would reserve some attribute numbers for this purpose, but, for our example here, high numbers are safe enough. Now there is no need to wrap the rule in a box.

```
\startluacode
function FramedRule()
    local width     = 0
    local height    = 0
    local depth     = 0
    local linewidth = 0
    local radius    = 0
    local type      = 0
    --
    while true do
        if token.scan_keyword("width") then
            width = token.scan_dimen()
        elseif token.scan_keyword("height") then
            height = token.scan_dimen()
        elseif token.scan_keyword("depth") then
            depth = token.scan_dimen()
        elseif token.scan_keyword("line") then
            linewidth = token.scan_dimen()
        elseif token.scan_keyword("type") then
            type = token.scan_int()
        elseif token.scan_keyword("radius") then
            radius = token.scan_dimen()
        else
            break
        end
    end
    --
    local r   = node.new("rule")
    r.width   = width
    r.height  = height
```

[1] I still am considering adding variants of `\hbox` and `\vbox` where no callback would be triggered.

[2] This is one of the things I wanted to have for a long time but seems less useful today.

```
    r.depth   = depth
    r.subtype = 4    -- user rule
    r[20000]  = type
    r[20001]  = linewidth
    r[20002]  = radius or 0
    node.write(r)
end
\stopluacode
```

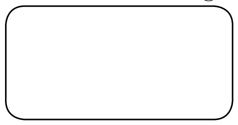
Nodes with subtype 4 (user) are intercepted and passed to a callback function, when set. Here we show a possible implementation:

```
\startluacode
local bpfactor = (7200/7227)/65536
local f_rectangle = "%f w 0 0 %f %f re %s"
local f_radtangle = [[
    %f w %f 0 m
    %f 0 l %f %f %f %f y
    %f %f l %f %f %f %f y
    %f %f l %f %f %f %f y
    %f %f l %f %f %f %f y
    h %s
]]

callback.register("process_rule",function(n,h,v)
    local t = n[20000] == 0 and "f" or "s"
    local l = n[20001] * bpfactor -- linewidth
    local r = n[20002] * bpfactor -- radius
    local w = h * bpfactor
    local h = v * bpfactor
    if r > 0 then
      p = string.format(f_radtangle,
          l, r, w-r, w,0,w,r, w,h-r, w,h,w-r,h,
          r,h, 0,h,0,h-r, 0,r, 0,0,r,0, t)
    else
      p = string.format(f_rectangle, l, w, h, t)
    end
    pdf.print("direct",p)
end)
\stopluacode
```

We can now also specify a radius and type, where 0 is a filled and 1 a stroked shape.

```
\FrameRule
    type  1
    width 3cm    height 1cm   depth  5mm
    line  0.2mm  radius 2.5mm
```

Since we specified a radius we get round corners:

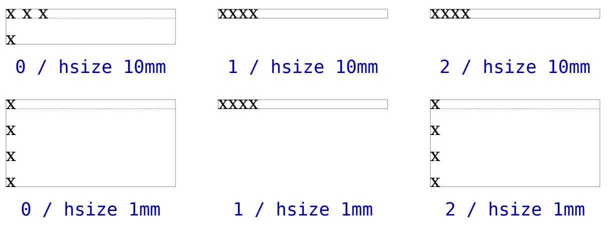


The nice thing about these extensions to rules is that the internals of TEX are not affected much. Rules are just blobs with dimensions and the par builder, for instance, doesn't care what they are. There is no need for further inspection. Maybe future versions of LuaTEX will provide more useful subtypes.

3 Spaces

Multiple successive spaces in TEX are normally collapsed into one. But, what if you don't want any spaces at all? It turns out this is rather hard to achieve. You can, of course, change the catcodes, but that won't work well if you pass text around as macro arguments. Also, you would not want spaces that separate macros and text to be ignored, but only those in the typeset text. For such use, LuaTEX introduces `\nospaces`.

This new primitive can be used to overrule the usual `\spaceskip`-related heuristics when a space character is seen in a text flow. The value `1` specifies no injection, a value of `2` results in injection of a zero skip, and the default `0` gets the standard behavior. Below we see the results for four characters separated by spaces. (Output has been rescaled.)



In case you wonder why setting the space related skips to zero is not enough: even when it is set to zero you will always get something. What gets inserted depends on `\spaceskip`, `\xspaceskip`, `\spacefactor` and font dimensions. I must admit that I always have to look up the details, as, normally, it's wrapped up in a spacing system that you implement once then forget about. In any case, with `\nospaces`, you can completely get rid of even an inserted zero space.

4 Token lists

The following four new primitives are provided because they are more efficient than macro-based variants: `\toksapp`, `\tokspre`, and `\e...` (expanding) versions of both. They can be used to append or prepend tokens to a token register.

However, don't overestimate the gain to be found in simple situations with not that many tokens involved (read: there is no need to instantly change all code that does it the traditional way). The new method avoids saving tokens in a temporary register. Then, when you combine registers (which is also possible), the source gets appended to the target and, afterwards, the source is emptied: we don't copy but combine!

Their use can best be demonstrated by examples. We employ a scratch register `\ToksA`. The examples here show the effects of grouping; in fact, they were written for testing this effect. Because we don't use the normal assignment code, we need to initialize a local copy in order to get the original content outside the group.

```
\ToksA{}
\bgroup \ToksA{}
  \bgroup \toksapp\ToksA{!!} [\the\ToksA=!!]
  \egroup [\the\ToksA=]
\egroup
[\the\ToksA=]
```

result: `[!!=!!] [=] [=]`

```
\ToksA{}
\bgroup \ToksA{A}
  \bgroup \toksapp\ToksA{!!} [\the\ToksA=A!!]
  \egroup [\the\ToksA=A]
\egroup
[\the\ToksA=]
```

result: `[A!!=A!!] [A=A] [=]`

```
\ToksA{}
\bgroup \ToksA{}
  \bgroup
    \ToksA{A} \toksapp\ToksA{!!}[\the\ToksA=A!!]
  \egroup [\the\ToksA=]
\egroup
[\the\ToksA=]
```

result: `[A!!=A!!] [=] [=]`

```
\ToksA{}
\bgroup \ToksA{A}
  \bgroup
    \ToksA{} \toksapp\ToksA{!!} [\the\ToksA=!!]
  \egroup [\the\ToksA=A]
\egroup
[\the\ToksA=]
```

result: `[!!=!!] [A=A] [=]`

```
\ToksA{}
\bgroup \ToksA{}
  \bgroup
    \tokspre\ToksA{!!} [\the\ToksA=!!]
  \egroup [\the\ToksA=]
\egroup
[\the\ToksA=]
```

result: `[!!=!!] [=] [=]`

```
\ToksA{}
\bgroup \ToksA{A}
  \bgroup
    \tokspre\ToksA{!!} [\the\ToksA=!!A]
  \egroup [\the\ToksA=A]
\egroup
[\the\ToksA=]
```

result: `[!!A=!!A] [A=A] [=]`

```
\ToksA{}
\bgroup \ToksA{}
  \bgroup
    \ToksA{A} \tokspre\ToksA{!!}[\the\ToksA=!!A]
  \egroup [\the\ToksA=]
\egroup
[\the\ToksA=]
```

result: `[!!A=!!A] [=] [=]`

```
\ToksA{}
\bgroup \ToksA{A}
  \bgroup
    \ToksA{} \tokspre\ToksA{!!} [\the\ToksA=!!]
  \egroup [\the\ToksA=A]
\egroup
[\the\ToksA=]
```

result: `[!!=!!] [A=A] [=]`

Here we used `\toksapp` and `\tokspre`, but there are two more primitives, `\etoksapp` and `\etokspre`; these expand the given content while it gets added.

The next example demonstrates that you can also append another token list. In this case the original content is gone after an append or prepend.

```
\ToksA{A}
\ToksB{B}
\toksapp\ToksA\ToksB
\toksapp\ToksA\ToksB
[\the\ToksA=AB]
```

result: `[AB=AB]`

This is intended behaviour! The original content of the source is not copied but really appended or prepended. Of course, grouping works well.

```
\ToksA{A}
\ToksB{B}
\bgroup
    \toksapp\ToksA\ToksB
    \toksapp\ToksA\ToksB
    [\the\ToksA=AB]
\egroup
[\the\ToksA=AB]
```

result: `[AB=AB] [AB=AB]`

5 Active characters

We now enter an area of very dirty tricks. If you have read *The TEXbook* or listened to talks by TEX experts, you will, for sure, have run into the term 'active character'. In short, it boils down to this: each character has a catcode and there are 16 possible values. For instance, backslash normally has catcode zero, braces have values one and two, and normal characters can be 11 or 12. Very special are characters with code 13 as they are 'active' and behave like macros. In Plain TEX, the tilde is one such active character, and it's defined to be a 'non-breakable space'. In ConTEXt, the vertical bar is active and used to indicate compound and fence constructs.

Below is an example of a definition:

```
\catcode`A=13
\def A{B}
```

This will make the `A` into an active character that will typeset a `B`. Of course, such an example is asking for problems since any `A` is seen that way, so a macro name that uses one will not work. Speaking of macros:

```
\def\whatever
  {\catcode`A=13
   \def A{B}}
```

This won't work out well. When the macro is read it gets tokenized and stored and at that time the catcode change is not yet done so when this macro is called the A is frozen with catcode letter (11) and the `\def` will not work as expected (it gives an error). The solution is this:

```
\bgroup
\catcode`A=13
\gdef\whatever
  {\catcode`A=13
   \def A{B}}
\egroup
```

Here we make the `A` active before the definition and we use grouping because we don't want that to be permanent. But still we have a hard-coded solution, while we might want a more general one that can be used like this:

```
\whatever{A}{B}
\whatever{=}{{\bf =}}
```

Here is the definition of `whatever`:

```
\bgroup
\catcode`~=13
\gdef\whatever#1#2%
  {\uccode`~=`#1\relax
   \catcode`#1=13
   \uppercase{\def\tempwhatever{~}}%
   \expandafter\gdef\tempwhatever{#2}}
\egroup
```

If you read backwards, you can imagine that `\tempwhatever` expands into an active `A` (the first argument). So how did it become one? The trick is in the `\uppercase` (a `\lowercase` variant will also work). When casing an active character, TeX applies the (here) uppercase and makes the result active too.

We can argue about the beauty of this trick or its weirdness, but it is a fact that for a novice user this indeed looks more than a little strange. And so, a new primitive `\letcharcode` has been introduced, not so much out of necessity but simply driven by the fact that, in my opinion, it looks more natural. Normally the meaning of the active character can be put in its own macro, say:

```
\def\MyActiveA{B}
```

We can now directly assign this meaning to the active character:

```
\letcharcode`A=\MyActiveA
```

Now, when `A` is made active this meaning kicks in:

```
\def\whatever#1#2%
  {\def\tempwhatever{#2}%
   \letcharcode`#1\tempwhatever
   \catcode`#1=13\relax}
```

We end up with less code but, more important, it is easier to explain to a user and, in my eyes, it looks less obscure, too. Of course, the educational gain here wins over any practical gain because a macro package hides such details and only implements such an active character installer once.

6 `\csname` and friends

You can check for a macro being defined as follows:

```
\ifdefined\foo
    do something
\else
    do nothing
\fi
```

which, of course, can be obscured to:

```
do \ifdefined\foo some\else no\fi thing
```

A bit more work is needed when a macro is defined using `\csname`, in which case arbitrary characters (like spaces) can be used:

```
\ifcsname something or nothing\endcsname
    do something
\else
    do nothing
\fi
```

Before ε-TeX, this was done as follows:

```
\expandafter
 \ifx\csname something or nothing\endcsname
 \relax
    do nothing
\else
    do something
\fi
```

The `\csname` primitive will do a lookup and create an entry in the hash for an undefined name that then defaults to `\relax`. This can result in many unwanted entries when checking potential macro names. Thus, ε-TeX's `\ifcsname` test primitive can be qualified as a 'necessity'.

Now take the following example:

```
\ifcsname do this\endcsname
    \csname do this\endcsname
\else\ifcsname do that\endcsname
    \csname do that\endcsname
\else
    \csname do nothing\endcsname
\fi\fi
```

If `do this` is defined, we have two lookups. If it is undefined and `do that` is defined, we have three lookups. So there is always one redundant lookup.

Also, when no match is found, TEX has to skip to the `\else` or `\fi`. One can save a bit by uglifying this to:

```
\csname do%
    \ifcsname do this\endcsname this\else
    \ifcsname do that\endcsname that\else
                            nothing\fi\fi
\endcsname
```

This, of course, assumes that there is always a final branch. So let's get back to:

```
\ifcsname do this\endcsname
    \csname do this\endcsname
\else\ifcsname do that\endcsname
    \csname do that\endcsname
\fi\fi
```

As said, when there is some match, there is always one test too many. In case you think this might be slowing down TEX, be warned: it's hard to measure. But as there can be (m)any character(s) involved, including multi-byte UTF-8 characters or embedded macros, there is a bit of penalty in terms of parsing token lists and converting to UTF-8 strings used for the lookup. And, because TEX has to give an error message in case of troubles, the already-seen tokens are stored too.

So, in order to avoid this somewhat redundant operation of parsing, memory allocation (for the lookup string) and storing tokens, the new primitive `\lastnamedcs` is now provided:

```
\ifcsname do this\endcsname
    \lastnamedcs
\else\ifcsname do that\endcsname
    \lastnamedcs
\fi\fi
```

In addition to the (in practice, often negligible) speed gain, there are other advantages: TEX has less to skip, and although skipping is fast, it still isn't a nice side effect (also useful when tracing). Another benefit is that we don't have to type the to-be-looked-up text twice. This reduces the chance of errors. In our example we also save 16 tokens (taking 64 bytes) in the format file. So, there are enough benefits to gain from this primitive, which is not a specific feature, but just an extension to an existing mechanism.

It also works in this basic case:

```
\csname do this\endcsname
\lastnamedcs
```

And even this works:

```
\csname do this\endcsname
\expandafter\let\expandafter\dothis\lastnamedcs
```

And after defining:

```
\bgroup
\expandafter
  \def\csname do this\endcsname{or that}
\global\expandafter
  \let\expandafter\dothis\lastnamedcs
\expandafter
  \def\csname do that\endcsname{or this}
\global\expandafter
  \let\expandafter\dothat\lastnamedcs
\egroup
```

We can use `\dothis` that gives `or that` and `\dothat` that gives `or this`, so we have the usual freedom to be able to use something meant to make code clean for the creation of obscure code.

A variation on this is the following:

```
\begincsname do this\endcsname
```

This call will check if `\do this` is defined, and, if so, will expand it. However, when `\do this` is not found, it does not create a hash entry. It is equivalent to:

```
\ifcsname do this\endcsname\lastnamedcs\fi
```

but it avoids the `\ifcsname`, which is sometimes handy as these tests can interfere.

I played with variations like `\ifbegincsname`, but we then quickly end up with dirty code due to the fact that we first expand something and then need to deal with the following `\else` and `\fi`. The two above-mentioned primitives are non-intrusive in the sense that they were relatively easy to add without obscuring the code base.

As a bonus, LuaTEX also provides a variant of `\string` that doesn't add the escape character: `\csstring`. There is not much to explain to this:

```
\string\whatever<>\csstring\whatever
```

This gives: `\whatever<>whatever`

The main advantage of these several new primitives is that a bit less code is needed and (at least for ConTEXt) leads to a bit less tracing output. When you enable `\tracingall` for a larger document or example, which is sometimes needed to figure out a problem, it's not much fun to work with the resulting megabyte (or sometimes even gigabyte) of output so the more we can get rid of, the better. This consequence is just an unfortunate side effect of the ConTEXt user interface with its many parameters. As said, there is no real gain in speed.

7 Packing of lists

Deep down in TEX, horizontal and vertical lists eventually get packed. Packing of an `\hbox` involves:

1. ligature building (for traditional TEX fonts),
2. kerning (for traditional TEX fonts),
3. calling out to Lua (when enabled) and

4. wrapping the list in a box and calculating the width.

When a Lua function is called, in most cases, the location where it happens (group code) is also passed. But say that you try the following:

```
\hbox{\hbox{\hbox{\hbox foo}}}
```

Here we do all four steps, while for the three outer boxes, only the last step makes any sense. And it's not trivial to avoid the application of the Lua function here. Of course, one can assign an attribute to the boxes and use that to intercept, but it's kind of clumsy. This is why we now can say:

```
\hpack{\hpack{\hpack{\hbox foo}}}
```

There are also `\vpack` for a `\vbox` and `\tpack` for a `\vtop`. There can be a small gain in speed when many complex manipulations are done, although in, for instance, ConTEXt, we already have provisions for that. It's just that the new primitives are a cleaner way out of a conceptually nasty problem. Similar functions are available on the Lua side.

8 Errors

We end with a few options that can be convenient to use if you don't care about exact compatibility.

```
\suppresslongerror
\suppressmathparerror
\suppressoutererror
\suppressifcsnameerror
```

When entering your document on a paper teletype terminal, starting TEX, and then going home in order to have a look at the result the next day, it does make sense to catch runaway cases, like premature ending of a paragraph (using `\par` or equivalent empty lines), or potentially missing `$$`s. Nowadays, it's less important to catch such coding issues (and be more tolerant) because editing takes place on screen and running (and restarting) TEX is very fast.

The first two flags given above deal with this. If you set the first to any value greater than zero, macros not defined as `\long` (not accepting paragraph endings) will not complain about `\par` tokens in arguments. The second setting permits and ignores empty lines (also pars) in math without reverting to dirty tricks. Both are handy when your content comes from places that are outside of your control. The job will not be aborted (or hang) because of an empty line.

The third setting suppresses the `\outer` directive so that macros that originally can only be used at the outer level can now be used anywhere. It's hard to explain the concept of outer (and the related error message) to a user anyway.

The last one is a bit special. Normally, when you use `\ifcsname` you will get an error when TEX sees something unexpandable or that can't be part of a name. But sometimes you might find it to be quite acceptable and can just consider the condition as false. When the fourth variable is set to non-zero, TEX will ignore this issue and try to finish the check properly, so basically you then have an `\iffalse`.

9 Final remarks

I mentioned performance a number of times, and it's good to notice that most changes discussed here will potentially be faster than the alternatives, but this is not always noticeable, in practice. There are several reasons.

For one thing, TEX is already highly optimized. It has speedy memory management of tokens and nodes and unnecessary code paths are avoided. However, due to extensions to the original code, a bit more happens in the engine than in decades past. For instance, Unicode fonts demand sparse arrays instead of fixed-size, 256-slot data structures. Handling UTF involves more testing and construction of more complex strings. Directional typesetting leads to more testing and housekeeping in the frontend as well as the backend. More keywords to handle, for instance `\hbox`, result in more parsing and pushing back unmatched tokens. Some of the penalty has been compensated for through the changing of whatsits into regular nodes. In recent versions of LuaTEX, scanning of `\hbox` arguments is somewhat more efficient, too.

In any case, any speedup we manage to achieve, as said before, can easily become noise through inefficient macro coding or user's writing bad styles. And we're pretty sure that not much more speed can be squeezed out. To achieve higher performance, it's time to buy a machine with a faster CPU (and a huge cache), faster memory (lanes), an SSD, and regularly check your coding.

◇ Hans Hagen
Pragma ADE
http://pragma-ade.com

Hyphenation languages in LuaTEX 0.90

Hans Hagen

In LuaTEX you can define up to 16,383 separate languages, and words can be up to 256 characters long. The language is stored with each character. You can set `\firstvalidlanguage` (a new variable) to, for instance, 1 and thereby make language 0 an ignored hyphenation language. Because the language is stored in the glyph nodes this is an efficient way to disable hyphenation locally.

The new primitive `\hyphenationmin` can be set to specify the minimum length of a word considered for hyphenation. This value is stored with the (current) language and applies to the whole paragraph. Because `\lefthyphenmin` and `\righthyphenmin` are stored with the glyphs you can temporarily change them. The `\uchyph` value is also saved in the actual nodes, therefore its handling is different from TEX82: changes to `\uchyph` become effective immediately, not at the end of the current partial paragraph.

LuaTEX now uses the new language-specific variables `\prehyphenchar` and `\posthyphenchar` when creating implicit discretionaries, instead of TEX82's `\hyphenchar`, and new variables `\preexhyphenchar` and `\postexhyphenchar` (also language-specific) for explicit discretionaries, instead of TEX82's empty discretionary.

Typeset boxes now always have their language information embedded in the nodes themselves, so there is no longer a dependency on the surrounding language settings. In TEX82, a mid-paragraph statement like `\unhbox0` would process the box using the current paragraph language unless there was a `\setlanguage` issued inside the box. In LuaTEX all language variables are already frozen.

