TUGboat

Volume 39, Number 2 / 2018
TUG 2018 Conference Proceedings

TUG 2018 98 Conference sponsors, participants, program
100 Joseph Wright / *TUG goes to Rio*
104 Joseph Wright / *TEX Users Group 2018 Annual Meeting notes*

Graphics 105 Susanne Raab / *The* `tikzducks` *package*

LATEX 107 Frank Mittelbach / *A rollback concept for packages and classes*
113 Will Robertson / *Font loading in LATEX using the* `fontspec` *package: Recent updates*
117 Joseph Wright / *Supporting color and graphics in* `expl3`
119 Joseph Wright / `siunitx`*: Past, present and future*

Software & Tools 122 Paulo Cereda / *Arara — TEX automation made easy*
126 Will Robertson / *The Canvas learning management system and LATEXML*
131 Ross Moore / *Implementing PDF standards for mathematical publishing*

Fonts 136 Jaeyoung Choi, Ammar Ul Hassan, Geunho Jeong / *FreeType_MF_Module: A module for using METAFONT directly inside the FreeType rasterizer*

Methods 143 S.K. Venkatesan / *WeTEX and Hegelian contradictions in classical mathematics*

Abstracts 147 TUG 2018 abstracts (Behrendt, Coriasco et al., Heinze, Hejda, Loretan, Mittelbach, Moore, Ochs, Veytsman, Wright)

---

149 *MAPS*: Contents of issue 48 (2018)
150 *Die TEXnische Komödie*: Contents of issues 2–3/2018
151 *Eutypon*: Contents of issue 38–39 (October 2017)

General Delivery 151 Bart Childs and Rick Furuta / *Don Knuth awarded Trotter Prize*
152 Barbara Beeton / *Hyphenation exception log*

Book Reviews 153 Boris Veytsman / *Book review: W. A. Dwiggins: A Life in Design by Bruce Kennett*

Hints & Tricks 155 Karl Berry / *The treasure chest*

Cartoon 156 John Atkinson / *Hyphe-nation; Clumsy*

Advertisements 157 TEX consulting and production services

TUG Business 158 TUG institutional members
159 TUG 2019 election

News 160 Calendar

TEX Users Group

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

Individual memberships

2018 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. Members also have the option to receive *TUGboat* and other benefits electronically, at a discount.

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 carries with it such rights and responsibilities as voting in TUG elections. All the details are on the TUG web site.

Journal subscriptions

TUGboat subscriptions (non-voting) are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate for 2018 is $110.

Institutional memberships

Institutional membership is primarily a means of showing continuing interest in and support for TEX and TUG. It also provides a discounted membership rate, site-wide electronic access, and other benefits. For further information, see 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.

[printing date: September 2018]

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

tug.org
tug.org/TUGboat

Electronic Mail

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

2018 Conference Proceedings

TEX Users Group
Thirty-eighth annual TUG meeting
Rio de Janeiro, Brazil
July 20–22, 2018

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
TUGboat Editor Barbara Beeton
Proceedings Editor Karl Berry

Volume 39, Number 2, 2018
Portland, Oregon, U.S.A.

TUG 2018 — Rio de Janeiro, Brazil

https://tug.org/tug2018 ▪ tug2018@tug.org

Room 232
IMPA — Instituto de Matemática Pura e Aplicada
Estr. Dona Castorina, 110
Jardim Botânico
Rio de Janeiro, RJ 22460-320

A SATELLITE OF

TUG 2018 was an official satellite conference of ICM 2018.