In traditional TEX, hyphenation is driven by the so-called `\lccode` table. In LuaTEX we made this dependency less strong. Several strategies are possible. When you do nothing, the currently-used `\lccode`'s are still the default when loading patterns, setting exceptions or hyphenating a list.

When you set `\savinghyphcodes` to a value larger than zero the current set of `\lccode`s will be saved with the language but in a dedicated namespace reflecting hyphenation-justification codes. In this case changing a `\lccode` afterwards has no effect. Instead of `\lccode` you can use `\hjcode`. These are per-language and when set take precedence over the shared `\lccode`s. So, LuaTEX doesn't store `\lccode`s per language but has a dedicated hyphenation-justification code instead. You can change these values at any time with `\hjcode`a=`a`.

This change is global which makes sense if you keep in mind that the moment when hyphenation happens is (normally) when the paragraph or a horizontal box is constructed. If `\savinghyphcodes` was zero when the language was initialized you start out with nothing, otherwise you already have a set. Beware: the `\hjcode` values are always saved in the format, independent of the value of `\savinghyphcodes` when the format is dumped.

The value of the two counters related to hyphenation, `\hyphenpenalty` and `\exhyphenpenalty`, are now stored in the discretionary nodes. This permits a local overload when explicit `\discretionary` commands are used. The implementation is downward compatible but permits control for special situations.

MetaPost arrowhead variants

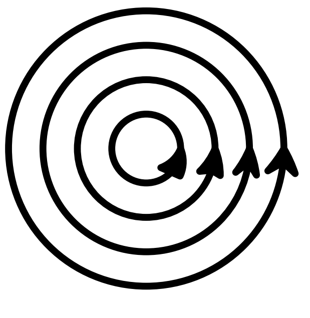
Alan Braslau, Hans Hagen

Some colleagues have complained that the arrowheads in MetaPost graphics are too blunt, and that they would like to see a more stylish arrowhead. Perhaps they do not appreciate how MetaPost produces arrowheads that actually follow a curved path? Still, we realized that one can easily modify the arrowhead macro to produce variants to satisfy everyone while remaining simple and elegant.

We settled on a backwards-compatible solution of adding two new global, internal variables to the existing `ahlength` and `ahangle`: `ahvariant` and `ahdimple`. With the default `ahvariant:=0`, one gets the traditional MetaPost arrowhead. A nonzero value will give a more stylized arrowhead that uses the value of `ahdimple`, a unitless fraction of `ahlength`, by default 0.2, to create a dimple at the base. The `variant:=1` uses `...` to give a rounded dimple or "ear" and `variant:=2` uses `--` to create a barb. Finally, a sort of "broadhead" can be produced by making `ahdimple` negative. (We also made an efficiency change in PDF drawing that leads to an improvement when drawing arrowheads.)

This change is now part of MetaFun and ConTEXt and can be easily included as MetaPost macros. Examples follow (scaled for *TUGboat*).

```
pickup pencircle scaled 2mm;
ahlength := 8mm;drawarrow fullcircle scaled 2cm;
ahvariant := 1; drawarrow fullcircle scaled 4cm;
ahdimple := .5; drawarrow fullcircle scaled 6cm;
ahangle := 60;  drawarrow fullcircle scaled 8cm;
```



A personal book catalogue: `bookdb`

Peter Wilson

1 Introduction

For many years I have been collecting books. They are located in five rooms, as well as two bookcases in my printing and binding workshop. But recently I found that I was buying books that I already had. I decided that the best way to save money was to catalogue all the books that I owned and where they were kept.

I searched on the web for free database programs that would be appropriate. There were only a few that would run under Linux, which is the operating system I am most comfortable with. Of those, some I could not install, and the one that I could I couldn't get to work for me.

I contacted several booksellers that I dealt with and asked them what they used for cataloguing their stock. They all replied, but most used proprietary and expensive software that included things like preparing invoices that were irrelevant as far as I was concerned. The two responses that grabbed my attention were 'use a card index' (but I needed space for books, not card indexes) and 'try bibtex', which immediately appealed as I had used it for many years; why I hadn't thought of it myself I'll never know.

2 Usage

The standard BibTeX entries did not meet my needs. I looked at `biblatex` [3] but its entries also didn't match my requirements so I decided to tweak BibTeX. To that end I used Patrick Daly's `makebst` program [2] for generating a `*.bst` file that went some way towards meeting my needs. This required some hand-coded additions later; I read the articles by Oren Patashnik [5], the creator of BibTeX, and Nicolas Markey [4], which helped me on my way. In the end I had a BibTeX file called `bookdb.bst` [6] that included all the regular entries but a greatly expanded `book` entry, as follows:

`book`	A book with a publisher *Required:* `author` or editor, `title`, `publisher`, `year` *Optional:* `volume` or `number`, `series`, `address`, `edition`, `month`, `note`, `collator`, `foreword`, `preface`, `introduction`, `volumes`, `pages`, `illustrations`, `binding`, `size`, `condition`, `copy`, `location`, `category`, `value`

Also, I created a new entry called `heading`, as follows:

`heading`	A heading in the bibliography *Required:* `key` *Optional:* `note`

The additional fields in the `book` entry are:

`binding:`	Information about the book's binding. Output as: Binding: 'binding.'
`category:`	The general theme of the book. Output as: Category: '**category.**'
`collator:`	The name(s) of those who collated the book contents. Output as: 'collator' (collator(s))
`condition:`	The book's condition. Output as: Condition: 'condition.'
`copy:`	For a limited edition, the particular copy. Output as: Copy: 'copy.'
`foreword:`	The names(s) of the author(s) of the book's Foreword, if not written by the author(s) of the main text. Output as: foreword by 'foreword'
`illustrations:`	Information about the number and kind of any illustrations and possibly who created them. Output as: Illustrations: 'illustrations.'
`introduction:`	The name(s) of the author(s) of the book's Introduction, if not written by the author(s) of the main text. Output as: introduction by 'introduction'
`location:`	Where the book is located. Output as: Location: 'location.'
`pages:`	The total number of pages. Output as: 'pages' pp.,
`preface:`	The names(s) of the author(s) of the book's Preface, if not written by the author(s) of the main text. Output as: preface by 'preface'
`size:`	The book's physical dimensions. Output as: Size: 'size.'

`value:`	The book's value. Output as: Value: '**value**.'
`volumes:`	The number of volumes. Output as: Volumes: 'volumes.'

I use the `heading` entry for putting a heading or division marker into a bibliography. The `key` is required so that the `heading` is sorted into the correct position in the bibliography (normally sorting is based on the author or editor). The contents of the `note` field form the printed heading. For instance if you wanted a heading before each alphabetical group of authors you could do something like:

```
@heading{A1,
  key = {A1},
  note = {\ahead{AAAAAAAA...}}
}
@heading{B1,
  key ={B1},
  note = {\ahead{BBBBBBBB...}}
}
etc
```

where `\ahead` might be defined as:

```
\providecommand{\ahead}[1]{%
  \textbf{\large #1}}
```

To help clarify matters, Figure 1 shows a possible entry in a `*.bib` file. The output after processing in a document using `bookdb.bst` is illustrated in Figure 2.

As an example this is a file that I use for printing a catalogue of my books, where the book details are in file `mybooks.bib`. Note that using `bookdb` requires the use of the `natbib` package [1].

```
% books.tex  a catalogue of my books
\documentclass[11pt,a4paper]{memoir}
\usepackage[T1]{fontenc}
\usepackage{natbib}
\pagestyle{empty}
\begin{document}
\nocite{*}
\bibliographystyle{bookdb}
\bibliography{mybooks}
\end{document}
```

3 Implementation

As I said earlier I had to extend the `bookdb.bst` file produced by the `makebst` program. I didn't really know how it all worked but after much trial and many errors I got something that on the whole met my needs. My basic process was to copy elements of the original `bst`, change some names, and see what was produced.

First of all I added the new `book` fields to the `bookdb.bst` `ENTRY` command as:

```
@book{ABOOK,
  author = {A. N. Author and A. Nother},
  title = {A Book Entry},
  editor = {Smith and Jones},
  collator = {Jane and Tim},
  translator = {Jo and Mary},
  foreword = {Alpha},
  preface = {Zoe},
  introduction = {Bloggs and Friend},
  volume = 7,
  publisher = {Herries Press},
  year = 2020,
  pages = {xii + 278 + vi},
  edition = {Third},
  isbn = {0-201-36299-8},
  volumes = 9,
  illustrations = {11 wood engravings},
  binding = {full red leather},
  size = {11 by 17 inches},
  note = {This is a note},
  condition = {Hot off the press},
  copy = {23 of 125},
  location = {my study},
  category = {private press},
  value = {\$270}
}
```

Figure 1: An example entry for `bookdb` processing

A. N. Author and A. Nother. *A Book Entry*, Smith and Jones (eds.), Jane and Tim (collators), Jo and Mary (translators), foreword by Alpha, preface by Zoe, introduction by Bloggs and Friend, volume 7 (Herries Press, 2020), xii + 278 + vi pp., third edition. ISBN 0-201-36299-8. Volumes: 9. Illustrations: 11 wood engravings. Binding: full red leather. Size: 11 by 17 inches. This is a note. Condition: Hot off the press. Copy: 23 of 125. Location: my study. Category: **private press**. Value: **$270**.

Figure 2: The example's output

```
ENTRY
  { ...
    binding
    category
    collator
    condition
    copy
    foreword
    illustrations
    introduction
    location
    pages
    preface
    size
    translator
    value
    volumes
  }
```

The next thing was to add the new entries in the correct order to the function that output the `book` bibliography entries, together with how they should be formatted. This was the final result after many repetitions of try it, BibTeX it, change it. The original code is in a typewriter font and my additions are in slanted typewriter.

```
FUNCTION {book}
{ output.bibitem
  author empty$
    { format.editors "author and editor"
                       output.check
      editor format.key output
      add.blank
    }
    { format.authors output.nonnull
       crossref missing$
        { "author and editor" editor
                       either.or.check }
        'skip$
       if$
    }
  if$
  new.block
  format.btitle "title" output.check
  format.editors output
  format.collator output
  format.translator output
  format.foreword output
  format.preface output
  format.introduction output
  crossref missing$
   { format.bvolume output
     new.block
     format.number.series output
     new.sentence
     format.publisher.address output
   }
   {
    new.block
    format.book.crossref output.nonnull
   }
  if$
  format.book.pages output
  format.edition output
  format.isbn output
  format.volumes output
  format.illustrations output
  format.binding output
  format.size output
  new.block
  format.note output
  format.condition output
  format.copy output
  format.location output
  format.category output
  format.value output
  fin.entry
}
```

Effectively 'all' I had left to do was specify the formatting of my new fields. I used three basic forms:

1. Some introductory text, like 'introduction by' or 'Illustrations:', followed by the field data.
2. Like the first form but with the field data in a bold font.
3. Name(s) forming the field data followed by what their contribution was in parentheses.

As an example of the first form here is the code for `binding`:

```
FUNCTION {format.binding}
{ binding "binding" bibinfo.check
  duplicate$ empty$ 'skip$
    {
      new.block
      "Binding: " swap$ *
    }
  if$
}
```

Life was a little more complicated for the second form. This is the code for the `value` field which requires two functions, the first dealing with the bolding and the second with the output.

```
FUNCTION {boldval}
{ duplicate$ empty$
  { pop$ "" }
  { "Value: \textbf{" swap$ * "}" * }
 if$
}
FUNCTION {format.value}
{ value "value" bibinfo.check
  duplicate$ empty$ 'skip$
    {
      new.block
      boldval
    }
  if$
}
```

The third form required several functions, as in the code for `collator`, where if there is a single collator this is output as ‘Name (collator)’ but if the are multiple collators the output is ‘Name1 and Name2 … (collators)’.

```
FUNCTION {bbl.collator}
{ "collator" }
FUNCTION {bbl.collators}
{ "collators" }
FUNCTION {get.bbl.collator}
{ collator num.names$ #1 >
    'bbl.collators 'bbl.collator
  if$
}
FUNCTION {format.collator}
{ collator "collator" format.names
    duplicate$ empty$ 'skip$
  { " " *
    get.bbl.collator
  "(" swap$ * ")" *
  }
  if$
}
```

4 My book database

The `*.bib` for my book catalogue looks somewhat like this:

```
%%% mybooks.bib 2015/04/22
%%% for formatting headings
@preamble{ "\providecommand{\ahead}[1]{%
 \textbf{\large #1}}" }

%%% publishers
@string{CUP =
  "Cambridge University Press"}
% etc
%%% categories
@string{science =
  "science, mathematics, computers"}
% etc

@heading{A1,
  key = {A},
  note = {\ahead{AAAAAAAAAAA....}
}

@book{A1KEY,
author = {First A. Author},
% etc
}

@book{A2KEY,
author = {Second A. Author},
% etc
}
% etc
```

I used the BibTeX `@preamble` command to provide a definition of the `\ahead` macro. This, if required, can be overridden by an existing definition in the document used to print the bibliography.

I added various `@string` commands to provide shorthands for many of the fields in the `.bib` file, such as publisher, location, category, that would have the same value. This meant that I could have a shortened field entry that looked like:

```
publisher = CUP,
```

instead of:

```
publisher = {Cambridge University Press},
```

5 Afterthoughts and unresolved problems

BibTeX uses a stack-based language which I find hard to understand. Many years ago I wrote an interpreter for a stack-based language whose name I have forgotten but even so I was unable to use the language itself. I think that it is a little like crosswords. I like doing ‘cryptic’ crosswords but I find that with some setters I can follow their clues easily but with others I haven’t a clue.

My basic approach was to take an existing `*.bst` file, try and see what it did, then copy and modify what seemed relevant to my needs.

I did have a couple of infelicities that I did not manage to resolve.

The first was that no matter what I tried I could not stop the `heading` from outputting its `key`, so it should be made as short and unobtrusive as possible.

The second was that I couldn’t stop the warning message issued by BibTeX if both an `author` and `editor` were supplied although the output was printed including both.

In spite of these, if you are a collector then you may want to consider tweaking a `*.bst` file to meet your particular needs.

6 Other collections

Perhaps, like me, you have collections other than books. These can also be catalogued via BibTeX. For instance I collect Japanese woodblock prints and Western engravings, while you might collect pictures in general. I have no need to create a catalogue of my prints as I keep them in folders in one place, together with information about each one. If I did want to create such a catalogue I would start with `bookdb.bst`, renaming it to perhaps `pictures.bst`. Then add in the new fields for the prints and engravings, such as:

`artist:`	The name (and perhaps the date) of the artist. Output like author.
`engraver:`	The name of the engraver.

`censor:`	The name(s) of the censors. Output like collators.

Then define a new `japanese` entry, based on the `book`, with maybe the fields:

`japanese`	A Japanese woodblock print with an artist *Required:* `artist, title` *Optional:* `engraver, censor, publisher, date, series, note, size, value, category`

And a new `engraving` entry, based on the `book`, with possibly the fields:

`engraving`	An engraving with an engraver *Required:* `engraver, title` *Optional:* `date, series, note, size, value, category`

Then delete everything that might be irrelevant, such as `inproceedings`, etc.

You could use similar enhancements to catalogue, say, a collection of watercolours or Dinky toys or model trains.

It's up to you!

References

[1] Patrick W. Daly. `natbib` — natural sciences citations and references, September 2010. `http://ctan.org/pkg/natbib`.

[2] Patrick W. Daly. `custom-bib` — customizing bibliographic style files, November 2011. `http://ctan.org/pkg/custom-bib`.

[3] Philipp Lehman and Philip Kime. `biblatex` — bibliographies in LaTeX using BibTeX for sorting only, March 2016. `http://ctan.org/pkg/biblatex`.

[4] Nicolas Markey. Tame the BeaST, October 2009. `http://ctan.org/pkg/tamethebeast`.

[5] Oren Patashnik. Designing BibTeX styles, February 1988. `http://mirror.ctan.org/biblio/bibtex/base/btxhak.pdf`.

[6] Peter Wilson. `bookdb` — a personal book catalogue, June 2015. `http://ctan.org/pkg/bookdb`.

◇ Peter Wilson
12 Sovereign Close
Kenilworth, CV8 1SQ
UK
`herries dot press (at) earthlink dot net`

OPmac-bib: Citations using `*.bib` files with no external program

Petr Olšák

Introduction

The OPmac package [1] is a set of plain TeX macros which implements the basic LaTeX functionality in a simple way. The OPmac-bib module is part of OPmac. It provides the `*.bib` manipulation without any external program (such as BibTeX [2] or `biber` [3]). This allows you to forget about encoding problems when using the ancient BibTeX, forget about calling an external program and forget about working with the highly unconventional `*.bst` language. We don't need such complications in plain TeX. We are able to generate bibliography references directly from `*.bib` files only by simple TeX macros. This article introduces the main principles of OPmac-bib.

The `librarian.tex` [4] package is used by OPmac-bib for scanning the `*.bib` databases.

The OPmac and OPmac-bib packages are successfully used by students at CTU in Prague for creating their theses using the CTUstyle template [5].

Common principles

If we need to refer to the existence of another document in our document then such a reference is the subject of formal rules. The rules depend on the scientific discipline, journal requirement, normal conventions, or a mix of these aspects. We first need to declare the terminology about this.

We use *actual document* for the document where the citation is used (linked from) and *cited document* for the mentioned source (linked to). The place in the text of actual document where we need to refer to the cited document is the *citation point.* Common rules say that authors of an actual document only have to have a *citation mark* at the citation point, and they must put the *bibliographic entry* at the end of the actual document in a special section usually called "References" or "Bibliography". The citation mark can be repeated next to the bibliographic entry. The list of all bibliographic entries cited in the actual document is the *references list.* Each bibliographic entry must include *fields* (author, title, year of edition etc.). They must unambiguously specify the cited document. There are *mandatory fields* and *optional fields*, they depend on the *type* (book, article etc.) of the cited document and, of course, on the formal rules. The citation marks may be *numbered* or *non-numbered.*

The non-numbered marks may be *long* (like Knuth 1984) or *short* (like Kn84). Only one type of the citation marks must be used in the whole document.

The TEX user typically doesn't care about citation marks and bibliographic entries. These are generated automatically. The user puts only a *label* at the citation point into the source text of the actual document. The label is invisible in the printed version of the document but it must be the same as the label which determines the bibliographic entry in the `*.bib` database(s) used.

Various rules, conventions and standards govern the following:

- The format of citation marks.
- The rules for grouping citation marks if multiple documents are cited at the same citation point.
- The type of brackets used around citation marks, or around groups of them, possibly with other rules of printing the citation marks.
- Which fields are mandatory and which are optional (dependent on the type of the cited document) when printing the bibliographic entry.
- How to format the bibliographic entry: what separators (punctuation, reserved words or abbreviations) must be used between fields, what font to use for what types of fields, what ordering of fields to use in a given entry.
- The reserved words or abbreviations mentioned above (like "edition", "et al.", "pp.", "available from" etc., can be printed in the language of the actual document or in the language of the cited document. It depends on the kind of reserved word.
- The list of authors is a special field in the entry. There are rules about how to print the names of each individual author (the ordering of first name, last name and other names, the abbreviations of these names etc.), what separators are used between authors, what ordering of authors may be used (alphabetical or by credit).
- There are conventions about ordering the bibliographic entries in the reference list: alphabetically by the first author, by the year of printing, a mix of these, by the occurrence of citation points in the actual document, etc.

It is clear that implementation of these various rules for automatic generation of bibliographic references is a very complicated task. But OPmac-bib solves it and uses only clear and simple TEX macros. No more special external format, no external program is used. Anyone can simply change the predefined TEX macros in order to follow different conventions.

OPmac without bib

The OPmac package without the OPmac-bib module provides the basic manipulation with citations. We give a short summary in this section.

A user writes `\cite[`⟨*label*⟩`]` or `\cite[`⟨*more comma separated labels*⟩`]` at the citation point. The bibliographic entry (at the end of the document) can be created manually with:

`\bib [`⟨*label*⟩`]` ⟨*text of the entry*⟩

The citation marks are auto-generated, numbered by default. When the `\sortcitations` declaration is used then multiple citation marks at a shared citation point are sorted sequentially; and when `\shortcitations` is used then a continuous sequence of the marks is converted to ⟨*from*⟩–⟨*to*⟩ form, for example [3, 4, 5, 11, 12] is converted to [3–5, 11–12].

The OPmac documentation describes how to print different brackets around citation marks (square brackets are the default) or how to use a different format for printing marks.

When `\nonumcitations` is declared then non-numbered citation marks are used. The format of these marks can be declared as a `\bib` parameter when the entries are set manually:

`\bib [`⟨*label*⟩`] = {`⟨*mark*⟩`}` ⟨*text of the entry*⟩

OPmac without OPmac-bib allows the use of BIBTEX as an external program. You can write the following at the place of the reference list:

`\usebibtex{`⟨*bib-base*⟩`}{`⟨*bst-style*⟩`}`

and the reference list is generated automatically. The ⟨*bib-base*⟩ is the name (without extension) of the `*.bib` file used and ⟨*bst-style*⟩ is the name the `*.bst` style file used. The ⟨*labels*⟩ used in the actual document must, of course, correspond with the labels in the `*.bib` file. Four steps must be processed to create the document in such case: TEX, BIBTEX, TEX, TEX.

The format of the generated reference list is given by the `*.bst` style but the obscure language used in these files makes it difficult to modify the formatting. So, OPmac-bib (described in next sections) gives better flexibility for setting the format of reference list.

Bibliographic databases

The common format used for bibliographic entries in the TEX world is derived from BIBTEX input and the files have extension `.bib`. The main advantage of this format is that it is a text file, and humans can read it and modify it with an ordinary text editor. It is a well-arranged text file from a human point of view (not the typically obfuscated XML, for example). Although it is a very old format, it can be exported from almost all current bibliographic software. And there are large amounts of data prepared in this format. GUI-oriented programs for manipulating and managing bibliographic databases in this format are available as well.

An entry in the `*.bib` database looks like this:

```
@Book{Knuth:1984:TB,
  author =    "Donald E. Knuth",
  title =     "The {\TeX}book",
  publisher = pub-AW,
  address =   pub-AW:adr,
  pages =     "ix + 483",
  year =      "1984",
  ISBN =      "0-201-13448-9 (paperback),
               0-201-13447-0 (hardcover)",
  ISBN-13 =   "978-0-201-13448-3 (paperback),
               978-0-201-13447-6 (hardcover)",
  LCCN =      "Z253.4.T47 K58 1984",
  bibdate =   "Fri Jul 22 09:08:51 1994",
  bibsource = "http://www.math.../texbook3.bib",
  price =     "US\$15.95 (paperback),
               US\$32.95 (hardcover)",
}
```

The main syntax of this format for one bibliographic entry can be expressed by:

```
@⟨type⟩ { ⟨label⟩,
   ⟨key⟩ = "⟨value⟩",
   ⟨key⟩ = "⟨value⟩",
   ...
}
```

where ⟨*type*⟩ is a type of the cited document (book, article etc.) and ⟨*label*⟩ is the label used in `\cite[⟨label⟩]`.

The ⟨*key*⟩ gives the type of the field and ⟨*value*⟩ of the field can be enclosed in quotes or braces. If no such delimiter is used, the value is purely numeric, or it is a string identifier or string operations. For example, `pub-AW` and `pub-AW:adr` are string identifiers in the example above. They are declared in the same `*.bib` file as:

```
@String{pub-AW  = "Ad{\-d}i{\-s}on-Wes{\-l}ey"}
@String{pub-AW:adr = "Reading, MA, USA"}
```

Unfortunately for our purposes, the `*.bib` format was designed to be read by BIBTEX, and was never intended to be read directly by TEX. So, there are many TEX-unfriendly rules: the two types of delimiters for ⟨*value*⟩s, the case-insensitive identifiers for ⟨*type*⟩s and ⟨*key*⟩s, the very loose rule for setting names of given authors (described in the next section), the special `@string` manipulation (we don't need it because we now will have much more powerful TEX macro language for this), etc.

The OPmac-bib module uses the macro file `librarian.tex` by Paul Isambert for scanning the `*.bib` databases. This macro solves all the TEX-unfriendly aspects mentioned above except for one: the `@string` manipulation. I hope that this does not matter because nowadays many `*.bib` files (generated by bibliographic software or managed by GUI-oriented software) don't use the `@string` feature. And if somebody uses a `*.bib` file where `@string` is present then he/she can apply the `@string` operations manually or with a conversion script.

The Wikipedia page for BIBTEX [6] mentions the common ⟨*type*⟩s and ⟨*key*⟩s used in original BIBTEX and especially in the original `*.bst` styles interpreted by BIBTEX (which are still widely used today). If your citations go beyond this scope, then you have a problem. For example, fields such as `isbn`, `url`, and `doi` are not interpreted by the original `*.bst` styles. You must modify the `*.bst` file — typically a difficult task because of the obscure postfix-based language. Or you can use OPmac-bib to be able to modify the "bib-style" files implemented by straightforward TEX macros.

OPmac-bib from a user's point of view

You write `\input opmac-bib` at the beginning of your document. You don't need `\input opmac` itself because `opmac-bib` loads `opmac.tex` if this isn't done already. The `librarian.tex` macro file is loaded too, so the Librarian package must be installed. It requires ε-TEX activated in the format.

You can use `\cite[⟨label⟩]` or `\cite[⟨labels⟩]` as usual. All ⟨*label*⟩s must correspond with ⟨*label*⟩s in the used `*.bib` database. Similarly, you can use `\nocite[⟨labels⟩]` to insert the corresponding bibliographic entry in the reference list without printing a citation mark at the citation point. If you want to print the whole `*.bib` database then you can write `\nocite[*]`.

The reference list is generated by

```
\usebib/s (⟨style⟩) ⟨bib-base⟩
```

where ⟨*bib-base*⟩ is the name of a `*.bib` file, without extension. You can read multiple `*.bib` files: use a comma-separated list of such file names (without spaces). The ⟨*style*⟩ parameter specifies the used bib-style which determines the printed format of the reference list. Namely, the ⟨*style*⟩ must be a part of a file named `opmac-bib-`⟨*style*⟩`.tex` which will be used for the reference list format. The option `/s` says that the ordering of bibliographic entries is determined by the ⟨*style*⟩. Instead of `/s`, you can use `/c`, which says that the ordering of entries is given by the order of `\cite` or `\nocite` commands in the actual document.

Two processing steps are usually required to create the document: TEX, TEX. In the first, the connection between citation marks and labels is established and in the second, the citation marks are printed correctly.

Two "bib-style" files are provided in the OPmac-bib package: `opmac-bib-simple.tex` and `opmac-bib-iso690.tex`. The second of these prints the reference list in accordance with the ISO 690 standard; the first one is a simple implementation of the style and you can use it as a starting point for your own projects. Moreover, a file `op-example.bib` is included with OPmac-bib, as an example of a `*.bib` file.

You can start experimenting with the following code:

```
\input opmac-bib

Here is \cite[tbn,texbook] and also \cite[lech].
\bigskip
\usebib/s (simple) op-example
\bye
```

For instance, you can try changing `simple` to `iso690` for the style. If you are not using `csplain` but normal `pdftex`, then you will see that the accented letters in the name Olšák are lost. This is due to the fact that `op-example.bib` uses accented letters in the UTF-8 encoding and classic `pdftex` is unable to easily interpret this. You can use `pdfcsplain` instead of `pdftex`. Or you can use `xetex`, but then appropriate Unicode-ready fonts must be loaded, for example with

```
\input ucode
\input lmfonts
```

If multiple entries in the database have the same ⟨*label*⟩ then the rule "first entry wins" is applied (as of the Jan. 2016 version of OPmac-bib). This makes it possible to store the exceptions for a particular document in a (for example) `local.bib` file saved in the same directory as the document and then use a list of database files like this:

```
\usebib/s (iso690) local,global,op-example
```

The list of database files can be arbitrarily long. Once all desired entries (declared by `\cite` and `\nocite`) have been satisfied, then any remaining files in the list are simply skipped.

Features of the `iso690` bib-style

The detailed documentation of the `iso690` style is placed in the file `opmac-bib-iso690.tex` after `\endpinput`. We mention only the basic features in this article. On the other hand, we write here in more detail of the `author` field in order to show the complexity of the problem.

The `iso690` style ultimately accepts the same ⟨*type*⟩s and ⟨*key*⟩s in a `*.bib` file as the standard `*.bst` styles used by BIBTEX. So, you can use existing `*.bib` files and the result is essentially the same. Moreover, you can use the following fields for each entry in a `*.bib` file:

```
option    ... space separated parameters, they
              specify more rules for formatting
lang      ... two-letters specification of the
              language of cited document
              (en, cs, sk, de, etc.)
bibmark   ... the non-numbered citation mark
ednote    ... the editorial info
              (illustrations etc.)
citedate  ... the date of citation in the
              YYYY/MM/DD format.
numbering ... alternative format for
              numbering of Journal volumes.
isbn      ... ISBN
issn      ... ISSN
doi       ... DOI
url       ... URL
```

The `author` field. This field type (i.e. a ⟨*key*⟩ used in `*.bib` files) is well-known from BIBTEX. But I'll include a short review here and describe new features of `author` field using the `iso690` style.

The `author` field includes one or more authors of the cited document. All names in the author field must separated by `␣and␣` separator. Each author can be written in various formats (the "von" part is typically missing):

```
Firstname(s) von Lastname
or
von Lastname, Firstname(s)
or
von Lastname, After, Firstname(s)
```

Only the Lastname part is mandatory. Example:

```
Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
or
da Vinci, painter, Leonardo Piero
```

The separator `␣and␣` between authors is usually changed to a comma for printing, but between the second-to-last and final author the word "and" (or something equivalent, depending on the language of the cited document) is printed.

The name of the first author is printed in reverse order: "LASTNAME, Firstname(s) von, After", while all following authors are printed in normal order: "Firstname(s) von LASTNAME, After". This follows the ISO 690 standard. The Lastname is capitalized using uppercase letters, but if the `\sc` command is defined, then it is used as a font switcher in the form `{\sc Lastname}`. You can declare the "Caps and small caps" font here.

You can specify an option `aumax:`⟨*number*⟩. Here, the ⟨*number*⟩ denotes the maximum number of authors to be printed. Any additional authors are ignored and the "et al." is appended to the list of printed authors. This text is printed only if the `aumax` value is less than the real number of authors. If you have the same number of authors in the `*.bib` file as you need to print but you want to append "et al." then you can use the `auetal` option.

There is also an option `aumin:`⟨*number*⟩ to specify the definitive number of printed authors if the author list cannot be fully printed due to `aumax`. If `aumin` is unused then `aumax` authors are printed in such case.

All authors are printed if the `aumax:`⟨*number*⟩ option isn't given. There is no internal limit. But you can set the global options in your document by setting the `\biboptions` macro. For example:

```
\def\biboptions {aumax:7 aumin:1}
```

means that if there are 8 or more authors, only the first author name is printed.

Some general examples:

```
author = "John Red and Bob Brown and Tom Black",
```

output: RED, John, Bob BROWN, and Tom BLACK.

```
author = "John Red and Bob Brown and Tom Black",
option = "aumax:1",
```

output: RED, John, et al.

```
author = "John Red and Bob Brown and Tom Black",
option = "aumax:2",
```

output: RED, John, Bob BROWN, et al.

```
author = "John Red and Bob Brown and Tom Black",
option = "aumax:3",
```

output: RED, John, Bob BROWN, and Tom BLACK.

```
author = "John Red and Bob Brown and Tom Black",
option = "auetal",
```

output: RED, John, Bob BROWN, and Tom BLACK, et al.

If you need to add text before or after the authors list, you can use the `auprint:{`⟨*value*⟩`}` option. The ⟨*value*⟩ is printed instead of the authors list. The ⟨*value*⟩ can include the `\AU` macro which expands to the authors list. Example:

```
author = "Robert Galbraith",
option = "auprint:{\AU\space [pseudonym
                     of J. K. Rowling]}",
```

output: GALBRAITH, Robert [pseudonym of J. K. Rowling].

Another option is `autrim:`⟨*number*⟩. All Firstnames of all authors are trimmed (i.e. reduced to initials) if the number of authors in the author field is greater than or equal to ⟨*number*⟩. There is an exception: `autrim:0` means that no Firstnames are trimmed. This is the default behavior. Another example: `autrim:1` means that all Firstnames are trimmed.

```
author = "John Red and Bob Brown and Tom Black",
option = "auetal autrim:1",
```

output: RED, J., B. BROWN, T. BLACK, et al.

Language of reserved words. There are two kinds of reserved words and abbreviations which are automatically inserted in the bibliographic entries:

- The word is a part of a field. Two examples are the conjunction "and" between the second-to-last and final author and the "et al." phrase. Such words have to be printed in the language of the cited document.
- The word is prepended to a field value, and it is desired to use the same language for it throughout the entire reference list. For example, the prefix phrase "available from" is used before the `url` field, required by the ISO 690 norm. Such words have to be printed in the language of the actual document.

The language of the actual document is declared by the selection of hyphenation patterns. For example, when we are using `csplain` the default hyphenation is for English, but this is changed to Czech when `\chyph` or `\cslang` is selected at the beginning of the document. You can experiment with our example of `\cite[tbn,texbook]` mentioned in the previous section. Replace `simple` by `iso690`, and see the result: "Available from" is prefixed before the url. But if you use `csplain` and declare `\chyph`, then "Available from" is changed to "Dostupné na" (the Czech equivalent).

The language of the cited document is supposed to be the same as the language of the actual document unless the `lang` field is specified. If the `lang` field is given then it declares the language of the cited document. For example:

```
author = "John Red and Bob Brown and Tom Black",
option = "auetal autrim:1",
lang   = "cs"
```

output: RED, J., B. BROWN, T. BLACK a kol.
We see that "a kol." (the Czech phrase) is used instead of "et al.".

Each entry is printed with hyphenation patterns locally set to the language of the cited document, as declared in the `lang` field.

OPmac knows only the `en`, `cs` and `sk` languages by default. If you are using the `etex.src` macros (i.e. you are using `etex`, `pdftex`, `xetex` or `luatex`) then you can declare new languages with `\isolangset{`⟨*long*⟩`}{`⟨*short*⟩`}`; for example, `\isolangset{espanol}{es}`. However, if you are using `csplain` or `pdfcsplain` then you must reinstall the format with new hyphenation patterns; see the file `hyphen.lan`. Each language has an explicitly declared number in this file. Use this number for declaration of the language identifier by `\sdef{lan:`⟨*number*⟩`}{`⟨*short*⟩`}`, for example `\sdef{lan:26}{es}` and `\sdef{lan:126}{es}`.

Sorting of entries. When `\usebib/c` is used then the order of the entries in the reference list is given by the order of `\cite` and `\nocite` in the document. If (more usually) `\usebib/s` is specified, then the order of the entries follows the ISO 690 standard: sort by the first author (last name, first names) alphabetically; if entries remain the same from this, then use the year of the edition (from older to newer). If both of these sort keys (first author and the year) are the same then the norm does not specify the ordering.

But ... what does *alphabetical ordering* mean? We can have many cited documents with authors from different parts of the world. We may need to sort names from ancient Babylon, but there is no standard for this. Sorting has standards for particular languages only. Thus, the idea of using the standard for sorting in the language of the actual document is bad because no language includes all characters used in possible author names.

This is the reason why I did not attempt to solve this problem; instead, I simply used the character-code sorting provided by `librarian.tex`. Of course, this may give bad results, but we can deal with exceptions when needed. For example, an entry with the author "Světla Čmejrková" is placed at the end of the reference list and this is wrong; "Č" must be sorted between "C" and "D" in Czech. But we can declare the `key` field. When this field is used, its value is used for sorting instead of the names in the `author` field. So,

```
author    = "Světla Čmejrková",
key       = "Czzmejrkova Svetla",
```

does the desired correction of sorting.

If someone needs the rule of automatically putting self-citations before all others, `key = "@"` can be added to all entries with his/her name. This works because the code of the `@` character is less than the codes of all alphabetic characters.

Writing bib-styles

Documentation for bib-style programmers can be found in the `opmac-bib.tex` file after its `\endinput`. The existing styles `opmac-bib-simple.tex` and `opmac-bib-iso690.tex` may be helpful for examples and inspiration.

The style file is read inside a TeX group when the `\usebib` macro is processed. The `*.bib` files are read in the same group and the reference list is printed afterwards. Then the group is closed. This means that all settings and definitions done in a style file are local to this group. The `\bibtexhook` macro is expanded immediately after the style file is read, but before processing any `*.bib` files. Users can set this macro in order to redefine some bib-style features. The macro is empty by default.

The bib-style file must define a `\print:`⟨*type*⟩ macro for each ⟨*type*⟩ of bibliographic entry. These macros are expanded to print an entry with the given ⟨*type*⟩. For example, `\print:book` is expanded when a processed entry has the type `@book`. You must use lowercase letters in the control sequence.

The programmer can use the helper macros `\bprinta` and `\bprintb`. Both have the same syntax:

```
\bprinta [⟨key⟩] {⟨if exists⟩} {⟨if doesn't exist⟩}
```

If the field given by the ⟨*key*⟩ exists then the ⟨*if exists*⟩ part is processed, else the ⟨*if doesn't exist*⟩ part. The first parameter of `\bprinta` can include the `*` character: this symbol is replaced by the value of the given field. The `*` character cannot be "hidden" in next level of braces `{...}`. You can use `\bprintb` instead of `\bprinta` with the same effect but use `##1` instead of `*` and this parameter can be hidden in braces — though nested macro calls need more hashes.

Example for printing an `@Book` entry:

```
\sdef{print:book}{%
 \bprinta [!author] {*\.\ }        {\bibwarning}%
 \bprintb [title]   {{\em##1}\.\ } {\bibwarning}%
 \bprinta [edition] {*~\mtext{bib.edition}.\ }{}%
 \bprinta [address] {*: }          {\bibwarning}%
 \bprinta [publisher] {*, }        {\bibwarning}%
 \bprinta [year]    {*.\ }         {\bibwarning}%
 \bprinta [isbn]    {ISBN~*.\ }    {\bibwarning}%
 \bprintb [url]     {\preurl\url{##1}. }      {}%
}
```

The list of authors is printed first. The exclamation mark before the key `author` means that the value of this field is printed by a special rule (using `\authorname` macro, see below). The list of authors is printed in place of `*`, followed by the "maybe dot" represented by the `\.` macro. This macro prints a dot only if the preceding character is not a dot, exclamation or question mark. Why do we need this? The name can be ended with a dot (due to abbreviating the first name, for example) and we do not want to print a second dot in such cases. A normal interword space (`\ `) follows after the "maybe dot". If the field `author` is missing then `\bibwarning` prints a warning about a missing mandatory field for type `@Book`.

Then the title is printed. It is printed in italics using `\em` macro. The "maybe dot" is used again, because the title might end with a question mark, for example. A space follows. `title` is another mandatory field, so again `\bibwarning` prints the warning if the field is missing.

The optional field `edition` follows. The text "edition" is appended but this text depends on the language. So, the `bib.edition` label is declared in the style file by:

```
% Multilinguals:    English   Czech    Slovak
\mtdef{bib.edition} {edition} {vydání} {vydanie}
```

As seen here, only the Czech, Slovak and English languages are provided by default. If you need to support another language then you need to add the phrases of the language like this:

```
\sdef{mt:bib.edition:es}{liberación}
```

Let us return to the example above. The mandatory field `address` follows. (Typically the city of the publisher is here, not a full address.) Then a colon and space are printed, then the mandatory `publisher` field, and a comma and space. The mandatory field `year` follows, then a period and space. The mandatory field `isbn` is prefixed by the text "ISBN" and a non-breaking space, and followed by a period and normal space. Finally, the optional field `url` is printed; the `\preurl` macro can print something before such a url.

The `\authorname` macro must be defined in the style file. This macro processes the authors' names. It is called for each author name (repeatedly, if there is more than one author in one `author` field). The following data are available in the macro body: `\Namecount` is the number of this author name in the `author` field, `0\namecount` expands to the number of all authors in the field, the `\Lastname` macro expands to the last name of the processed author and similarly with `\Firstname`, `\Von` and `\Junior`. Here is the definition from the "simple" style:

```
\def\authorname{%
  \ifnum\NameCount>1
     \ifnum0\namecount=\NameCount
        \mtext{bib.and}\else , \fi
  \else
     \ifx\dobibmark\undefined
        \edef\dobibmark{\Lastname}\fi
  \fi
  \bprintc\Firstname{* }\bprintc\Von{* }%
  \Lastname\bprintc\Junior{, *}%
}
```

We can see that the last name of the first author is saved to the `\dobibmark` macro (for further processing of non-numeric citation mark). Otherwise the comma+space is printed before the author, except for the last author, when `\mtext{bib.and}` is used instead. This expands to the "and" conjunction, according to the language of the cited document.

Next, the first name(s) is printed, then the "Von" part (if it exists), then the last name, and then the "Junior" part (if this exists, it means the "After" part of the name). The optional parts of the name are printed using the `\bprintc` macro. This prints nothing if its first parameter is empty, otherwise it prints its second parameter and replaces `*` by its first parameter.

The style must generate non-numeric citation marks too, but this is another story not shown here. Interested readers can see the existing implementations in the files `opmac-bib-simple.tex` and `opmac-bib-iso690.tex`.