Sponsors

TEX Users Group ▪ DANTE e.V. ▪ ICM'18
with special assistance from individual contributors. *Thanks to all!*

Special guests from the Lua team

Waldemar Celes ▪ Luiz Henrique de Figueiredo ▪ Roberto Ierusalimschy

Conference committee

Karl Berry ▪ Paulo Cereda ▪ Robin Laakso ▪ Paulo Ney de Souza ▪ Boris Veytsman

Participants

Francisco J Alves, Rio de Janeiro
Maynara Azevedo Aredes, Federal University of Rio de Janeiro
Felipe da Silva Barreto, Rio de Janeiro
Nelson Beebe, University of Utah
Doris Behrendt, DANTE e.V.
Waldemar Celes, PUC-Rio & Lua team
Paulo Cereda, University of São Paulo
Jaeyoung Choi, Soongsil University
Marcel de Sena Dall'Agnol, Rio de Janeiro
Luiz Henrique de Figueiredo, IMPA & Lua team
Gert Fischer, Mönchengladbach, Germany
Ulrike Fischer, Mönchengladbach, Germany
Mylena da Silva Gomes, Rio de Janeiro
Joachim Heinze, Senior Advisor Mathematics Sciences, Heidelberg
Tom Hejda, Charles University Prague
Roberto Ierusalimschy, PUC-Rio & Lua team
Eduardo Kalinowski, Brasilia
Mico Loretan, Zurich, Switzerland
Jonas Malaco, Rio de Janeiro
Frank Mittelbach, LaTeX Project
Ross Moore, Macquarie University
Eduardo Ochs, UFF
Matheus Rocha de Souza Ramos, Ufes
Ana Claudia Ribeiro, E-papers Serviços Editoriais
Will Robertson, University of Adelaide
Chris Rowley, PCEC and LaTeX
Volker RW Schaa, DANTE e.V.
Paulo Ney de Souza, University of California, Berkeley & BooksInBytes
Arthur Szasz, Protocubo
S.K. Venkatesan, TNQ
Boris Veytsman, George Mason University & Chan Zuckerberg Initiative
Joseph Wright, LaTeX Project

TUG 2018 program

Friday	8:00 am	*registration*	
July 20	8:55 am	Paulo Ney de Souza, UC Berkeley & BooksInBytes	*Opening*
	9:00 am	Roberto Ierusalimschy, PUC-Rio & Lua team	*The making of Lua*
	9:45 am	Eduardo Ochs, UFF	*Dednat6: An extensible (semi-)preprocessor for LuaLaTeX that understands diagrams in ASCII art*
	10:20 am	*break*	
	10:45 am	Mico Loretan, Zurich, Switzerland	*Selective ligature suppression with the* `selnolig` *package*
	11:15 am	Joseph Wright, LaTeX Project	*Fly me to the moon: (LA)TeX testing (and more) using Lua*
	11:50 am	Paulo Cereda, University of São Paulo	*From parrot 1.0 to phoenix 4.0: 6 years of* `arara`*, the beginning of a new era*
	12:25 pm	*lunch*	
	1:45 pm	Will Robertson, University of Adelaide	*Creating teaching material with LaTeXML for the Canvas Learning Management System*
	2:25 pm	Ross Moore, Macquarie University	*Authoring accessible 'Tagged PDF' documents using LaTeX*
	3:00 pm	*break*	
	3:20 pm	S. Coriascoet al., Università di Torino	*An automated method based on LaTeX for the realization of accessible PDF documents containing formulae*
	3:55 pm	Doris Behrendt, DANTE e.V.	*The General Data Protection Regulation (GDPR) in the European Union*
	4:30 pm	*TUG Annual General Meeting*	
	4:30 pm	*Workshop: Accessibility challenges in LaTeX*	
Saturday	8:55 am	*announcements*	
July 21	9:00 am	Frank Mittelbach, LaTeX Project	*A quarter century of* `doc`
	9:35 am	Joseph Wright	*Through the looking glass, and what Joseph found there*
	10:10 am	*break*	
	10:30 am	Boris Veytsman, George Mason Univ. & Chan Zuckerberg Initiative	*Stubborn leaders six years later*
	11:05 am	Joseph Wright	`siunitx`*: Past, present and future*
	11:40 am	Frank Mittelbach	*Compatibility in the world of LaTeX*
	12:40 pm	*lunch*	
	2:00 pm	Paulo Ney de Souza	*Minimizing LaTeX files — First steps to automated journal processing*
	2:35 pm	Tom Hejda, Charles University Prague	`yoin` *— Yet another package for automation of journal typesetting*
	3:10 pm	*break*	
	3:30 pm	Joachim Heinze	*The unchanged changing world of mathematical publishing*
	4:05 pm	Boris Veytsman	*R+knitr workshop (*`tug.org/tug2018/workshops.html`*)*
Sunday	8:55 am	*announcements*	
July 22	9:00 am	S.K. Venkatesan and TNQ Lab	*WeTeX (WYSIWYG and LaTeX) and Hegelian contradictions in classical mathematics*
	9:35 am	Susanne Raab, Paulo Cereda*	*A short introduction to the TikZducks package*
	10:10 am	*break*	
	10:30 am	Jaeyoung Choi, Soongsil University	*FreeType_MF_Module: A module for using METAFONT directly inside FreeType's rasterizer*
	11:05 pm	Will Robertson	*Unicode fonts with* `fontspec` *and* `unicode-math`
	11:40 am	*lunch*	
	1:00 pm	*bus to Sugarloaf*	
	≈ 5:30 pm	*return to hotel*	