More features

The bib-style files specify the format of each entry in a reference list, but only in the context of printing the fields in an (imaginary) infinite line. How to break this into paragraph lines, what indentation to use, where to place the citation mark (if any) etc. is another story. This is solved by the `\printbib` macro defined in OPmac itself. This macro is expanded before each entry and it has the following default definition:

```
\def\printbib{\hangindent=\iindent
 \ifx\citelinkA\empty
   \noindent\hskip\iindent\llap{[\the\bibnum] }%
 \else \noindent \fi
}
```

This means: The indentation is set by `\iindent` value and when `\nonumcitations` isn't used, the numbered citation mark `\the\bibnum` is printed by `\llap`. Else no citation mark is printed and only `\noindent` is processed.

This is a typical OPmac approach: it doesn't provide/document a plethora of options and doesn't define complicated macros. It gives a simple default macro with the idea: "hey, you can just redefine it if you want something else". This is an important difference between OPmac and most LaTeX packages or the ConTeXt format. You needn't remember (or repeatedly read in documentation) the syntax and values of dozens of options. Only one thing you need: to be able to create macros in TeX.

There are many variants of the `\printbib` macro on the OPmac tricks web page [7]. For example, OPmac trick 0040 [8] gives a recipe for indenting all entries by the width of the citation mark with the maximal number. OPmac trick 0041 [9] solves the analogous problem for non-numbered citation marks.

The default `\printbib` macro assumes that non-numbered citation marks are in "long" format with the full last name of the first author, and the reference list is ordered by this last name. This is a reason why these long citation marks are not printed in the reference list again. If you do need to (re)print these citation marks then you can follow OPmac trick 0096 [10].

The style files generate non-numbered citation marks in the format [Last name, Year] by default. If the `bibmark` field is present then the value from this field is used instead of the generated value. OPmac trick 0097 [11] shows how to convert these long non-numbered marks such as (Knuth, 1984) to the abbreviations like [Kn84] without any change in the style file or `*.bib` file. These abbreviations are sometimes part of citation rules.

There are special rules for grouping long non-numbered citation marks. For example, it isn't a good idea to repeat the full citation mark for multiple entries with the same author (Olšák, 1995, Olšák, 1997, Olšák, 2013). It is much better to print (Olšák, 1995, 1997, 2013). How to do this automatically using macros is described in OPmac trick 0035 [12]. The comma after the name can be removed with OPmac trick 0043 [13].

When you are using non-numbered citation marks, it might happen that two different entries have the same citation mark. OPmac trick 0098 [14] gives a recipe for diagnosing this problem. If such a situation occurs, the author can set a different citation mark using the `bibmark` field in `*.bib` file, for example.

When you are preparing a proceedings or a long monograph then you probably need to treat the citation marks and reference lists independently and locally in each part of the work (each article, each chapter, etc.). The citation marks and reference list in one part must be independent of the reference list in another part. How to handle this in a single document is solved in OPmac trick 0042 [15].

References

1. `http://petr.olsak.net/opmac-e.html`
2. `http://www.bibtex.org/`
3. `http://biblatex-biber.sourceforge.net/`
4. `https://www.ctan.org/pkg/librarian`
5. `http://petr.olsak.net/ctustyle-e.html`
6. `http://en.wikipedia.org/wiki/BibTeX`
7. `http://petr.olsak.net/opmac-tricks-e.html` = ⟨*opmac-tricks*⟩
8. ⟨*opmac-tricks*⟩`#bibnumindent`
9. ⟨*opmac-tricks*⟩`#bibmarkindent`
10. ⟨*opmac-tricks*⟩`#abib`
11. ⟨*opmac-tricks*⟩`#bibmark`
12. ⟨*opmac-tricks*⟩`#ccite`
13. ⟨*opmac-tricks*⟩`#modcite`
14. ⟨*opmac-tricks*⟩`#bibmarkcheck`
15. ⟨*opmac-tricks*⟩`#morebibs`

◇ Petr Olšák
Czech Technical University
in Prague
`http://petr.olsak.net`

Exploring \romannumeral and expansion

Joseph Wright

Abstract

The TeX \romannumeral primitive leads a double life. As well as its obvious use for making Roman numerals, it also offers a powerful method for controlling expansion. Here, I look at why this comes about, why we might want to use it, and give illustrative examples where only \romannumeral gives the results we want.

1 Expansion control

TeX is a macro expansion language, and so methods for manipulating exactly when tokens are expanded are a core part of its programmer toolbox. Experienced TeXers know that `\expandafter` will skip over *one* token and expand the next in the input stream, so for example

```
\def\foo{\baz}
\def\baz{a}
\expandafter\show\foo
```

gives

```
> \baz=macro:
->a.
\foo ->\baz
```

Another primitive that is used regularly to control expansion is `\edef`, which exhaustively expands everything in the input it is given. For example:

```
\def\foo{\baz}
\def\baz{a}
\edef\test{\foo}
\show\test
```

gives

```
> \test=macro:
->a.
```

The combination of `\expandafter` and `\edef`, along with the `\noexpand` primitive, can be used to carry out a wide range of reordering of TeX input. However, there are some provisos. The `\expandafter` primitive is itself expandable but only carries out *exactly one* expansion. Thus it cannot be used if we don't know exactly how many expansion might be needed.

On the other hand, `\edef` will completely expand input but is an assignment so cannot be used in an expansion context (for example, inside `\csname`, an assignment of a numerical register, *etc.*). Using `\edef` also gives us no (easy) way to stop an expansion part-way through some input. These cases need a different approach and take us away from primitives intended for expansion.

2 Enter \romannumeral

TeX's syntax for numbers (integers) allows an optional leading space (or spaces), optional leading sign (or signs), the integer itself and an optional trailing space. The number can be given literally (for example `1234`), can be the result of expansion or can be an 'internal integer'. The latter is, for example, what using a count register or ``` ` ``` syntax results in: a 'complete' number where TeX will not look for any further digits.

How does this help with expansion? TeX needs a number (as it understands them) in several places, for example after `\count` (to select a register) and `\number` (to produce a literal number from various forms of input).

TeX also needs a number after `\romannumeral`: this primitive is of course meant for conversion of that number into a Roman numeral! However, in contrast to `\number`, which produces some output in all cases, `\romannumeral` yields *nothing at all* if the number it is given to work with is *negative*. Using `\romannumeral` for expansion is all about exploiting this fact in combination with TeX's definition of a number.

With the simple input

```
\romannumeral-1%
```

TeX will continue expanding after the `1` character to search for a continuation of the number or an optional space. It will expand tokens as it goes but will stop (without error) at the first non-expandable non-digit. Thus, an indirect redefinition (of `\foo`) like this:

```
\def\demo#1{%
  \begingroup
    \toks0=\expandafter
      {\romannumeral-1#1}%
    \showthe\toks0 %
  \endgroup
}
\def\foo{\baz}
\def\baz{\def\foo{}}
\demo{\foo}
```

will give as a result the "interior" definition of `\foo` (defined in `\baz`):

```
> \def \foo {}.
```

(where TeX has added spaces in the usual way to delimit tokens in the `\show` output). Contrast this with the result of trying to use `\edef` here, for example:

```
\def\demo#1{%
  \begingroup
    \edef\temp{#1}%
    \toks0=\expandafter
```

```
      {\meaning\temp}%
    \showthe\toks0 %
  \endgroup
}
\def\foo{\baz}
\def\baz{\def\foo{}}
\demo{\foo}
```

which results in:

```
! TeX capacity exceeded, sorry
  [input stack size=5000].
\baz ->\def \foo
                {}
```

So, using `\romannumeral` for expansion looks useful but there's an issue: what if we supply additional digits? This probably won't be deliberate, but the input we want to expand might just start with some digits. The above will fail as such digits will be used as part of the number.

Happily, the syntax for an internal integer avoids this problem: TeX searches for an optional space but *not* for any more numerical input. The notation normally used to specify the internal integer is TeX's ` notation: `` `0`` or `` `\q`` are commonly used but have no special meaning.

```
\def\demo#1{%
  \toks0=\expandafter
    {\romannumeral-`0#1}%
  \showthe\toks0 %
}
\def\foo{\baz}
\def\baz{123\def\foo{}456}
\demo{\foo}
```

gives the desired

```
> 123\def \foo {}456.
```

3 In use

This '`\romannumeral` trick' is commonly used where it's desirable to provide a macro that will give a result in a known number of expansions. In general, with a template like

```
\def\demo#1{%
  \romannumeral-`0%
    % ... expandable code here ...
```

we can be sure that `\demo` will expand in exactly two steps, provided of course that the code we've supplied doesn't stop the expansion (we'll deal with that below).

The one thing that *will* disappear from the input when using `\romannumeral` expansion is a leading space: remember that TeX is looking for an optional trailing space to the integer we've used to trigger expansion. Luckily, it's rare to be worried about retaining leading spaces in the contexts where we might want to use this approach.

In fact, we can exploit the fact that TeX is looking for a space: deliberately inserting one can be used to halt expansion at a known point, leaving potentially-expandable material untouched. That's handy if we want to stop once we have a 'result'. That naturally leads to the question of how we can arrange to produce a 'result' that consists of unexpandable tokens without ending up stopping expansion. It turns out to be easy enough, but is best shown by an example, as follows.

4 Two examples

To show some practical uses for the `\romannumeral` trick, I'm going to take a couple examples based on some code in the expl3 language (LaTeX3 team, 2015). To keep a focus on what we want to think about, these are somewhat simplified from the 'parent' versions, and in particular will work with TeX90: no ε-TeX primitives are used, at the cost of dropping a few features from the actual expl3 code.

The first example is a macro to pick arbitrary cases from a list of possible integer values. The `\ifcase` primitive is of course extremely fast but becomes highly inconvenient when the values involved are not close to 0 or are spread out. The approach we can take is as follows:

```
\catcode`\@=11 %
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
\long\def\intcase#1#2#3{%
  \romannumeral-`0%
    \intcase@loop{#1}#2{#1}{#3}\stop
}
\long\def\intcase@loop#1#2#3{%
  \ifnum#1=#2 %
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
    {\intcase@end{#3}}
    {\intcase@loop{#1}}%
}
\long\def\intcase@end#1#2\stop{\space#1}
```

Here, the idea is that we don't know how many times we will need to apply the `\ifnum` test, so if the case statement needs to be fully expanded to a result, using `\expandafter` won't be practical. On the other hand, by using `\romannumeral` we can be sure that exactly two expansions of `\intcase` will leave the result (and no other tokens). This behaviour is going to be useful in the second example.

TEX provides us with the primitives \lowercase and \uppercase to carry out case changing. These primitives are not expandable, which makes using them a bit tricky. It turns out to be possible to implement fully-expandable case changing even for complex cases. Here, I'll set up a (much) simplified version that only works with 'text' input and will ignore any spaces.

The \LowerCase user-level macro shown here sets the \romannumeral expansion then starts off a loop. Notice that we have an end macro too followed by an empty brace group: this is going to allow us to keep expansion going for as long as we want.

```
\def\LowerCase#1{%
  \romannumeral-`0%
    \lowercase@loop#1\lowercase@end{}%
}
```

The first internal macro simply looks for the end of the loop, and either picks the end-of-loop or case-change helper.

```
\def\lowercase@loop#1{%
  \ifx#1\lowercase@end
    \lowercase@end
  \else
    \lowercase@change#1%
  \fi
}
```

To perform the actual case change, we can use the \intcase macro we just saw. This can be forced to yield a result *before* storing the value, which is a benefit here in terms of performance (we end up with fewer tokens), but also comes into play if the result we are providing needs to be examined by some other code (which might also be forcing expansion!).

Notice that I've used another \romannumeral to avoid a long \expandafter chain in this forced expansion.

```
\def\lowercase@change#1\fi{%
  \fi
  \expandafter\lowercase@store
    \expandafter{%
      \romannumeral-`0%
      \intcase{`#1}
        { {`A}{a}{`B}{b}{`C}{c}{`D}{d}
          {`E}{e}{`F}{f}{`G}{g}{`H}{h}
          {`I}{i}{`J}{j}{`K}{k}{`L}{l}
          {`M}{m}{`N}{n}{`O}{o}{`P}{p}
          {`Q}{q}{`R}{r}{`S}{s}{`T}{t}
          {`U}{u}{`V}{v}{`W}{w}{`X}{x}
          {`Y}{y}{`Z}{z} }
        {#1}%
    }%
}
```

Storing the result of a case change is done by using the marker end-of-loop macro to add the processed token to the growing result: keeping the number of tokens down means TEX is doing less work. Finishing the loop on the other hand needs the outer \romannumeral to terminate, which is done by inserting a space then the final result.

```
\def\lowercase@store#1#2\lowercase@end#3{%
  \lowercase@loop#2\lowercase@end{#3#1}%
}
\def\lowercase@end#1\fi#2{\fi\space#2}
```

This code can then be expanded in exactly two steps to a result

```
\toks0=\expandafter\expandafter\expandafter
  {\LowerCase{abgT&HYRI$*Z}}%
\showthe\toks0 %
...
> abgt&hyri$*z.
```

or indeed we could once again use \romannumeral

```
\toks0=\expandafter
  {\romannumeral-`0%
    \LowerCase{abgT&HYRI$*Z}}%
\showthe\toks0 %
...
> abgt&hyri$*z.
```

5 Conclusions

Using \romannumeral can offer expansion control that is otherwise difficult or impossible in TEX. Particularly when creating expandable function-like macros, it is an invaluable tool in a TEX programmer's arsenal.

References

LATEX3 team. "The expl3 programming language". http://ctan.org/pkg/l3kernel, 2015.

◇ Joseph Wright
Morning Star
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2.co.uk

The apnum package: Arbitrary precision numbers implemented in TeX macros

Petr Olšák

TeX doesn't provide a comfortable environment for calculations at the primitive level. There are the well-known commands `\advance`, `\multiply` and `\divide` but dividing two decimal numbers with these commands is a somewhat complicated task for macro programmers. The additional ε-TeX primitives `\numexpr` and `\dimexpr` do not make it easier. Of course, various LaTeX packages exist for more comfortable numerical calculations. None of them satisfied my needs so I decided to create my own solution: the `apnum.tex` package (`http://www.ctan.org/pkg/apnum`). You can set an arbitrary precision for the calculation when using this package. You can do addition, multiplication, division, power, square root, and evaluation of common functions sin, cos, tan, arcsin, arccos, arctan, exp and ln. The result is calculated with `\apFRAC` decimal positions after the decimal point. You can set this register to an arbitrary value. Of course, if you need thousands of decimal digits then you must wait a while. Nevertheless, optimization techniques were used when implementing algorithms.

The documentation is in the file `apnum.pdf`. It is not only the user-level documentation, but also detailed technical documentation is included. You can find the description of all internal macros and all the numerical algorithms used.

The expression scanner

After `\input apnum` in your document, you can use the macro `\evaldef`⟨*sequence*⟩`{`⟨*expression*⟩`}`. It makes comfortable calculation available. The ⟨*expression*⟩ can include binary operators `+`, `-`, `*`, `/` and `^` with the usual precedence. The operands are "numbers". Users can use parentheses `()` as usual. The result is stored to the ⟨*sequence*⟩ as a "literal macro". Examples:

```
\evaldef\A {2+4*(3+7)}
   % the macro \A is the result, 42
\evaldef\B {\the\pageno * \A}
   % \B is 42 times page number
\evaldef\C {123456789 * -123456789123456789}
   % \C is -15241578765432099750190521
\evaldef\D {1.23456789 + 12345678.9 - \A}
   % \D is 12345596.13456789
\evaldef\X {1/3}
   % \X is .33333333333333333333
```

The result of division doesn't have absolute precision; the number of digits after the decimal point is limited by the value of `\apFRAC`, which is 20 by default. Absolute precision is implemented when `+`, `-`, `*` and `^` operators are used. When using `/` or evaluating math functions like $\sin x$, only `\apFRAC` digits are calculated after the decimal point.

The operands in the ⟨*expression*⟩ are most simply numbers in the format

⟨*sign*⟩⟨*digits*⟩`.`⟨*digits*⟩

where optional ⟨*sign*⟩ is a sequence of `+` and/or `-` characters. A nonzero number is treated as negative if and only if there is an odd number of `-` signs. The first group or second group of decimal ⟨*digits*⟩ (but not both) can be empty. The decimal point is optional if the second group of ⟨*digits*⟩ is empty.

Alternatively, you can specify an operand in scientific notation in the format

⟨*sign*⟩⟨*digits*⟩`.`⟨*digits*⟩`E`⟨*e-sign*⟩⟨*digits*⟩

The sequence before `E` determines the mantissa and the sequence after `E` is the exponent. The ⟨*e-sign*⟩ is `+` or `-` or nothing. If you are using scientific notation of operands then the result (calculated by `\evaldef`) is usually in the same form. The reason is simple. If you want to calculate (for example) `3E+2000 * 5E+1300` then `apnum` will not waste time working with "full numbers" with a lot of digits (converted from scientific form) but calculates only `3*5=15` and the exponent of the result `3300` is appended. Much more information about scientific format is in the documentation of the `apnum` package.

The operands in the ⟨*expressions*⟩ can be any of the following:

- Numbers, as described above.
- `\the`⟨*register*⟩ or `\number`⟨*register*⟩. This allows accessing TeX register values.
- A macro which expands directly to a number. This allows working with "variables".
- A "function-like" macro which returns a value. This allows the implementation of functions. The identifier of a function-like macro can be followed by zero or more parameters, each of which must be enclosed in braces `{}`.

For instance, `\EXP` is a function-like macro. This macro has one parameter which is another (nested) ⟨*expression*⟩. The `\EXP` macro returns the value of the exponential function e^x where x is the given ⟨*expression*⟩. Example:

```
\def\X{.25}
\evaldef\A{\EXP{2*\X} - 1}
   % \A is the result of e^{2X} - 1
```

Users can define their own function-like macros; see the next section. The package apnum defines following function-like macros (with one parameter as nested ⟨*expression*⟩): \SQRT, \EXP, \LN, \SIN, \COS, \TAN, \ASIN, \ACOS, \ATAN. The meaning of these macros is clear from their names.

Note that you must use parentheses () for precedence settings in an ⟨*expression*⟩, but use braces {} as delimiters of parameters of function-like macros. The spaces in the ⟨*expression*⟩ are ignored. Example:

```
\def\A{20}
\evaldef\B{ 30*\SQRT{ \SIN{\PI/6} +
            1.12*\the\widowpenalty } / (4-\A) }
```

The evaluation of operators and function-like macros works at the main processor level of TeX. Unlike some comparable LaTeX packages, the apnum package doesn't support calculations at the expansion processor level only. The reason is calculation speed optimization.. Moreover, I wanted it to be possible to use apnum in classical TeX, without ε-TeX primitives. And without the ε-TeX primitives then expansion-level calculation is very complicated. So, I rejected the expansion-only calculation. Another significant advantage of this decision is related to the possibility of creating function-like macros by users: they need not worry about main processor vs. expansion processor evaluation when creating their own function-like macros.

In my opinion, a skillful macro programmer doesn't require expansion-level calculation. He/she can use

```
\evaldef\V{⟨expression⟩}\edef\foo{...\V...}
```

instead of \edef\foo{...⟨*expression*⟩...} whenever he/she needs to do this.

There are some side effects of \evaldef processing:

- The value of the \apSIGN register. It is set to -1, 0, 1 according to whether the result is negative, zero or positive.
- The internal macro \OUT is a copy of the result.

Creating function-like macros

Let us start with creating our own function-like macros. We must follow two rules:

- The first token must be \relax after the first level of expansion. This is a signal that this is a function-like macro and not a normal numerical constant. The expression scanner creates a new TeX group and executes the macro in it.
- The function-like macro must define the macro \OUT as the result of processing, as a number, and the \apSIGN register must be set to the sign of the result. The expression scanner takes control again and uses these values as one operand in the ⟨*expression*⟩ currently being processed.

Several examples of function-like macros follow.

Hyperbolic sine (and inverse). There are some well-known mathematical functions not predefined in the apnum package. I believe that remembering the name of such a function is not markedly easier than remembering its natural definition, and the latter is much more useful and educational. So, I left such work to users. For example the hyperbolic sine can be defined by

```
\def\SINH#1{\relax % mandatory \relax
   \evaldef\myE{\EXP{#1}}%
   \evaldef\OUT{ (\myE - 1/\myE) / 2 }%
}
```

This corresponds to the formula

$$\sinh x = \frac{e^x - e^{-x}}{2}.$$

First, the mandatory \relax is given in the macro. Then the value e^x is saved in a temporary macro \myE. We need not worry about a name conflict (\myE being used elsewhere) because the macro is processed in the TeX group. The final \evaldef gives the desired result (including \apSIGN setting).