* = presenter

TUG goes to Rio

Joseph Wright

TUG 2018 took place in Rio de Janeiro, Brazil, at the *Instituto de Matemática Pura e Aplicada* (IMPA). Most of the foreign attendees had chosen to stay at the 'official' hotel, which meant that as well as the formal business, there was plenty of time to talk over breakfast and in the evenings. That was evident in reception the evening before the meeting, and even more so at breakfast on the first morning. This was a good chance to catch up with old friends.

1 Day one

After a (brief) introduction from the conference chair, Paulo Ney de Souza, the floor was handed to Roberto Ierusalimschy to start us with a bang: an overview of Lua development. He gave us an insight into how Lua grew from early beginnings, and how it got picked up by games developers: a big part of Lua's importance. He then gave us an insight into the two key aspects of Lua's success: the ability to embed and extend the language. That led to Lua being embedded in a range of applications, not only games but also devices as varied as cars and routers. We had a lively question session, ranging from Unicode support to what might have been done differently if the Lua team didn't have any users to worry about!

We then moved on to Eduardo Ochs, talking about using Lua as a pre-parser to convert 'ASCII art' into complex mathematical diagrams. He explained the history: the origin of ASCII art as comments to help understand sometimes complex TEX code! After a summary of the original pre-processor, he showed how using Lua(TEX), the processing can be done in-line in the file with no true pre-processing step. He showed how this can be set up in an extensible and powerful way, using Lua to do the 'heavy lifting'.

After the coffee break, we reconvened for three talks. Mico Loretan started, describing his package selnolig [4]. He started by showing us examples of 'unfortunate' ligatures in English words, and how they can appear when suppressed by babel and by selnolig. He then focussed in on some details: what a ligature is, why they are needed and how different fonts provide them. He moved on to describe why you need to suppress ligatures, in particular where they cross *morpheme* boundaries. Mico then gave us a very useful summary of how the linguists work here and how they need to link to typography. After showing us the issues with other approaches, he moved on to how selnolig uses LuaTEX callbacks to influence ligatures 'late' in processing. His rule-based interface means that ligatures can be suppressed for whole classes of words with only a small number of discrete settings. An interesting aspect of this work is how variable the ligature support is in OpenType fonts, in particular what constitutes a 'common' ligature.

I spoke next, focussing on l3build [11]. I gave a brief overview of LATEX testing, from the earliest days of the team to the current day. I covered why we've picked Lua for our current testing set-up, what works and what (currently) doesn't. Looking forward to other talks, the need for PDF-based testing came up in discussions of tagging, and is very much on the l3build 'to do' list.