A reader may have another idea:

```
\def\SINH#1{\relax
  \evaldef\OUT{(\EXP{#1} - \EXP{-(#1)})/2}}
```

This implementation of \SINH also works, but it is not optimal because the slow calculation of \EXP is done twice. The internal ⟨*expression*⟩ #1 must also be evaluated twice.

Another example is the inverse of hyperbolic sine:

```
\def\ASINH#1{\relax
  \evaldef\X{#1}\LN{\X+\SQRT{\X^2+1}}}
```

The following identity is used here

$$\sinh^{-1} x = \ln\left(x + \sqrt{x^2+1}\right).$$

Here, we do not need to explicitly define the \OUT macro because \LN is another function-like macro, so it does this work.

Sine of argument in degrees. The default function-like macros \SIN, \COS and \TAN expect their argument in radians, meaning they have a period 2π. Sometimes, it is useful to use degrees instead of radians. There is a simple way to define functions `\SINdeg`, `\COSdeg` and `\TANdeg` with argument in degrees:

```
\def\SINdeg#1{\relax \SIN{(\PI/180)*(#1)}}
\def\COSdeg#1{\relax \COS{(\PI/180)*(#1)}}
\def\TANdeg#1{\relax \TAN{(\PI/180)*(#1)}}
```

Note the parentheses around the `#1` argument. This is because the argument may be an expression with (say) an addition.

We have another problem: the values in degrees are typically expressed in sexagesimal numeral system (degrees, minutes, seconds). Thus we create the function-like macro `\DEG` to take the value in sexagesimal notation and return a normal decimal number. Namely:

```
\DEG{12;30'45.756''}% means
  % 12 degrees, 30 minutes, 45.756 seconds
\evaldef\a{\DEG{12;30'45.756''}} % = 12.51271
\evaldef\a{\DEG{12;30'}}%        % = 12.5
\evaldef\a{\DEG{12}}%            % = 12
\evaldef\a{\DEG{12.5}}%          % = 12.5
```

The conversion is done only if a semicolon is used after the degrees value. On the other hand, a dot means the normal decimal dot in the number. The symbol for minutes `'` and seconds `''` at the end of the value is optional, so `\DEG{12;30}` also returns `12.5`. The `\DEG` macro can be defined using this code:

```
\def\DEG#1{\relax \DEGa#1;''\relax}
\def\DEGa#1;#2'#3'#4\relax{%
  \evaldef\OUT{#1}%
  \ifnum\apSIGN<0 \def\tmps{-}%
  \else \def\tmps{+}\fi
  \DEGb#2;\relax{\OUT+\tmp/60}%
  \DEGb#3;\relax{\OUT+\tmp/3600}%
}
\def\DEGb#1;#2\relax#3{\edef\tmp{#1}%
  \ifx\tmp\empty \else
  \edef\tmp{\tmps\tmp}\evaldef\OUT{#3}\fi
}
```

The macro reads the argument using `\DEGa`, where `#1` is the degrees, `#2` minutes and `#3` seconds. The first `\evaldef` calculates degrees. If the result is negative, we need to subtract the possible minutes and seconds (`\tmps` includes `-`).

The `\DEGb` macro removes the semicolon and next value. The raw value is stored in `\tmp`. Finally, `\evaldef` adds the minutes part and seconds part to the result `\OUT`.

Given the `\DEG` macro, we can now define macros `\SINd`, `\COSd` and `\TANd`. They accept the sexagesimal notation in its argument. For example, `\SINd{12;30}` is the sine of 12 degrees and 30 minutes.

```
\def\SINd#1{\relax \SIN{(\PI/180)*\DEG{#1}}}
\def\COSd#1{\relax \COS{(\PI/180)*\DEG{#1}}}
\def\TANd#1{\relax \TAN{(\PI/180)*\DEG{#1}}}
```

If a semicolon is not present in the argument, it is processed as a normal ⟨*expression*⟩ in degrees.

Maximum. We create the function-like macro

`\MAX{`⟨*expression*⟩`,`⟨*expression*⟩`,...,`⟨*expression*⟩`}`

Thus, the argument of this `\MAX` macro is a comma-separated list of any number of ⟨*expression*⟩s (at least one ⟨*expression*⟩ is needed). The macro returns the maximum of all given ⟨*expression*⟩s.

We cannot use the `\ifnum` or `\ifdim` primitives for comparison of two values because these values can be very large or have a very small difference, for example the first difference may be at the 50th decimal digit after decimal point. The documentation `apnum.pdf` recommends to use the `\TEST` macro, which can be defined:

```
\def\TEST#1#2#3#4{%
  \evaldef\tmp{#1-(#3)}\ifnum\apSIGN #2 0 }
```

and used:

`\TEST {`⟨*expression1*⟩`}` ⟨*relation*⟩ `{`⟨*expression2*⟩`}`
  `\iftrue` ⟨*true part*⟩ `\else` ⟨*false part*⟩ `\fi`

where ⟨*relation*⟩ is one of the characters `<`, `>`, `=`. The implementation of the `\TEST` macro is based on subtraction of the given two ⟨*expression*⟩s and testing the resulting `\apSIGN`. Note that the space after zero in the `\TEST` definition is needed because it closes the scanning of the zero number.

The `\MAX` implementation looks like this:

```
\newcount\mynum
\def\TEST#1#2#3#4{%
  \evaldef\tmp{#1-(#3)}\ifnum\apSIGN #2 0 }
\def\MAX#1{\relax \MAXa#1,,}
\def\MAXa#1,{%
  \evaldef\maxOUT{#1}\mynum=\apSIGN \MAXb}
\def\MAXb#1,{%
  \ifx,#1,\let\OUT=\maxOUT \apSIGN=\mynum \else
    \evaldef\maxNEXT{#1}%
    \ifnum\apSIGN>\mynum
       \mynum=\apSIGN \let\maxOUT=\maxNEXT
    \else \TEST\maxNEXT>\maxOUT \iftrue
       \let\maxOUT=\maxNEXT \fi\fi
    \expandafter \MAXb \fi
}
```

The `\MAXa` macro prepares the first value into `\maxOUT` and the sign of this value is `\mynum`. Then `\MAXb` is called repeatedly until `#1` is empty. If `#1` is empty, the resulting `\apSIGN` is set from `\mynum` and `\OUT` is `\maxOUT`; else, the next expression `#1` is evaluated. If the sign of this partial result is greater than the sign of the current result, then we do not need to execute the `\TEST` macro and can simply set new `\maxOUT` and `\mynum`. Else `\TEST` is processed and new value of `\maxOUT` is set only if the new result is greater than the current result.

We can create the analogous macro `\MIN` without copying all this code. We can just define the macro `\mmREL` as `<` or `>` and then use this macro instead of the `>` character in the above code.

Linear interpolation. We can use "declaration macros" `\setF...\endF` to define the function `\F` via a table of values. For example:

```
\setF
  \F{2} = 15 ;
  \F{3} = 10 ;
  \F{8} = 11 ;
\endF
```

Some `\F{`⟨*expression*⟩`} =` ⟨*expression*⟩`;` entries are listed here. A finite number of values of the function `\F` is given this way. Next, we define the function-macro `\F{`⟨*expression*⟩`}` which returns the value of the given `\F` using linear interpolation.

For the sake of simplicity, we don't implement a sorting algorithm and instead assume that the input values are sorted by the user. The previous example complies with this condition because $2 < 3 < 8$. Next, we suppose that the function `\F` is undefined outside the boundary input values (i. e. outside the $[2, 8]$ interval in our example). If the user tries to evaluate `\F` outside this interval then an "out of range" message is printed.

The `\setF` macro saves the information to the `\Flist` macro. This is a list of input values, i. e. `2;3;8;` in our example. Moreover, the macros `\F:`⟨*number*⟩ are defined as the function value. The first entry has the number 1, second 2, etc. In our example, the `\F:1` macro is defined as `15`, `\F:2` as `10` and `\F:3` as `11`. The `\setF` macro is defined by this code:

```
\newcount\mynum
\def\setF{\mynum=0 \def\Flist{}\setFa}
\def\endF{end setF}
\def\setFa#1{%
  \ifx#1\endF \else \expandafter \setFb \fi
}
\def\setFb#1#2#3;{%
  \evaldef\X{#1}%
  \evaldef\Y{#3}%
  \advance\mynum by1
  \expandafter
    \edef\csname F:\the\mynum\endcsname{\Y}%
  \edef\Flist{\Flist\X;}%
  \setFa}
```

The function-like macro `\F` evaluates the input parameter $x =$ `\X` and scans the `\Flist` contents to find the sub-interval I of two consecutive input values where $x \in I$. The boundary values of the interval I are denoted by $a =$ `\A` and $b =$ `\B`, i. e. $I = [a, b)$. The values $F(a) =$ `\FA`, $F(b) =$ `\FB` are known and linear interpolation is applied:

$$\texttt{\textbackslash OUT} = F(x) = F(a) + \frac{(x-a)\big(F(b)-F(a)\big)}{b-a}$$

```
\def\TEST#1#2#3#4{%
  \evaldef\tmp{#1-(#3)}\ifnum\apSIGN #2 0 }
\def\F#1{\relax
  \evaldef\X{#1}%
  \expandafter\Fa\Flist;\endF}
\def\Fa#1;{%
  \TEST{#1}>\X \iftrue
    \Fe \expandafter \Fc \fi % out of range
  \def\A{#1}\mynum=0
  \Fb
}
\def\Fb#1;{%
  \advance\mynum by1
  \ifx;#1;% the last item
    \TEST\X=\A \iftrue
      \evaldef\OUT{\Xa}\else \Fe \fi % out range
    \expandafter \Fc \fi
  \TEST{#1}>\X \iftrue
    \def\B{#1}%
    \edef\FA{\csname F:\the\mynum\endcsname}%
    \advance\mynum by1
    \edef\FB{\csname F:\the\mynum\endcsname}%
    \evaldef\OUT{\FA+(\X-\A)*(\FB-\FA)/(\B-\A)}%
    \expandafter \Fc
  \fi
  \def\A{#1}%
  \Fb
}
\def\Fc#1\endF{}
\def\Fe{\def\OUT{0}\apSIGN=0
  \message{F{\X} OUT OF RANGE}}
```

Printing and evaluating from shared source

So far, we have seen how the `apnum` package evaluates ⟨*expression*⟩s. As of version 1.5, `apnum` is able to print the same ⟨*expression*⟩s in math mode. The format of printing is determined automatically and is close to mathematical tradition. This is done with the `\eprint{`⟨*expression*⟩`}{`⟨*declaration*⟩`}` macro. You can specify the identifiers of "variables" in the ⟨*declaration*⟩.

We illustrate this feature by defining a macro `\ep{`⟨*expression*⟩`}`. It prints the ⟨*expression*⟩ and then evaluates the same ⟨*expression*⟩, showing the value. The macros `\X`, `\Y` and `\Z` are variables for these examples.

```
\def\vars{\def\X{x}\def\Y{y}\def\Z{z}}
%
\def\ep#1{$\displaystyle
  \eprint{#1}\vars% printing
  \evaldef\OUT{#1}% evaluation
  \ROUND\OUT6%      round result to 6 digits
  \corrnum\OUT    % .digits -> 0.digits
  \ifx\XOUT\empty =\else\doteq\fi \OUT$}
```

We need to give values to the variables x, y, z before starting the experiment:

```
\def\X{0.51} \def\Y{-2.7}  \def\Z{17}
```

Now, let's apply the `\ep` macro in a variety of cases:

```
\ep{(\X^2+1)/((\X+1)*(\X-2))}
```

$$\frac{x^2+1}{(x+1)\cdot(x-2)} \doteq -0.560069$$

```
\ep{-((\X^2-1)/((\X+1)*(\X-1)))}
```

$$-\frac{x^2-1}{(x+1)\cdot(x-1)} = -1$$

```
\ep{\SIN{\Y}^2 + \COS{\Y}^2}
```

$$\sin^2 y + \cos^2 y \doteq 0.999999$$

```
\ep{\ASIN{\X} + \ATAN{\X+1}}
```

$$\arcsin x + \arctan(x+1) \doteq 1.521041$$

```
\ep{\SIN{\PI/4}}
```

$$\sin\frac{\pi}{4} \doteq 0.707106$$

```
\ep{\SQRT{2}/2}
```

$$\frac{\sqrt{2}}{2} \doteq 0.707106$$

```
\ep{\PI}
```

$$\pi \doteq 3.141592$$

```
\ep{\FAC{\Z}}
```

$$z\,! = 355687428096000$$

```
\ep{\SQRT{\iFLOOR{\Y}^2+1}}
```

$$\sqrt{\lfloor y\rfloor^2+1} \doteq 3.162277$$

```
\ep{\iFLOOR{\Y} + \iFRAC{\Y}}
```

$$\lfloor y\rfloor + \{y\} = -2.7$$

```
\ep{\LN{\X/\Y^2}+1}
```

$$\ln\frac{x}{y^2} + 1 \doteq -1.659848$$

```
\ep{(\X+\Y)*-3}
```

$$(x+y)\cdot(-3) = 6.57$$

```
\ep{-3*-(\X+\Y)}
```

$$-3\cdot(-(x+y)) = -6.57$$

```
\ep{\BINOM{5}{1}+\BINOM{5}{2}}
```

$$\binom{5}{1} + \binom{5}{2} = 15$$

```
\ep{2^5/2}
```

$$\frac{2^5}{2} = 16$$

```
\ep{4^3^2}
```

$$4^{3^2} = 262144$$

```
\ep{(4^3)^2}
```

$$\left(4^3\right)^2 = 4096$$

```
\ep{\EXP{\LN{2}+\LN{3}}}
```

$$\mathrm{e}^{\ln 2+\ln 3} \doteq 5.999999$$

Note that the `\eprint` macro does not insert redundant parentheses and follows traditional math typesetting. For example `\SIN{\X}^2` prints as $\sin^2 x$. On the other hand, new parentheses are sometimes needed, for example `-3*-(\X+\Y)` is printed in the form $-3\cdot(-(x+y))$.

About the implementation

The algorithms used are described in detail in the technical part of the `apnum.pdf` documentation. This section introduces only the basic ideas.

Expression interpreter. When I was young (about 15 years old) I was a participant in a hobby course on programming. We were working with a mainframe EC 1010. Our teacher taught me how to program an expression interpreter (with operators of various priorities) using stacks. My first implementation of this was in FORTRAN. Now, many years later, I was able to use this knowledge and implemented the expression scanner again, now in `apnum`. The `apnum` package implements the expression scanner in two steps: first the ⟨*expression*⟩ is converted to Polish notation and this format is used for evaluating (or printing) in the second step.

Basic operations. Addition, subtraction, multiplication and division are implemented similarly to the way pupils learn to do these operations in school. The main difference is the base of the number system used. Students use base 10, manipulating with the ten different digits of this system and drilling the "small multiplication table" up to 100. On the other hand, `apnum` uses a number system with base 10000, each "digit" has up to four decimal digits and TEX supports a "multiplication table" to 10^8 using the `\multiply` primitive. This is possible because the maximum number that can be represented in TeX registers is $2^{31} \doteq 2 \cdot 10^9$. For example, to multiply two ten digit numbers, pupils need to do 100 multiplications but TeX needs only 9 multiplications.

TeX can only do direct access to its memory using new macro definitions. But this is not a good approach for implementing "digits" values. I did many tests of various methods. I found that the linear access to the sequence of "digits" in the input stream is the most efficient. Data are expanded to the input stream and read again. One problem is that we have only one input stream, but we typically need to read digits from two sources. So `apnum` uses a special interleaved format for these calculations. The data are converted from human-readable form to this interleaved format when we need to convert pairs of four decimal digits to one internal "digit". Then the calculation is processed (typically in a loop) over this interleaved format.

The division algorithm is well known from school too: the "tail" of partially calculated remainders is constructed. The `apnum` package optimizes this processing if the divisor is only one "digit" (i. e. at most four decimal digits). Then the complexity of division depends linearly on the desired number of digits in the result. When the divisor has more "digits", then `apnum` uses the special interleaved data format mentioned above.

Many other optimizations were done. For example, suppose a big number with many digits is given in the parameter `#1` and a macro is written roughly like this:

```
\def\macro#1{%
  \ifA \ifB  do something{#1}%
       \else do something{#1}\fi
  \else do something{#1}\fi}
```

This `\macro` approach above is not a good idea. Why? Because the big parameter is expanded three times here, thus much data is skipped many times by `\if...\else...\fi` primitives. This is time-consuming. So it is much better to do `\def\tmp{#1}` at the beginning of `\macro` and then do skipping over `\tmp` only.

Mathematical functions. As a student I did a school assignment on "long numbers" on the mainframe installed at our university. Punch cards were used. I implemented addition, subtraction, multiplication and division. My dream was to continue with this work and implement classical math functions as well. But lack of time and the unsuitable technology was too much of barrier, so the dream wasn't realized at that time. But now, I returned to my student days and started to implement math functions in `apnum`. The dream has been fulfilled now.

Square root. One of the algorithms for computing square roots is similar to the division algorithm, but its complexity isn't linear with the number of desired digits in the result. I had started working on this algorithm on the mainframe as a student, but now, I decided to use Newton's method.

We need to find the first approximation x_0 of $\sqrt{a}$. Then the tangent line to the graph of the function $f(x) = x^2 - a$ is constructed in the point $[x_0, f(x_0)]$. The position of the tangent can be found using calculus. The intersection of this line with the x axis is the next approximation of $\sqrt{a}$. This step is repeated until the desired precision is reached. I decided to use the linear interpolation of the function $\sqrt{x}$ in the interval $[1, 100]$ for calculating the first approximation used by Newton's method. The linear interpolation uses known values in the points $1, 4, 9, 16, \ldots, 81, 100$. Only classical TeX operations (no `apnum` operations) are used for calculating the first approximation. If we need 20 digits in the result then 5 iterations of Newton's method is sufficient because the number of calculated digits is doubled in each iteration step and the linear interpolation starts with 1 digit calculated.

If the argument is outside the interval $[1, 100]$ then we can shift its decimal point by an even number M of positions. Then we do the calculation of square root. Finally, we shift the decimal point back by $M/2$ positions in the result. This idea is based on the fact that $\sqrt{100} = 10$.

Exponential. The well-known Taylor series is used in `apnum`:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

This series converges well for $|x| < 1$ because of the factorial in the denominators. But what to do if the argument x is outside $[-1, 1]$? First of all, negative arguments are converted to positive using identity $e^{-x} = 1/e^x$. If the argument $x \geq 4$ then we calculate $d = \lfloor x/\ln 10 \rfloor$ and use the identity

$$e^x = e^{x - d \cdot \ln 10} \cdot 10^d$$

This means that we need to calculate the exponential of the argument $x' \in [0, \ln 10] \subset [0, 4)$ and then we shift the decimal point of the result by d digits.

If the argument $x \in [1, 4)$ then we divide it by two or by four in order to have $x' \in [0, 1)$. Then we use the Taylor series mentioned above for x' and finally the result is $(e^{x'})^2$ if $x' = x/2$ or $((e^{x'})^2)^2$ if $x' = x/4$. This is based on the identity $e^{2x} = (e^x)^2$.

We need to do about 20 steps in the Taylor series for 20 digits of precision because $20! \approx 10^{19}$.

Logarithm. The following series derived from the inverse of the hyperbolic tangent is used:

$$\ln x = 2\,\tanh^{-1}\frac{x-1}{x+1} =$$
$$= 2\left(\frac{x-1}{x+1} + \frac{1}{3}\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}\left(\frac{x-1}{x+1}\right)^5 + \cdots\right).$$

The disadvantage of this series is that it converges well only for $x \approx 1$. But we are able to modify the argument so that it is approximately equal to one. First, we calculate $A = x/\exp(\widetilde{\ln x})$, where $\widetilde{\ln x}$ is an approximation of $\ln x$. We use linear interpolation. It is evident that $A \approx 1$ because $\exp(\ln x) = x$, if exact $\ln x$ is used. Next, we calculate $\ln A$ using the series above. Finally, $\ln x = \ln A + \widetilde{\ln x}$ because $x = A \cdot \exp(\widetilde{\ln x})$ and $\ln(ab) = \ln a + \ln b$.

This algorithm need be implemented only for $x \in [1, 10)$. If the argument is outside this interval, then we shift the decimal point by M positions and then calculate $x = x' \cdot 10^M$, $\ln x = \ln x' + M \cdot \ln 10$. The frequently used value $\ln 10$ is calculated to the needed precision only once and saved into memory.

Because the linear interpolation for $\widetilde{\ln x}$ differs from the exact result at the second decimal digit, the same is true for the difference between A and 1. Each step of the Taylor series improves precision by four digits because there are only odd powers.

Sine and cosine. The known Taylor series for sine and cosine are similar to the Taylor series for the exponential. So, we need to have the argument in the interval $[0, 1)$. We can shift it by a multiple of period (or half-period), but we need to know the constant π first. The `apnum` package calculates and saves π to 30 digits in its memory. If more precision is desired then π is re-calculated by the Chudnovsky formula. It converges very well, with 14 new exact digits per one step. It has only one problem: to calculate $\sqrt{640320}$. This constant is used in the Chudnovsky formula. So `apnum` stores the initial approximation for Newton's method (for $\sqrt{x}$ calculation) with 12 decimal digits for this special case. This saves several steps of Newton's method.

After the sine or cosine argument x is shifted by a half-period multiple, we have $x \in [0, \pi)$. If x is outside of $[0, \pi/2)$ then we can use the identities $\sin x = \sin(\pi - x)$ or $\cos x = -\cos(\pi - x)$. Now, we have a new argument $x \in [0, \pi/2)$. If x is outside the interval $[0, \pi/4)$ then we can use identities $\cos x = \sin(\pi/2 - x)$ or $\sin x = \cos(\pi/2 - x)$. The new argument is in the interval $[0, \pi/4) \subset [0, 1)$ and the Taylor series for sine or cosine can be used.

Inverse of tangent. The function $\tan^{-1} x = \arctan x$ is implemented by the series for the argument $1/x$:

$$\arctan\frac{1}{x} = \frac{x}{1+x^2} + \frac{2}{3}\frac{x}{(1+x^2)^2} +$$
$$+ \frac{2}{3}\frac{4}{5}\frac{x}{(1+x^2)^3} + \frac{2}{3}\frac{4}{5}\frac{6}{7}\frac{x}{(1+x^2)^4} + \cdots$$

It converges well for $x > 1$. If $x \in (0, 1)$ then we can use the identity $\arctan x = \pi/2 - \arctan 1/x$ and if the argument is negative we use the fact that the function is odd.

Other common mathematical functions can be expressed directly with the functions mentioned above.

The final joke

The `apnum` package uses only TeX primitives and the basic plain TeX macro `\newcount`. Thus, the package works in classical or extended TeX with any format. This is the general approach of almost all my macros. On the other hand, typical LaTeX packages require the LaTeX format and don't work with anything else. This is shown explicitly for example by `\NeedsTeXFormat{LaTeX2e}` in such macros. The LaTeX macros are usually a mix of TeX primitives and LaTeX constructs: a mix of `\def` and `\newcommand`, a mix of `\newcount` and `\newcounter`, a mix of `\advance` and `\addtocounter`, a mix of `\hbox` and `\mbox`, a mix of `\setbox` and `\sbox`, a mix of `\vrule` and `\rule` etc. Pure TeX macros (which can be used in plain TeX too) are infrequent in the LaTeX world, unfortunately (in my view).

So, I decided to put the following code at the end of my `apnum.tex`:

```
% please, don't remove this message
\ifx\documentclass\undefined \else
\message{WARNING: the author of apnum
  package recommends: Never use LaTeX.}\fi
```

Thus the above message is printed on the terminal and in the log file when LaTeX is used. This expresses my opinion about LaTeX. And I hope that this does not matter, because a typical LaTeX user reads neither the log file nor the terminal output, because plenty of useless information is printed there.

◇ Petr Olšák
Czech Technical University in Prague
`http://petr.olsak.net`

The Treasure Chest

The following is a list of selected new packages posted to CTAN (`http://ctan.org`) from October 2015 through March 2016, with descriptions based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. More information about any package can be found at `http://ctan.org/pkg/`*pkgname*. A few entries which the editors subjectively believe 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
`tugboat (at) tug dot org`

fonts

`cochineal` in `fonts`
Major extension of the Crimson fonts, including Greek and Cyrillic.

`crimson` in `fonts`
The original Crimson (a new old style design) font family.

`libertinus` in `fonts`
New math font and bug fixes for Libertine (see note in this issue).

`libertinust1math` in `fonts`
Extended Type 1 support for the new Libertinus.

`libertinegc` in `fonts`
Support for Greek and Cyrillic in Linux Libertine.

* `librebodoni` in `fonts`
The Libre Bodoni font family.

`miama` in `fonts`
The Miama Nueva handwriting/script font, with support for Latin, Cyrillic, and Greek.

* `nimbus15` in `fonts`
Support for the 2015 Nimbus fonts, including Greek and Cyrillic and a new weight and width for Courier.

* `noto` in `fonts`
Support for NotoSerif and NotoSans.

`yinit-otf` in `fonts`
OpenType version of the `yinit` fancy initials font.

graphics

`ellipse` in `graphics`
Draw ellipses and elliptical arcs using the standard `picture` environment.

`mparrows` in `graphics/metapost/contrib/macros`
Draw different types of arrowheads with MetaPost.

`tikz-feynman` in `graphics/pgf/contrib`
Draw Feynman diagrams with Ti*k*Z.

indexing

`upmendex` in `indexing`
Multi-lingual index processor supporting UTF-8 and Unicode sorting, mostly compatible with `makeindex`.

info

`einfuehrung2` in `info/examples`
Examples from the book *Einführung in LATEX*.

`formation-latex-ul` in `info`
Materials for a two-part introductory course in French.

`lshort/estonian` in `info`
Estonian translation of *The Not So Short* ...

`visualtikz` in `info`
Visual help for Ti*k*Z: an image per command or parameter.

language

`greektoni` in `language/greek`
Support for multi-accented Greek.

macros/generic

`gobble` in `macros/generic`
Remove following arguments, possibly optional.

* `luatex85` in `macros/generic`
Emulation of pdfTEX and other primitives changed in LuaTEX 0.85+ (see note in this issue).

`tex-ini-files` in `macros/generic`
Build TEX formats.

* `unicode-data` in `macros/generic`
Text files from the Unicode Consortium and TEX files to load them.

macros/latex/contrib

`adtrees` in `macros/latex/contrib`
Draw adpositional (natural language) trees, defined by Gobbo and Benini.

`asciilist` in `macros/latex/contrib`
Typeset lists without macros.

`bibletext` in `macros/latex/contrib`
Insert Bible passages by reference.

`bitpattern` in `macros/latex/contrib`
Typeset bit pattern diagrams for hardware, etc.

`bxdvidriver` in `macros/latex/contrib`
Enables specifying a driver option only for DVI output.

`bxpapersize` in `macros/latex/contrib`
Synchronize output and layout paper size.

`carbohydrates` in `macros/latex/contrib`
Draw carbohydrate molecules with `chemfig`.

`continue` in `macros/latex/contrib`
Print continuation marks on verso pages.

`delimseasy` in `macros/latex/contrib`
Easy commands for controlling size and boldness of delimiters.

`drawmatrix` in `macros/latex/contrib`
Visually represent matrices.

`econometrics` in `macros/latex/contrib`
Simplify notation for economic writing, based on the proposal by Abadir and Magnus.

`emisa` in `macros/latex/contrib`
Support for *EMISA* journal manuscripts.

`eq2db` in `macros/latex/contrib`
Convert self-contained `exerquiz` quiz to a server-side submission.

* `fixcmex` in `macros/latex/contrib`
Support scalable `cmex` fonts.

`ffslides` in `macros/latex/contrib`
Free-form slides based on the `article` class.

`getitems` in `macros/latex/contrib`
Gather items from a list environment.

`gitlog` in `macros/latex/contrib`
Typeset `git` changelogs.

`gloss-occitan` in `macros/latex/contrib`
Polyglossia support for Occitan.

`glossaries-extra` in `macros/latex/contrib`
Add-ons to the `glossaries` package.

`glossaries-swedish` in `macros/latex/contrib`
Swedish language support for `glossaries`.

`graphics-cfg` in `macros/latex/contrib`
Default driver options for `color` and `graphics`.

`iffont` in `macros/latex/contrib`
Conditionally load fonts with `fontspec`.

`keyvaltable` in `macros/latex/contrib`
Typeset tables with reusable layout.

`ksp-thesis` in `macros/latex/contrib`
Support KIT Scientific Publishing works.

`longfox` in `macros/latex/contrib`
Draw framed boxes with standard CSS attributes that can break over multiple pages.

`lroundrect` in `macros/latex/contrib`
LaTeX support for the MetaPost `roundrect`.

`mathpartir` in `macros/latex/contrib`
Typeset sequences of math formulas, e.g., for type inference rules.

`moodle` in `macros/latex/contrib`
Generate Moodle quizzes via LaTeX.

`multidef` in `macros/latex/contrib`
Define multiple macros with similar definitions.

`mynsfc` in `macros/latex/contrib`
XeLaTeX template for Natural Science Foundation of China proposals.

`nihbiosketch` in `macros/latex/contrib`
Class for 2015 NIH biosketches.

`normalcolor` in `macros/latex/contrib`
Save and restore a color.

`nucleardata` in `macros/latex/contrib`
Data about atomic nuclides.

`parades` in `macros/latex/contrib`
Tabulators and vertical paragraph spacing using a galley approach (see article in this issue).

`macros/latex/contrib/tkz/pgfornament`

`pgfornament` in `macros/latex/contrib/tkz`
Draw 196 Victorian ornaments with PGF/Ti*k*Z.

`prooftrees` in `macros/latex/contrib`
Forest-based symbolic logic proof trees.

`scrlttr2copy` in `macros/latex/contrib`
Creating copies of letters made with `scrlttr2`.

`seuthesisx` in `macros/latex/contrib`
Thesis class for Southeast University, Nanjing.

`signchart` in `macros/latex/contrib`
Typeset sign charts.

`simpler-wick` in `macros/latex/contrib`
Simpler Wick contractions.

`smartunits` in `macros/latex/contrib`
Convert between some metric and Imperial units.

`texvc` in `macros/latex/contrib`
Support MediaWiki LaTeX commands, for copy-and-paste of formulae from MediaWiki to documents.

`xassoccnt` in `macros/latex/contrib`
Step associated counters simultaneously.

`xduthesis` in `macros/latex/contrib`
XeLaTeX template for Xidian University theses.

`xsavebox` in `macros/latex/contrib`
Saving content without output duplication, using PDF Form OBjects.

`ycbook` in `macros/latex/contrib`
Versatile book class.

macros/latex/contrib/babel-contrib

`macedonian` in `m/l/c/babel-contrib`
Babel support for Macedonian Cyrillic.

`occitan` in `m/l/c/babel-contrib`
Babel support for Occitan.

macros/latex/contrib/beamer-contrib/themes

`beamer-verona` in `m/l/c/beamer-contrib/themes`
The ‘Verona’ Beamer theme by Till Tantau.

`beamercolorthemeowl` in `m/l/c/beamer-contrib/themes`
Color theme to maximize visibility.

`metropolis` in `m/l/c/beamer-contrib/themes`
The modern ‘Metropolis’ theme.

macros/latex/contrib/biblatex-contrib

`ecobiblatex` in `m/l/c/biblatex-contrib`
Global ecology and biogeography styles.

macros/luatex/latex

`arabluatex` in `macros/luatex/latex`
Generate Arabic from an ASCII transliteration, similar to ArabTeX.

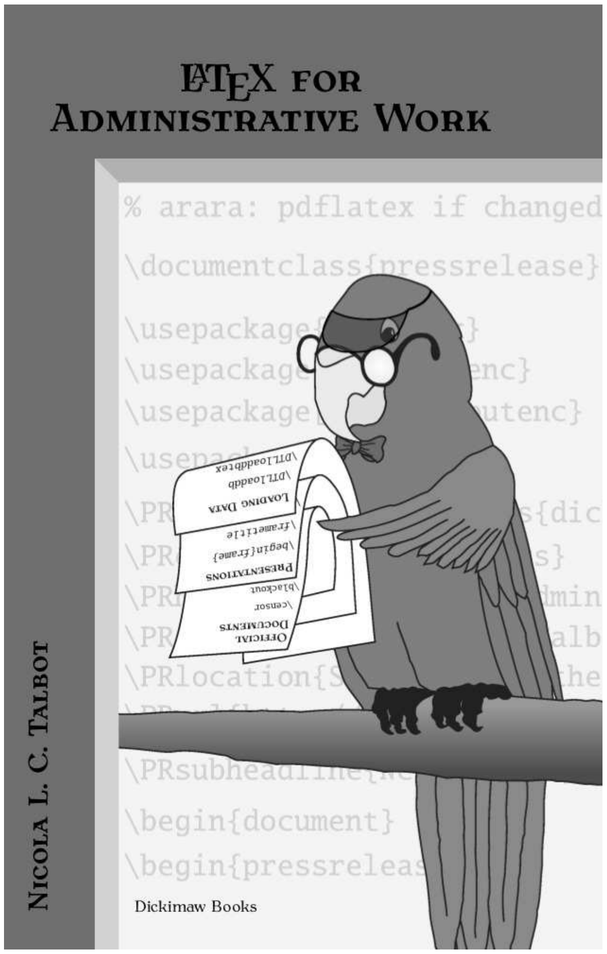
`luatex-def` in `macros/luatex/latex`
LuaTeX option file for the `color` and `graphics` packages.

Book reviews: *LaTeX for Administrative Work*

Boris Veytsman

Nicola L. C. Talbot, *LaTeX for Administrative Work*. Dickimaw Books, 2015, xiv+637 pp. Paperback, US$33.99. ISBN 978-1-909440-07-4.

Also available from `http://dickimaw-books.com/latex/admin/`; released under the GNU FDL.



This is the third volume in Nicola Talbot's LaTeX series published by her own Dickimaw Books (`http://dickimaw-books.com`). These books fill an important void in the LaTeX literature. A combined review of the first two volumes was published in *TUGboat* **35**:1 (2014) and on the TUG web site at `http://tug.org/books/reviews/tb109reviews-talbot.html`.

Let me start with a personal reminiscence. In the middle of the 1990s I decided to learn about all LaTeX packages and spent several days studying the documentation for all packages in the `macros/latex` directory on CTAN. This was an interesting exercise. Unfortunately, now, two decades and many talented CTAN contributors later, it would be almost impossible. At the very least, it would require a much longer time just to read through the descriptions of all the packages. On the one hand, this abundance is a good thing: it indicates the health of TeX and the LaTeX community. On the other hand, however, it poses a problem: how can one find the package that fulfills the given task? Probably many TeXnicians have had the embarrassing experience of re-inventing the wheel: writing code duplicating the functionality of an existing package (I myself certainly have!). The TeX Catalogue at `http://texcatalogue.ctan.org/` is of great help, but sometimes a keyword search is just not enough. Also, the Catalogue, by design, presents a neutral point of view. A practitioner often needs a guide to a curated collection of packages, where one can find the discussion about different ways to perform the task, advice about the best methods, pitfalls and bugs, etc. For many years the invaluable *LaTeX Companion* series used to be such a guide. However, since the last edition of the *Companions* there have been many changes in the LaTeX world. There are new tasks and new ways to deal with them. Thus new books are needed.

Nicola Talbot has a clear writing style and a great knowledge of TeX. She is the author of many useful packages. Thus she is well positioned to produce this guide to the modern LaTeX. The first and especially the second volume in the Dickimaw series look like an attempt to write such a guide. I am glad to report that the third volume is of the same high quality as the first two.

The title of the volume may require some explanation (actually the titles are the most problematic aspect of the series; only the first book fully corresponds to its title). It seems that by "administrative work" the author means repetitive tasks, those that could (and should) be automated. The book describes such tasks as processing data, creating correspondence (including mail merging), invoices, résumés, memos, minutes, presentations, exams and tests, business cards, flyers, letters, filled forms, charts, bar codes, as well as date and time formatting and even version control systems and their interaction with TeX. It has a separate chapter on online LaTeX editors and the ways to organize joint work on a LaTeX document with less TeX-savvy colleagues. For each task the author describes the CTAN packages that help to solve the problem. She tries to use only the packages that both exist in MiKTeX and TeX Live *and* have English documentation. The book discusses the main features of the packages, compares their strengths and gives recommendations about their usage. While reading the book I wished time and again that I had had it earlier: it would have saved me much time and effort.

The task-oriented approach adopted by Nicola Talbot has its advantages and disadvantages. Its main strength is that it gives the reader a hands-on understanding of the concepts, and a novice always knows why she is asked to learn what's at hand. This may be preferable here to a common alternate approach, where a book first describes a set of language constructs, and only then shows how these constructs are used. However, the present approach does sometimes prevent systematic exposition. For example, the book explains how TEX expansion works, with the (in)famous `\expandafter` in the chapter about reading and processing data from databases. A (quite interesting) lesson on package writing is placed in the chapter about making fillable forms in LATEX. Random number generation is discussed in the context of typesetting exams and tests, and so on. The book strives to be both a textbook and a reference book, but sometimes the different requirements of these different kinds of texts clash.

Perhaps it would make sense to repeat the notes about TEX and LATEX programming in a separate volume intended for aspiring TEXnicians. The advent of LATEX3 with its completely new programming paradigm makes such a book with chapters on TEX, LATEX 2_ε and LATEX3 quite appropriate.

It is rather difficult to write a book interesting for both novices and more experienced TEXnicians. Nicola Talbot has this rare skill: while the explanations are lucid and should be understandable to all users, some tricks (like the use of `\@afterheading` to overcome LATEX's tendency to redefine `\everypar`) are not trivial and could be quite illuminating even to seasoned TEX programmers.

It is inevitable that the choice of material reflects the author's tastes and views. For example, this book includes the chapter on résumés: is their typesetting logically an "administrative work"? In the same vein, *beamer* is even less related to administrative work and is well described in several recent books, so it might be omitted here (perhaps this chapter would be less out of place in the second volume of the series, intended for graduate students). Nevertheless the book does have an underlying idea, and one can elucidate the general principle the author used in the selection of the material.

The choice of packages also reflects the author's tastes. Most of the selected packages are quite important, like *datatool* and *datetime2* by the author herself, and *etoolbox* by Philipp Lehman and Joseph Wright. On the other hand, there are some unfortunate omissions. For example, the author describes the trusty but old `picture` environment, but does not mention the *pict2e* package, which eliminates many limitations mentioned in the book. Another unfortunate omission is the indispensable *latexdiff* program, which is quite useful in conjunction with the version control systems discussed in the book.

The typesetting of this book, as of the other books in the series, is very good. A book about a system for making beautiful texts should strive to be beautiful itself, and Nicola Talbot understands this well. The book has generous margins used for references and marking input and output. The author uses a nice typographic device to show the different levels of the text, marking them with different symbols. While this marking has been used in TEX books since *The TEXbook*, the idea to put the same symbols in the table of contents is relatively novel. It allows a reader to map her own route through the book according to the needs and the level of expertise.

The volume includes a bibliography of 151 titles, a short (in my opinion, rather too short) glossary, a summary of commands and environments with useful explanations and sources for each command, and a detailed index. Perhaps a running index for all three volumes in the series published so far would be a good addition. The book has many exercises ranging from simple ones to full blown projects.

The book is free (as in free speech): it is published under the GNU Free Documentation License and is available from the author's web site (`http://dickimaw-books.com`), along with the other volumes in the series. Instructions there also tell how to obtain a printed and bound copy.

My copy was made by Lightning Source Ltd.; the paper and binding are very good — the latter being rather important for a book of this length.

This volume, as well as the previous two volumes, deserves to be on a bookshelf of any serious LATEX user. The series can be of considerable interest to a TEX educator: it would be quite feasible to create courses for students at almost any level from novices to advanced users using different chapters of the books. The presence of exercises makes this task easier, and the free license allows students with limited means to download the textbooks.

◇ Boris Veytsman
Systems Biology School and Computational Materials Science Center, MS 6A2,
George Mason University,
Fairfax, VA 22030
`borisv (at) lk dot net`
`http://borisv.lk.net`

Book reviews: *Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander* by Vincent Lozano

Charles Thomas

Vincent Lozano, *Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander. Ou comment utiliser LaTeX quand on n'y connaît goutte*. Framasoft; 2013, 304pp, ill. Seconde édition. €15.00. ISBN 9791092674002. http://framabook.org/tout-sur-latex.

Vincent Lozano
Tout
ce que vous avez toujours voulu savoir
sur LaTeX
sans jamais oser le demander
Framabook

In regards to reviewing and judging books about LaTeX, the following was mentioned in [1]:

> 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.

This book, *All You Always Wanted to Know about LaTeX but Never Dared to Ask*, by Vincent Lozano, does well in both categories mentioned above. This book is in French and is divided into three parts. The book's layout is nicely designed and clean. The author sprinkles in some humor throughout the book to make it an enjoyable read.

The first part deals with introductory information concerning LaTeX. It starts off with the installation of LaTeX and goes on from there. One caveat: the installation information is most useful to those readers who are familiar with Unix. The rest of the first part tackles such things as creating documents, boxes, working with graphics and BibTeX, and given the francophone audience, the peculiarities of LaTeX documents in the French language. Though most of the information can be readily found in other introductory LaTeX books, for a French audience this is a very good starting point. The beginner is not inundated with so much information as to become overwhelmed. The author does a good job in striking the fine balance between "coolness" and usefulness for the beginner.