Paulo Cereda then talked about his build tool, arara [1]. He started with an overview of other tools, before explaining how arara is different: it is at heart a 'no-guesswork' approach. He showed us the core, simple, syntax, before moving on to a time-line of releases to date. He summed up the new features in version 4.0, before moving to a series of live demonstrations. These started with simple ideas and moved on to new, complex ideas such as conditionals and taking user input. He then finished by looking to the future, both of arara and of *araras* (macaws).

Lunch was arranged in the IMPA café, giving us all a chance to absorb the morning's information and take on all-important sustenance. The café is right by the forest, so we also took the opportunity to take in the local wildlife.

We started back after lunch with a couple of slides from Barbara Beeton, absent from the meeting, presented by TUG President Boris Veytsman. I think everyone was pleased to hear from Barbara, even if at one remove.

Will Robertson then took the podium. He started with some thoughts on questions he gets as an Australian (no TEX involved). His koala pictures were particularly fun. His talk proper was on his work with the Learning Management System (LMS) used by his employer. This system (Canvas) has a programmable API for controlling information made available to students. He laid out the issues with the documentation he had: a very large, unmaintainable word processing document. Will talked about various tools for creating HTML from LATEX, the workflow he has chosen, and then showed more detail on the system he is using, LATEXML. He then expanded on how using LATEXML plus scripting, he can populate the LMS in a (semi)automated way, making his work more efficient.

The second speaker in the 'Australian panel' session was Ross Moore. Ross started with a demo of why tagging PDFs is needed: making the information accessible not just to people but widely to the computer, to allow re-use in alternative views. He

expanded on the drivers for this, in particular legal requirements for accessible documents.

Our next talk came in remotely from Sandro Coriasco, also dealing with aspects of tagged PDFs for accessibility. He started by outlining the team involved in this work, focussed on making material accessible to the blind. The aim of their work has been targeted at mathematical formula, generating 'actual text' which can then be used by screen readers or similar. He finished with a 'live demo', and left us with lots to think about.

We then had a non-TEX talk: Doris Behrendt on GDPR. She started by looking at the EU Official Journal on the GDPR, and we had an excursion into the font used for typesetting (Albertina). She then gave details of the regulations, along with a number of extremely amusing examples of how people have approached them.

Presentations over, Boris Veytsman took up the baton again as TUG President, and led a lively TUG AGM discussion: details in a separate item.

The business of the day ended with a discussion (workshop) session looking at technical aspects of tagging PDFs. As one might expect, this large topic could have filled the entire day, and it was clear that the hour we had available was very much laying out a framework for future meetings.

Those of us staying at the official hotel took the minibus back to Ipanema, heading out in the evening to what became the unofficial 'team bar' for the meeting. We split into small groups, and talking about TEX or otherwise went on for quite some time!

2 Day two

Frank Mittelbach started the day's proceedings, talking about his doc [5] package for literate programming. He explained the background, what works and more importantly what didn't. The success of doc as a standard makes change challenging, but at the same time there is a need for updates. He then laid out goals for a new version: backward-compatibility, new markup and out-of-the-box hyperref support. He showed us the features for creating new markup. There are some wrinkles, for example that hyperref support still has to be manually activated. Frank wrapped up by pointing to the testing version, and gave us a likely release date (for TL'19).

I then gave my first talk of the day, looking at expl3 [10] concepts related to colour and graphics. I outlined the LATEX 2_ε background, what is happening with the LATEX 2_ε drivers and then moved on to my expl3 experiments. First I talked about colo(u)r, and the idea of colour expressions as introduced by xcolor [3]. These are trivial to work out in expl3 due to the expandable FPU we have there. I then looked at creating graphics, particularly how I've been inspired by pgf/Ti*k*Z [9]. I showed how I've used the fact that pgf has a clear structure, and mapped that to expl3 concepts. I showed some examples of the existing drawing setup, and where I'll be going next.

We returned after coffee for a short talk from Boris Veytsman on tackling an apparently simple issue: putting leaders level with the first line of a long title! He showed that this is non-trivial, and how as a contractor he has to explain this to clients. He then showed how he solved the issue, leading to a lively discussion about other possible approaches.

I then came back for my second talk of the day, this time about siunitx [13]. I started by explaining the history of the package, starting with the initial `comp.text.tex` post that led to its creation. I outlined the core features, present from version 1, and why I've now twice re-written it. I finished by promising a first alpha version of version 3.

Frank then returned for a morning of symmetry, talking about compatibility requirements. He talked about the historical situation, starting from Knuth's introduction of TEX and taking us through the development of LATEX, PDF support and Unicode engines. He then moved on to look at the LATEX 2_ε approach to compatibility, starting with the 1994 approach, fixltx2e. He explained how that was intended to work, and why it didn't. The new approach, latexrelease [12], tackles the same problems but starts with the idea that it applies to both the kernel and to packages. Frank covered the idea of roll-back for packages, and how this works at the user and developer levels. Frank finished off with some thoughts about the future, and the fact that most new users probably pick up these ideas without issue.

Our conference chair, Paulo Ney de Souza, took the first slot after lunch to speak on how he's approached a major challenge, managing the abstracts for the upcoming ICM 2018 meeting. His talk ranged over topics such as citation formatting, small device output, production workflows and dealing with author preambles. He covered the wide range of tools his team has assembled to automate PDF creation from a heterogeneous set of sources. His wide-ranging talk was a tour de force in automated publication.

After a brief break, we moved to Tom Hejda (who TeX.sx users know as yo'), on his tool yoin [2]. He explained that his current workflow for producing journal issues is at present a mix of tools, and this is likely not long-term sustainable. He then moved to showing how yoin can be used to compile both the master file for an issue and, as required, each article within it.

Figure 1: Knuthduck

The last talk of the day was from Joachim Heinze, formerly of Springer. He talked about journal publishing, and how online accessibility of publications has changed the landscape for publishers. He gave an entertaining look into this world, posing the question 'Where is the information we have lost in data?'

With the formal business done, some of the group remained at IMPA for a workshop on R and Knitr, led by Boris Veytsman. I decided to skip that, and to conserve energy for the real business of the meeting: the conference meal! We all met up again for that event at Rubaiyat Rio. This was a chance to unwind, compare notes and of course to present the all-important Duane Bibby original: this year it was given to the Lua team.

3 Day three

The final day of TUG2018 followed the conference banquet, which of course meant that there were a few tired (or missing!) delegates. Luckily, the talks kept us all awake.

The first talk of the day came from S. K. Venkatesan, focussing on his WaTEX tool, and the link to countability of computing problems. He ranged over several fundamental questions in computability.

We then moved to Paulo Cereda (on behalf of Susanne Raab), looking at the Ti*k*Zducks package [8]. He started by pointing out that whilst drawing ducks is fun, there is serious coding behind it. He showed us a range of examples of how key–value settings allow a wide range of (wacky) customisation of duck drawings. A particular highlight was rendering Don Knuth as a Ti*k*Zduck (Figure 1).

Once we'd all refuelled, Jaeyoung Choi took the podium to describe work on using Metafont directly inside FreeType. He laid out the advantages of Metafont, and the problems for use by font designers. He then moved to look at the particular challenges faced in developing CJK fonts: the very large number of characters, and resulting significant time/cost investment required. With modern computing power, this can be solved using Metafont to parametrise this large number of glyphs. Jaeyoung demonstrated a GUI which allows control of the appearance of characters in an (almost) interactive way. He then moved on to look at how to integrate Metafont directly into the TrueType rasteriser.