The material that makes the book of great worth to many readers is in the second part of the book. This part walks the reader through the process of creating the book itself. This section deals entirely with customizing the look of LaTeX documents. I highly recommend going through the examples and following the explanations. It will be a beneficial exercise for the reader.

The final part of the book contains appendices on generating pdfs and a listing of LaTeX symbols, which makes for a good reference.

Throughout the book, the author gives ample suggestions about packages that would be of benefit to the readers. These suggestions will be of great value to readers, especially those new to LaTeX as it gives them a good starting point to delve even deeper into LaTeX and reap the full extent of the benefits LaTeX has to offer.

I recommend this book and though it might not be of great value to an experienced LaTeX user, it is a good resource for the beginning and intermediate user. Given that it is in French, this book's usefulness is amplified even more. The book is available under the Libre Art License, so anyone who wishes to can read and benefit from it. Finally, 10% of sales of the print edition are donated to TUG.

References

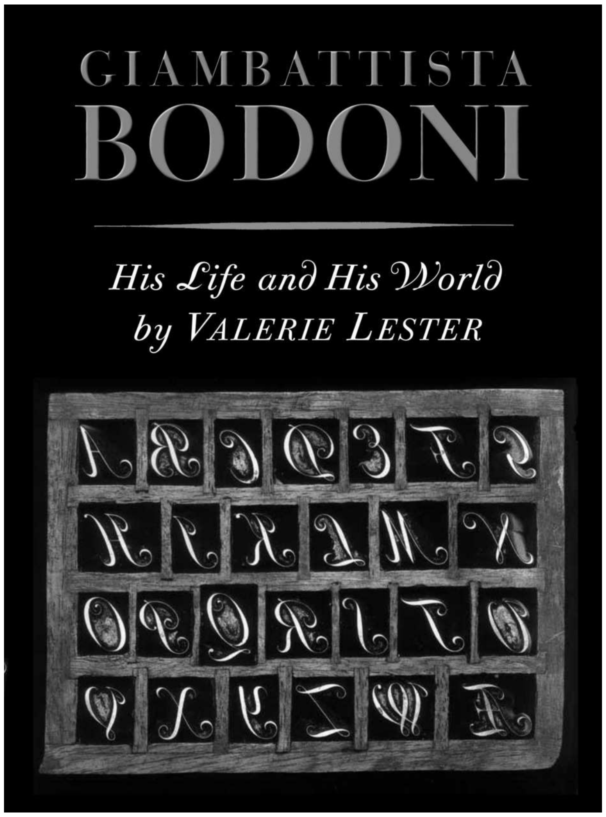
[1] Boris Veytsman, *Book Review: LaTeX and Friends*, *TUGboat*, Vol. 32 (2011), No. 3. http://tug.org/TUGboat/tb32-3/tb102reviews-ltxfriends.pdf

◇ Charles Thomas
Harris Corporation
sleonais (at) gmail dot com

Book reviews: *Giambattista Bodoni: His Life and His World* by Valerie Lester

Boris Veytsman

Valerie Lester, *Giambattista Bodoni: His Life and His World.* David R. Godine, Publisher; Boston, 2015, 288 pp., ill. US$40.00. ISBN 978-1567925289.



Rarely is a subtitle as precise as the one of this book by Valerie Lester. The reader who buys it will get exactly what is advertised: the description of Bodoni's life and world — no less, and no more.

A biographer of a scientist, an inventor or an artist has the following problem. How can one describe the *work* of the person without making the biography become a textbook in the corresponding field: relativity theory, or carburetor engines, or post-impressionism? Such a textbook inevitably is both too much and too little: the lay public is often lost in the technical details, while experts complain of the simplifications and pedestrian approach. It is very difficult to create a book about a complex subject which remains interesting for both the general public and specialists. Many biographers have been defeated by this task.

A way around this problem is to refuse to write too much about the intricacies of the person's work, focusing instead on his or her *life*, on the premise that a genius is interesting not only because of what he or she did, but also because of what he or she was. The popularity of biographies of Einstein, Turing or Dante shows that even people not much interested in physics, mathematics or medieval Italian poetry & politics may be fascinated by these remarkable lives. Moreover, biographies written in this style, as life stories rather than work stories, may also be of considerable value for experts. While the latter know only too well what the geniuses did, their backgrounds sometimes provide insights into *why* they did it. This is especially true with respect to artists' biographies, where the background is immensely relevant for the art. The best books of this kind give the reader the feeling of becoming acquainted with their subjects, of knowing *them* rather than knowing *about* them.

This book by Valerie Lester is an excellent example of this style of biography. It does not try to be a dissertation on typography. You will not find there a discussion about Garalde vs. Didone; nowhere does the author mention the difference between humanist and modern axes or other favorite topics of font cognoscenti. Instead she tells us about Giambattista Bodoni himself and his life.

Still, many details of his life as told by Valerie Lester are useful to understand Bodoni's contribution to the art of typography. For example, one of the striking features of Bodoni's legacy is the huge set of polyglot alphabets (one can enjoy the diversity of scripts in the reprint edition of his *Manuale Typographico*; see the review in *TUGboat*, Volume 32, No. 3, 2011 (`http://tug.org/books/reviews/tb102reviews-bodoni.html`). It is important to realize that Bodoni's interest in non-Latin scripts was caused by his youthful apprenticeship in the typography of *Sacra Congregatio de Propaganda Fide* (now *Congregatio pro Gentium Evangelizatione*), the missionary branch of the Roman Curia. Further, it is from the biography that one can conclude that the huge amount of ephemera published by Bodoni was the result of his position as the ducal printer in Parma as well as owner of a private press. It is fascinating to read about Bodoni's learned friends like Spanish diplomat José Nicolás de Azara chiding the typographer for wasting his talent on these works.

One may think the life of Giambattista Bodoni was rather uneventful: he spent most of his life as a well-respected printer in Parma, happily married (albeit child-less) and making many beautiful books, brochures, broadsides and other printed matter — as well as an immense quantity of punches for a huge variety of scripts. To the contrary, however, Valerie Lester tells a fascinating story of a great artist trying to live a fruitful life in a stormy world. This was a time of great political upheaval in Europe: the French revolution, Napoleonic wars, diplomatic intrigues. The Italian states around Bodoni changed their borders and allegiances. Bodoni responded to these calamities in his own way: he just worked more, producing beauty for the uncertain world. This is a powerful story with relevance even now.

The words "... and his world" in the subtitle of the book are also apt. Lester is genuinely interested in the details of Italian life in the late 18th–early 19th centuries, and this shows in many charming asides throughout the book. Speaking about Bodoni's youth, she discusses the food he could eat, and quotes a contemporary cookbook (p. 26):

> Although no portraits exist of Bodoni as a boy, those that appear after his arrival in Parma reveal a handsome man who steadily gains girth as he grows older. He clearly relishes his food; perhaps he enjoyed it too much, as gout would later suggest. One thing is certain: the food on offer in Saluzzo in his early years was of the highest quality, reflecting the cosmopolitan nature of the place.
>
> *Il cuoco piemontese perfezionato a Parigi* is a marvelous book, published in Turin in 1766, and giving the idea of the kind of food available to the growing Bodoni family. Its anonymous author (most likely a man) follows the order of the seasons, starting with spring, which he calls the most pleasant season of all, but sadly lacking in chicks and ducklings, small birds, vegetables, and fruit. However, young hares and rabbits, piglets, lambs, calves, and kids abound, and fortunately beef has no season. Nor do eels and frogs. (Frog fricassée is a featured recipe.) Freshwater and saltwater fish are available. Artichokes, asparagus, certain kinds of mushrooms, peas, cardoons, spinach, lettuce, turnip tops, sorrel, and chervil come into season, as do strawberries, gooseberries, and cherries.
>
> Summer sees an increase in poultry, game, and other birds, including songbirds. Beans, cauliflower, cabbage, and onions appear, along with peaches, plums, apricots, figs, currants, mulberries, melons, and pears. Autumn brings a bounty of fish, meat, cool weather vegetables and fruit, with the welcome addition of nuts, olives, and a huge variety of grapes. The lean winter months of December, January, and February, see an increase in the consumption of dairy products and dried and preserved food.

Then, with the same abundance reminding one of Dutch still life painters, Valerie Lester talks about coffee ("How tempting, but forbidden, the local cafés must have been for the young Bodoni"), wine, religion, festivities, and many, many more details of Italian life of these years.

Bodoni lives in Piazza di Spagna in Rome, which attracted many Roman girls and women. The author wonders whom he could meet there after a day of work — and this gives her a lead to talk about Roman women, their difference from contemporary Parisian women, the way the former and the latter used and use rouge and perfume, about Casanova, castrati, transvestites, harlequins, actors, etc., etc. We learn from the book how Italians at that time traveled, played, prayed, celebrated, grieved. Even the death of Bodoni was an occasion for Valerie Lester to talk about history (p. 180):

> In 1285, the citizens of Parma raised funds for the creation of a bell that could be heard as far away as Reggio Emilia, 26 kilometers distant. The first bell to be cast was not up to the challenge. A second bell, very beautiful, also failed the test, but in 1287, a third bell, Il Bajon, was struck, and this time the citizens were satisfied. (It is unclear, however, whether it actually made itself heard in Reggio Emilia.) Because of its enormous size, it was subject to rupture, and has been recast seven times since 1287. Strict regulations dictated that Il Bajon, the largest of the six bells in the cathedral tower, would only toll for the deaths of heads of state or members of Parma's most illustrious families. When the populace awoke on 30 November 1813 and heard the great bell tolling, it was clear that someone very important had died.

While the world of the book is full of things big and small, the author also loves to write about people. Bodoni's press in Parma became a favorite destination for many discerning visitors from Napoleon to traveling scholars. Even more people corresponded with Bodoni (a shrewd businessman, Bodoni sent a number of his books as gifts to various dignitaries including such distant ones as Benjamin Franklin, a printer himself). This gives Lester an opportunity to include portraits in her prose: sometimes long, sometimes short, but always vivid and interesting. Besides Giambattista and his wife Ghitta (a printer herself, who brilliantly finished her late husband's huge *Manuale*), the reader gets to know many contemporaries of the typographer: poets, printers, artists, kings, queens, princes, princesses, ministers, ambassadors, cardinals, apprentices, thieves and many others. Everyone is distinct and clear, like the characters in Bodoni's characteristic fonts.

The author's language is easy and conversational. Sometimes it becomes perhaps too chatty, and Valerie Lester's levity may occasionally seem overdone, as in "Saluzzo! Even the word sounds salutary, like a blessing for a sneeze." At times it seems that the copy editor missed an unfortunate word or phrase, for example, "Even though Bodoni's biographer, Giuseppe De Lama, presumably getting it straight from the horse's widow's mouth holds firm for [Bodoni's birthday on] 16 February [...]". Still, these occasions are rather rare, and the book overall is well written.

The book is well researched. It has 220 endnotes, quoting many sources, often not readily accessible, including the author's correspondence with a number of experts. The *Selected Bibliography* has 73 titles. This is very good for a popular biography. While the author, as mentioned above, does not try to go into the intricacies of typography, she adds four specialized appendices written by experts: *Cutting a Punch* and *Striking and Fitting a Matrix* by Stan Nelson, *Printing on a Hand Press* by Fred & Barbara Voltmer, and *The Trieste Leaf* by James Mosley (about the missing leaf from the 1788 edition of the *Manuale*). These appendices are well written and should be interesting both for the general public and for lovers of the art of typography.

Besides these appendices, bibliography and notes, there is a list of *Principal Characters*, divided into sections corresponding to the cities where the reader meets them, a nice map of Bodoni's Italy, and an index occupying ten pages of small print.

The book is abundant with illustrations. It shows leaves from Bodoni's books, from the books of his predecessors and competitors, as well as portraits, landscapes, and photographs. It has 28 plates (18 portraits and 10 plates of printed works), and a hundred more illustrations in the text, frontispiece, front and back end papers. Despite all this illustrative material, it does not give an impression of a coffee table book, intended to be leafed through rather than read: the pictures and photographs blend with the text. This is a notable achievement of the book designer, Jerry Kelly.

Speaking of this, I must say that a book about a great typographer places a high burden of expectation on the designer. It would be a disaster to publish a pedestrian-appearing book about a master of fine printing: too big a contrast. Fortunately, Jerry Kelly, a leading typographer, scholar and type designer himself (in October 2015 he received a prestigious Frederic W. Goudy Award; we have reviewed several of his books in *TUGboat*) was chosen, and was more than up to the task. Fittingly, he typeset the book in beautiful ITC Bodoni and Bauer Bodoni (we used Libre Bodoni from Impallari Type for this review). The proportions of type block, margins and marginal folios are exquisite, the typesetting is tasteful, and blending of text and illustrations is subtle. Kelly provided a number of illustrations for the book himself and drew the map of Bodoni's Italy. One feels that this was a work of love for him, a tribute to the classic of typography.

The book is printed on good stock and well bound. In our age of electronic books and sometimes shoddy typography it is a pleasure to see that the traditions of fine bookmaking are still alive.

The book was published by David R. Godine, a small independent company specializing in fine printing of "general interests for those with specific interests"; we have a note about it by Dave Walden in this issue.

I thoroughly enjoyed this book. I think it would find a good place on the bookshelf of anybody interested in the art of typography, the history of Europe during the French revolution, the past and present of Italy — and anybody with an appreciation for an interesting and beautiful book.

◇ Boris Veytsman
Systems Biology School and
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
`borisv (at) lk dot net`
`http://borisv.lk.net`

Note on the publisher of the Bodoni book: David R. Godine

David Walden

Near the end of the review in this issue of *Giambattista Bodoni: His Life and His World* by Valerie Lester, the reviewer emphasizes the high quality work of the typographer that went into the publication of this book. The reviewer also says of the book overall, "[I]t is a pleasure to see that the traditions of fine bookmaking are still alive."

It is also worth emphasizing that beyond the quality work of the author and the typographer, "fine bookmaking" also requires a special publisher, in this case the company of David R. Godine · Publisher.

David R. Godine · Publisher is well known in the fine printing and publishing world. This Boston-based company (`godine.com`) publishes 20 to 30 books a year. Occasionally they publish a volume on books and printing (an in-print list is at `godine.com/book-category/typography` — two of the books in that list are co-authored by Jerry Kelly, the designer of the Bodoni biography).

Another Godine-published book on printing (perhaps out of print) that I have on my own bookshelf is the 1989 reprint edition of Joseph Blumenthal's *The Printed Book in America*. Another Blumenthal related volume from Godine is *Art Of The Printed Book 1455–1955* (Masterpieces Of Typography Through Five Centuries From The Collections Of The Pierpont Morgan Library, New York, with An Essay By Joseph Blumenthal, 1973).

For publications such as the above, the Godine company won the 2014 Institutional Award from the American Printing History Association (`printinghistory.org/awards/godine`) — for "its services in advancing understanding of the history of printing and its allied arts."

From reading the Boston press over the decades, I know something more about David Godine and his dedication to fine book printing and publishing.

David Godine is himself a printer. A November 18, 2015, *Boston Globe* article shows Godine working at "at his Vandercook Proof Press" ("Beyond sales, Boston publisher's devotion speaks volumes" by Mark Shanahan, `tinyurl.com/jrlqn4t`).

Godine printed his first book while still a student at Dartmouth and started in business as a letterpress printer. In time, his business converted to being a publishing company rather than a printing company, with a continued narrow focus on fine book publishing. The *Globe* article quotes him as saying, "I'm interested in books as works of art." I myself exchanged a pair of emails with David Godine in May of 2012 (I don't actually know him personally) in which he stated that he had "a perspective that holds that perhaps the greatest books ever printed were issued in France between 1535 and 1560." Godine is also a collector of beautiful books. Some of his thinking in this area may be found in an except from a presentation he gave at Dartmouth in 1994: `tinyurl.com/gtrmnwx`

ART OF THE PRINTED BOOK
1455–1955
DAVID R. GODINE · PUBLISHER
FORTY YEARS OF INDEPENDENT PUBLISHING
1970–2010

Godine's company is clearly unusual. How many other publishers would emphasize their decades in business with an image of the president working with metal type? The 40-years-in-business poster shown on this page is of David Godine in his printer's apron, standing before a cabinet that holds cases of type, with (hanging) composing sticks and pica rules, and Godine looking down at a composing stick "to make sure that the sorts have been placed in it upside down and backwards with the nicks all aligned properly." (The poster was designed by Glenna Lang and provided to *TUGboat* by Godine associate publisher Sue Berger Ramin.)

The company now publishes books in six different series, each with its own imprint under the overall Godine name (`godine.com/imprints-and-series`). These "reflect the individual tastes and interests of its president and founder.” Perhaps such individualistic perspective and dedication is necessary to get books such as the Bodoni biography published in today’s world.

Valerie Lester, author of the Bodoni biography says, “David Godine is committed to publishing beautiful books, and his production standards are uncompromising. He’s a very hands-on publisher — he himself edited my book. I thoroughly enjoyed the experience of working with such an independent, idiosyncratic individual, and felt that the marriage of book and publisher was made in heaven!”

Carole Horne, general manager of the venerable Harvard Book Store (`harvard.com`), says, “David Godine is a legend in the book world. His dedication to producing beautifully-made books is only equalled by his extraordinary ability to find and publish important authors — among them the 2015 Literature Nobel Prize winner — and important subjects. Without his genius, the world of American publishing would be much poorer.”

◇ David Walden
`walden-family.com/texland`

Production notes

Karl Berry

As mentioned in the review of Lester’s book on Bodoni, we used Libre Bodoni from Impallari Type for the text. I need to thank Bob Tennent for creating the (LA)TEX support files for it, essentially instantaneously. Impallari Type (`impallari.com`) has many other excellent designs, and Bob has created support for nearly all their released text fonts, as well as fonts from many other sources (`ctan.org/author/id/tennent`) — all this in addition to his work with music typesetting (e.g., `ctan.org/pkg/musixtex`).

In the realm of font support, I’d also like to thank Michael Sharpe (`ctan.org/author/sharpe`), who has also created support for a plethora of high-quality fonts, including designing many new glyphs. Michael has also been instrumental in the latest releases of Lucida fonts through TUG (`tug.org/lucida`).

Finally, when looking for a so-called “modern” font to use for the review, as I have whenever looking for a font to use with TEX in past years, I’ve used Palle Jørgenson’s online font catalogue, `tug.dk/FontCatalogue`. It’s recently been extended to include a listing of fonts with OpenType support, and a listing of serif fonts by traditional classification (old-style, transitional, modern, slab). I can’t recommend this resource highly enough!

Die TEXnische Komödie 1/2016

Die TEXnische Komödie is the journal of DANTE e.V., the German-language TEX user group (`http://www.dante.de`). (Non-technical items are omitted.)

Joachim Lammarsch, Marion Lammarsch, Peter Breitenlohner; pp. 10–11

[Translated by the first author for this issue of *TUGboat*.]

Thomas Hilarius Meyer, LATEX in der Schule: Zeugniserstellung [LATEX at school: Creating school certificates]; pp. 12–20

In schools LATEX only plays a minor role. But some of these roles are notable. There are occasions when LATEX is more or less life-saving, for example when a huge number of identical, typographically-challenging documents need to be typeset: school certificates.

Rolf Niepraschk, Weg mit den Rändern! [Away with the margins!]; p. 21

A colleague of mine asked for help. His graphics, created with MS Excel were not satisfying: the surrounding frame was much too large, and had unnecessary white areas. Furthermore the requirement from his publisher was that it had to be exactly 85mm wide. With a small LATEX document I was able to help him.

Herbert Voss, Listen mit geschweiften Klammern markieren [Marking lists with braces]; pp. 22–23

It may sometimes be useful to group similar entries within environments like `itemize`, `description` or `enumerate` with braces and a corresponding label.

Herbert Voss, BibTEX-Felder auslesen [Reading BibTEX fields]; p. 24

The `biblatex` package defines a few macros to extract the contents of BibTEX database entries.

Herbert Voss, Spezielle Gleitumgebung [Special float environments]; pp. 25–26

Usually one uses the `float` package to define a new float environment. If the new environment is to have a special layout, e.g. a frame, the `floatrow` package may be helpful.

Herbert Voss, Eigene Beschnittmarken erstellen [Creating one’s own crop marks]; p. 27

The `crop` package is a good choice when crop marks are to be provided. It allows the author to get an overview of how the text area looks compared to the margin, while the receiving printer can use the crop marks for automated alignment of the print.

[Received from Herbert Voß.]

MAPS 46 (2015)

MAPS is the publication of NTG, the Dutch language TEX user group (`http://www.ntg.nl`).