The final talk came from Will Robertson, on fontspec [7] and unicode-math [6]. He started by showing us some issues in the fonts in books for children, before reviewing unicode-math. He showed how it handles complex maths, allowing re-use of copied material and changing the style of output. He then looked at the development approach he's taken in 'cleaning up' unicode-math and fontspec. He covered various aspects of the expl3/l3build/Git(Hub) workflow he's now perfected. He then moved on to fontspec, talking about the background, current interfaces and possible future developments. It was a great final talk: wide-ranging, thought-provoking and fun.

With the formal business done, we headed to the roof of IMPA for the traditional conference photograph. After a lunch break, it was off for most of us to the excursion to Sugarloaf Mountain, and the end of the meeting proper. We of course managed to pick the one afternoon of the week where the top of the mountain was hidden in cloud, but the trip up was great fun. Almost all of the foreign delegates were staying for Sunday evening, and several of us availed ourselves once again of the 'team bar' for a well-deserved evening of informal chat.

References

[1] P. Cereda, M. Daniel, et al. arara: The cool TEX automation tool, 2018. ctan.org/pkg/arara

[2] T. Hejda. yoin, 2018. https://github.com/tohecz/yoin

[3] U. Kern. Extending LATEX's color facilities: The xcolor package, 2016. ctan.org/pkg/xcolor

[4] M. Loretan. The selnolig package: Selective suppression of typographic ligatures, 2018. ctan.org/pkg/selnolig

[5] F. Mittelbach. The doc and shortvrb packages, 2018. ctan.org/pkg/doc,shortvrb

[6] W. Robertson. Experimental Unicode mathematical typesetting: The unicode-math package, 2018. ctan.org/pkg/unicode-math

[7] W. Robertson. The fontspec package: Font selection for XHLATEX and LuaLATEX, 2018. ctan.org/pkg/fontspec

[8] samcarter. The Ti*k*Zducks package: Using ducks in Ti*k*Z, 2018. ctan.org/pkg/tikzducks

[9] T. Tantau and C. Feuersänger. Ti*k*Z and pgf, 2015. ctan.org/pkg/pgf

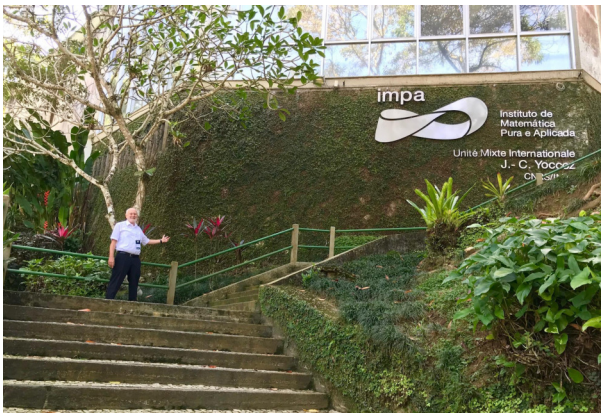
[10] The LATEX Project. The expl3 package and LATEX3 programming, 2018. ctan.org/pkg/l3kernel

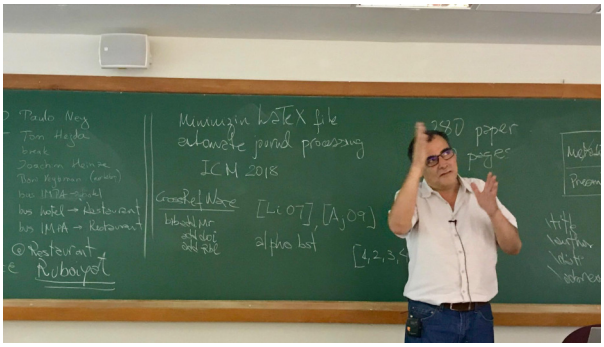
[11] The LATEX Project. The l3build package: Checking and building packages, 2018. ctan.org/pkg/l3build

[12] The LaTeX Project. The latexrelease package, 2018. ctan.org/pkg/latexrelease