Michael Guravage, Redactioneel [From the editor]; pp. 1–2

Erik Frambach, Memories of Kees; pp. 3–4
[Printed in *TUGboat* 36:2.]

Frans Absil, TEXShop review; pp. 5–7
[Reprinted in this issue of *TUGboat.*]

Willi Egger, TextMate; pp. 8–12

When editing text or code on a Mac, TextMate is an excellent choice. It offers an abundance of features which makes it a great editor for TEX as well as the macro packages LATEX and ConTEXt. Editing HTML, XML and CSS is also supported, as are many other programming languages. Specially tagged texts are quick and easy to write, thanks to autocompletion and placing of start and end tags. Further, the editor supports "projects" — a sidebar to the editing window that shows files belonging to the project. Another useful feature is the column selection method and the (near) end of line selection. TextMate offers a clipboard history, so multiple items can be retrieved. TextMate can be customized to a very large extent.

Sytse Knypstra, TEXworks; pp. 13–15
[Translated by the author for this issue of *TUGboat.*]

Siep Kroonenberg, TEXStudio: speciaal voor LATEX starters [TEXStudio: Especially for LATEX newbies]; pp. 16–22
[Revised and translated by the author for this issue of *TUGboat.*]

Kees van der Laan, `PSlib.eps` Catalogue, preliminary and abridged version; pp. 23–86

A selection of PostScript definitions collected in my `PSlib.eps` library and documented as e-book catalogue is presented. Now and then variant pictures have been included from `pic.dat` which comes with `Blue.tex`. Old Metafont code has been included which may be useful for MetaPost programmers. Variants of pictures enriched by post-processing in Photoshop show other possibilities. Escher's doughnut is a teaser which has to be done in MetaPost. Next to `PSlib.eps` comes the file `PDFsfromPSlib`, which contains the pictures in `.pdf` format. The complete `PSlib.eps/PDFsfromPSlib` as well as the catalogue as e-book, will be released on occasion of NTG's 25th anniversary which will be celebrated in the fall of 2014, on `www.ntg.nl`. A prerelease will be offered on the GUST file server. The (static) library for TEX-alone pictures, `pic.dat`, packaged with `Blue.tex`, will be redistributed as well.

Kees van der Laan, Spirals in PostScript; pp. 87–97

Curves specified in polar coordinates can be elegantly programmed in PostScript with the `rotate` command which performs rotations in user space. This has been shown for the Cardioid, the Limaçon, the Lemniscate, the Archimedes and the Growth spiral. The TEX Gyre logo has been analyzed and imitated in PostScript. Printing of text along spiral-like belts on a sphere in the projection plane has been done, yielding poor man's typesetting text on a sphere in projection.

Hans Hagen, SciTE; pp. 98–100

The SciTE editor is now about 15 years old, and still one of the nicest around. This editor is a wrapper around the `scintilla` editor framework. It is available for free for Windows and Linux, and there is a relatively cheap version for Mac OS X.

Hans Hagen, Lua in MetaPost; pp. 101–108

How to embed Lua code in MetaPost source, to be immediately executed. For example, for generating graphics from external data.

[Received from Wybo Dekker.]

Eutypon 34–35, October 2015

Eutypon is the journal of the Greek TEX Friends (`http://www.eutypon.gr`).

In memoriam: Hermann Zapf (1918–2015); pp. 1–13

Texts by Donald E. Knuth, Kaveh Bazargan, Adam Twardoch, Nadine Chahine (*in English*), and George D. Matthiopoulos (*in Greek with English abstract*).

In memoriam: Pierre A. MacKay (1933–2015); pp. 15–19

Texts by Diana Gilliland Wright and Lawrence J. Bliquez. (*In English.*)

Donald J. Mastronarde, GreekKeys: Keyboards and fonts for specialized scholarly use; pp. 21–28

GreekKeys is a custom polytonic Greek keyboard program with accompanying fonts that has been useful — for more than three decades now — to scholars, teachers, and students of the ancient and medieval Greek worlds. GreekKeys fonts and keyboards provide easy access to many specialized characters (e.g., for metrics, epigraphy, and papyrology) that are absent from most system fonts and that would otherwise have to be entered in roundabout or obscure ways. The latest version (GreekKeys 2015) runs both under Mac OS X and Microsoft Windows, and it includes four different Unicode-encoded OpenType fonts: New Athena Unicode, AtticaU, KadmosU, and BosporosU. (*Article in English.*)

Apostolos Syropoulos, Graphics with the Ti*k*Z/PGF package; pp. 29–43

Among all solutions for the creation of linear graphics, the author has found that the Ti*k*Z/PGF package is the best for use with Unicode-aware typesetting systems like XƎLATEX and LuaTEX. The article describes the basic principles of use of Ti*k*Z/PGF with examples taken from graphs the author created for his own publications. (*Article in Greek with English abstract.*)

George Matthiopoulos, The Archives of Design of Greece; pp. 45–50

The Archives of Design of Greece were founded in 2012 as a non-profit corporation, with the objectives of collecting, indexing and classifying all archival material related to Visual Communication in Greece. The Archives are already open to the public through the website `www.archivesofdesign.gr`. The first public presentation of the Archives was done at the House of the Onassis Foundation in Athens on October 22, 2015. (*Article in Greek with English abstract.*)

Apostolos Syropoulos, TEXniques: Rotations with Rubik's cube; pp. 51–56

One note explains how to verify with LATEX if a particular command or application exists in the operating system, and then to take appropriate action with LATEX. Another note explains how to make TEX convert token sequences into something different, e.g., to convert `D2` into two printed D's. (*Article in Greek.*)

Dimitrios Filippou, Book presentations; pp. 57–58

Short appraisal of two books:
(i) Dimitris Legakis, *For a Publishers' Archive of Typography, Design, Graphic Arts and Advertising*, The Archives of Design of Greece, Athens 2015 (book in Greek); and
(ii) Stefan Kottwitz, *LATEX Cookbook*, Packt Publishing, Birmingham, UK 2015. (*Article in Greek.*)

[Received from Dimitrios Filippou
and Apostolos Syropoulos.]

TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:
`http://tug.org/instmem.html`
Thanks to all members for their support!

American Mathematical Society, *Providence, Rhode Island*

Aware Software, Inc., *Midland Park, New Jersey*

Center for Computing Sciences, *Bowie, Maryland*

CSTUG, *Praha, Czech Republic*

Fermilab, *Batavia, Illinois*

Google, *San Francisco, California*

IBM Corporation, T J Watson Research Center, *Yorktown, New York*

Institute for Defense Analyses, Center for Communications Research, *Princeton, New Jersey*

Maluhy & Co., *São Paulo, Brazil*

Marquette University, *Milwaukee, Wisconsin*

Masaryk University, Faculty of Informatics, *Brno, Czech Republic*

MOSEK ApS, *Copenhagen, Denmark*

New York University, Academic Computing Facility, *New York, New York*

Overleaf, *London, UK*

River Valley Technologies, *Trivandrum, India*

ShareLaTeX, *United Kingdom*

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*

TNQ, *Chennai, India*

University College, Cork, Computer Centre, *Cork, Ireland*

Université Laval, *Ste-Foy, Québec, Canada*

University of Cambridge, Centre for Mathematical Sciences, *Cambridge, United Kingdom*

University of Ontario, Institute of Technology, *Oshawa, Ontario, Canada*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

VTEX UAB, *Vilnius, Lithuania*

TUG financial statements for 2015

Klaus Höppner, TUG treasurer

The financial statements for 2015 have been reviewed by the TUG board but have not been audited. As a US tax-exempt organization, TUG's annual information returns are publicly available on our web site: `http://tug.org/tax-exempt`.

Revenue (income) highlights

Membership dues revenue was slightly up in 2015 compared to 2014. Product sales returned to normal, after a spike in 2014 due to a large Lucida site license. The annual conference had a small margin, due to good attendance and DANTE e.V. support. Other income categories were close to steady. Overall, 2015 income was down 10%.

Cost of Goods Sold and Expenses highlights, and the bottom line

Payroll, *TUGboat*, DVD production, and other overhead continue to be the major expense items. Most were less than budgeted; overall, 2015 COGS was down about 16% from 2014, while office overhead was up slightly due to election costs.

The "prior year adjustment" compensates for estimates made in closing the books for the prior year; in 2015, no adjustment was needed.

The bottom line for 2015 was positive: a little more than $5,300.

Balance sheet highlights

TUG's end-of-year asset total is up around $1,800 (not quite 1%) in 2015 compared to 2014.

Committed Funds are reserved for designated projects: LaTeX, CTAN, the TeX development fund, and others (`http://tug.org/donate`). Incoming donations are allocated accordingly and disbursed as the projects progress. TUG charges no overhead for administering these funds.

The Prepaid Member Income category is member dues that were paid in earlier years for the current year (and beyond). Most of this liability (the 2015 portion) was converted into regular Membership Dues in January of 2015.

The payroll liabilities are for 2015 state and federal taxes due January 15, 2016.

Summary

Membership fees remain unchanged; the last general increase was in 2010. We ended 2015 with 20 more members than in 2014. TUG remains financially solid as we enter another year.

TUG 12/31/2015 (vs. 2014) Revenue, Expense

	Dec 31, 15	Dec 31, 14
ORDINARY INCOME/EXPENSE		
Income		
Membership Dues	92,455	91,780
Product Sales	5,736	13,529
Contributions Income	8,320	8,776
Annual Conference	1,837	8,720
Interest Income	484	425
Advertising Income	320	390
Services Income	2,616	671
Total Income	111,768	124,292
Cost of Goods Sold		
Membership Drive		(256)
TUGboat Prod/Mailing	(17,722)	(18,703)
Software Prod/Mailing	(3,200)	(3,076)
Postage/Delivery - Members	(2,147)	(2,294)
Lucida Sales to B&H	(2,195)	(5,993)
Member Renewal	(412)	(406)
Total COGS	(25,677)	(30,727)
Gross Profit	86,091	93,566
Expense		
Contributions made by TUG	(2,000)	(2,000)
Office Overhead	(15,444)	(13,134)
Payroll Expense	(63,256)	(64,752)
Total Expense	(80,700)	(79,886)
Net Ordinary Income	5,391	13,680
OTHER INCOME/EXPENSE		
Prior year adjust		423
Other Expense		(106)
Net Other Income		317
NET INCOME	5,391	13,997

TUG 12/31/2015 (vs. 2014) Balance Sheet

	Dec 31, 15	Dec 31, 14
ASSETS		
Current Assets		
Total Checking/Savings	205,582	201,400
Accounts Receivable	300	2,650
Total Current Assets	205,882	204,050
LIABILITIES & EQUITY		
Current Liabilities		
Committed Funds	31,248	30,837
Administrative Services	1,528	1,920
Deferred Contributions		45
Prepaid Member Income	4,085	7,610
Payroll Liabilities	1,087	1,094
Total Current Liabilities	37,948	41,507
Equity		
Unrestricted	162,543	148,546
Net Income	5,391	13,997
Total Equity	167,934	162,543
TOTAL LIABILITIES & EQUITY	205,882	204,050

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 `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`
We are your macro specialists for TEX or LATEX fine typography specs beyond those of the average LATEX macro package. If you use XETEX, 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.

Latchman, David
4113 Planz Road Apt. C
Bakersfield, CA 93309-5935
+1 518-951-8786
Email: `david.latchman (at) texnical-designs.com`
Web: `http://www.texnical-designs.com`
LATEX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized packages to meet your needs.

Call or email to discuss your project or visit my website for further details.

Peter, Steve
+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.

Sievers, Martin
Im Alten Garten 5
54296 Trier, Germany
+49 651 4936567-0
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`
As a mathematician with more than 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.

Sofka, Michael
8 Providence St.
Albany, NY 12203
+1 518 331-3457
Email: `michael.sofka (at) gmail.com`
Skilled, personalized TEX and LATEX consulting and programming services.

I offer over 25 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in TEX and LATEX: Automated document conversion; Programming in Perl, C, C++ and other languages; Writing and customizing macro packages in TEX or LATEX; Generating custom output in PDF, HTML and XML; Data format conversion; Databases.

If you have a specialized TEX or LATEX need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss 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 nineteen years of experience in TEX and three decades 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.

Webley, Jonathan
21 West Kilbride Road
Dalry, North Ayrshire, KA24 5DZ, UK
01294538225
Email: `jonathan.webley (at) gmail.com`
I specialize in math, physics and IT. However, I'm comfortable with most other science, engineering and technical material and I'm willing to undertake most LATEX work. I'm good with equations and tricky tables. I can also proofread and copy-edit if required. I've done hundreds of papers for journals over the years. Samples of work can be supplied on request.

Calendar

2016

Apr 8	**TUG 2016** deadline for bursary applications. `tug.org/tug2016`
Apr 29 – May 3	BachoTEX 2016, "Convergence — TEX, get out of the closet!" 24th BachoTEX Conference, Bachotek, Poland. `www.gust.org.pl/bachotex/2016-en`
May 1	**TUG 2016** deadline for presentation proposals. `tug.org/tug2016`
May 12 – 14	TYPO Berlin 2016, "Beyond Design", Berlin, Germany. `typotalks.com/berlin`
Jun 5 – Aug 5	Rare Book School, University of Virginia, Charlottesville, Virginia. Many one-week courses on type, bookmaking, printing, and related topics. `www.rarebookschool.org/schedule`
Jun 8 – 10	The Fourteenth International Conference on New Directions in the Humanities (formerly Books, Publishing, and Libraries), "The Event of the Book", University of Illinois at Chicago, Illinois. `thehumanities.com/2016-conference`
Jul 4 – 7	Book history workshop, École de l'institut d'histoire du livre, Lyon, France. `ihl.enssib.fr`
Jul 5 – 9	The 6th International Conference on Typography and Visual Communication (ICTVC), "Against lethe . . . ", Thessaloniki, Greece. `www.ictvc.org`
Jul 12 – 16	Digital Humanities 2016, Alliance of Digital Humanities Organizations, "Digital Identities: the Past and the Future", Kraków, Poland. `dh2016.org`
Jul 18 – 22	SHARP 2016, "The Generation and Regeneration of Books". Society for the History of Authorship, Reading & Publishing, "Languages of the Book"/ "Les langues du livre". Paris, France. `www.sharpparis2016.com`

TUG 2016
Toronto, Canada.

Jul 23, 24	Optional pre-conference tours. `tug.org/tug2016/excursions.html`
Jul 24	Evening reception and registration.
Jul 25 – 27	The 37th annual meeting of the TEX Users Group. Presentations covering the TEX world. `tug.org/tug2016`
Jul 28	Typographic excursions and banquet.
Jul 29	Optional post-conference tour [potential]

Jul 24 – 28	SIGGRAPH 2016, "Render the Possibilities", Anaheim, California. `s2016.siggraph.org`
Aug 1 – 5	Balisage: The Markup Conference, North Bethesda, Maryland. `www.balisage.net`
Aug 24 – 28	TypeCon 2016, "Resound", Seattle, Washington. `typecon.com`
Sep 11 – 16	XML Summer School, St Edmund Hall, Oxford University, Oxford, UK. `xmlsummerschool.com`
Sep 13 – 16	ACM Symposium on Document Engineering, Vienna, Austria. `www.doceng2016.org`
Sep 13 – 17	Association Typographique Internationale (ATypI) annual conference, "Convergence", Warsaw, Poland. `www.atypi.org`
Sep 16	The Updike Prize for Student Type Design, application deadline, 5:00 p.m. EST. `www.provlib.org/updikeprize`
Sep 25 – Oct 1	10th International ConTEXt Meeting, "Piece of Cake", Kalenberg, The Netherlands. `meeting.contextgarden.net/2016`
Sep 30 – Oct 2	Oak Knoll Fest XVIX, New Castle, Delaware. `www.oakknoll.com/fest`

Status as of 31 March 2016

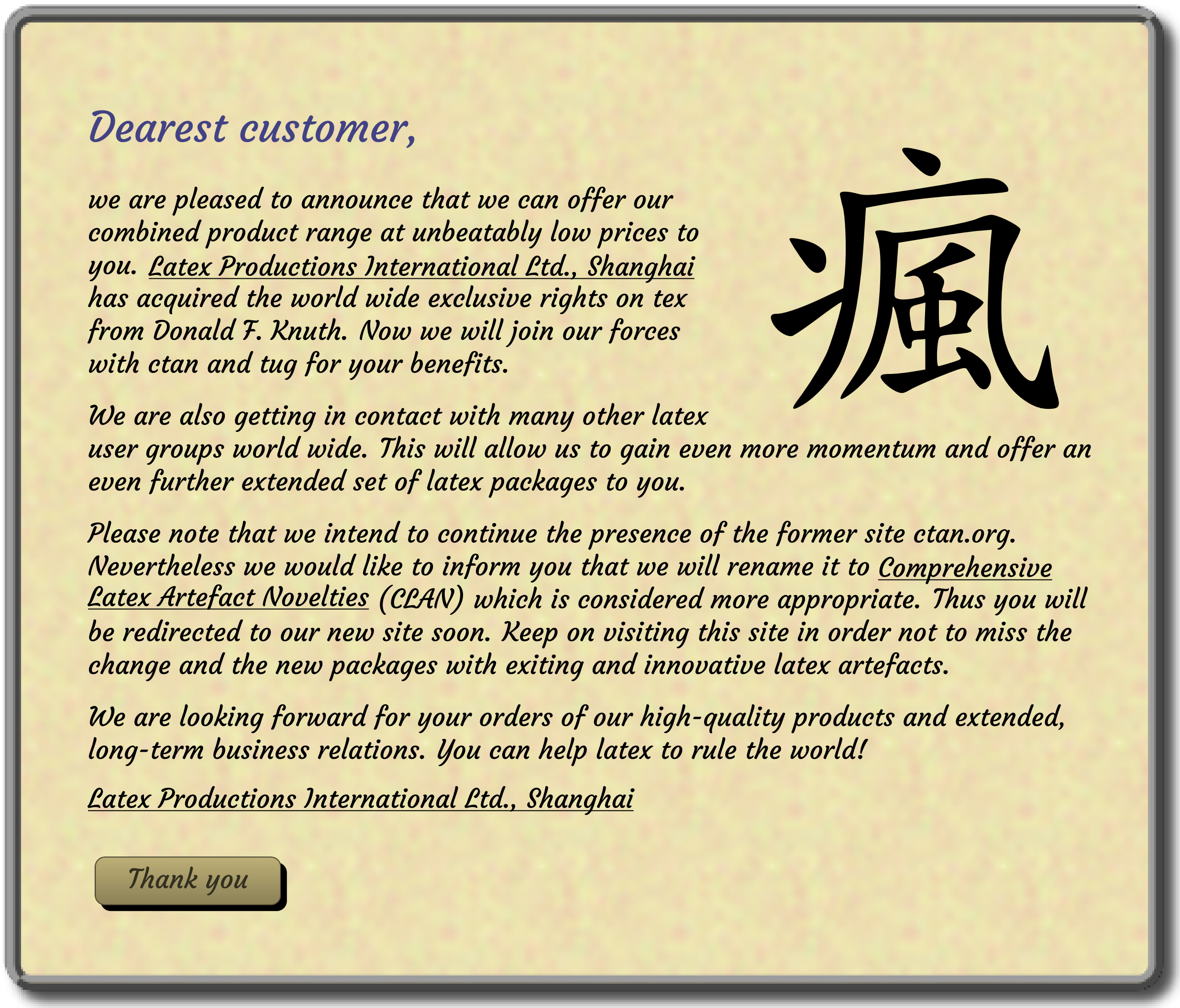
For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568. e-mail: `office@tug.org`). For events sponsored by other organizations, please use the contact address provided.

User group meeting announcements are posted at `lists.tug.org/tex-meetings`. Interested users can subscribe and/or post to the list, and are encouraged to do so.

Other calendars of typographic interest are linked from `tug.org/calendar.html`.

CTAN

(1 April 2016)



Dearest customer,

we are pleased to announce that we can offer our combined product range at unbeatably low prices to you. Latex Productions International Ltd., Shanghai has acquired the world wide exclusive rights on tex from Donald F. Knuth. Now we will join our forces with ctan and tug for your benefits.

瘋

We are also getting in contact with many other latex user groups world wide. This will allow us to gain even more momentum and offer an even further extended set of latex packages to you.

Please note that we intend to continue the presence of the former site ctan.org. Nevertheless we would like to inform you that we will rename it to Comprehensive Latex Artefact Novelties (CLAN) which is considered more appropriate. Thus you will be redirected to our new site soon. Keep on visiting this site in order not to miss the change and the new packages with exiting and innovative latex artefacts.

We are looking forward for your orders of our high-quality products and extended, long-term business relations. You can help latex to rule the world!

Latex Productions International Ltd., Shanghai

Thank you

Thanks to Gerd Neugebauer.

Reports and notices

Reports and notices