[13] J. Wright. siunitx: A comprehensive (SI) units package, 2018. ctan.org/pkg/siunitx

◇ Joseph Wright
Morning Star
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2.co.uk





(1) ? (2) E. Ochs (3) A. Ribeiro (4) D. Behrendt
(5) B. Veytsman (6) V.R.W. Schaa (7) W. Robertson
(8) U. Fischer (9) S.K. Venkatesan (10) M. Beebe
(11) P. Cereda (12) N. Beebe (13) P. Ney de Souza
(14) Mrs. Heinze (15) ? (16) C. Mittelbach (17) R. Ierusalimschy
(18) J. Choi (19) E. Kalinowski (20) C. Rowley (21) R. Moore
(22) M. Rocha de Souza Ramos (23) L. de Figueiredo
(24) J. Malaco (25) F. Mittelbach (26) J. Heinze
(27) M. Loretan (28) A. Szasz (29) J. Wright (30) T. Hejda
(∗) Der Bär

Photos in first column © Joachim Heinze; in second column, © Volker RW Schaa. Thanks to both.

TUG 2018 Annual General Meeting notes

Notes recorded by Joseph Wright

The TUG annual meeting took place in conjunction with the TUG'18 conference in Rio de Janeiro on 22 July 2018. The meeting was conducted by the TUG president, Boris Veytsman.

A letter from Jonathan Fine was read verbatim, per his request, by Boris, and the two questions it contained were answered as follows:

Q: First, please report on any Board actions and discussions relating to the TUG website. And what intentions does the Board have regarding the TUG website?
A: There is general agreement on making `tug.org` more mobile-friendly, and some discussion has been directed to that end.

Q: Second, is there any potential conflict between the personal interests of any Board member and the interests of TUG? And if so, what action has the Board taken to protect both sides from this potential conflict of interest?
A: There are no conflicts that we are aware of.

Question regarding *TUGboat* open access

The question was posed: should *TUGboat* be open access? Boris laid out the idea and background.

Joseph Wright said that (a) articles are often available anyway from authors, and (b) people would still join: members join for other reasons.

Tom Hejda also agreed that people would still join, and good articles could be linked on, e.g., Facebook which would then link back to TUG.

Mico Loretan agreed with the previous points, and in particular the marketing possibilities that this would make possible.

Will Robertson pointed out that open access makes our lives easier, however, the counter argument is that there is a push for joining from *wanting* articles that currently cannot be accessed unless one is a member.

Frank Mittelbach suggested that there are few such people today. He suggested that *TUGboat* is partly a research journal in document engineering: it is referenced by other researchers in this field. It also carries a number of articles focused on engine and macro development, and these may be important for other developers in the TEX community. The current embargo may slow down work in both areas.

Paulo Ney de Souza pointed out that a print journal may be picked up by (e.g.) libraries.

Frank felt that most people who join electronic-only are most likely mainly doing so for "supporting TUG". Mico takes the electronic-only option and agreed that he looks at the membership fee in just this way.

Membership drive

Boris outlined the issue: membership has been falling (though not this year). There are lots of TEX *users*: see the scale of, e.g., Overleaf.

Boris posed the question of how to convert those *users* to TUG members. Frank suggested, "Maybe you can't". TUG arguably missed the opportunity for development of "cloud" services offered, which was picked up by Overleaf (and others). The cloud services, and so ultimately the users, rely on TUG for the "back end" part of their business.

Boris asked the question: Our work is important, and the TEX community needs us, but do other people see this?

The nature of TUG

Tom asked, are we now the "TEX Developers Group"?

Frank: "Yes, to a large extent". Others agreed with this as reflecting the current reality.

Paulo Cereda: The nature of TUG meetings has changed: a few "power" users attend, mainly for developer discussion.

Frank: Why does one join? To support the "mission" as a user. Perhaps *TUGboat* is an exception in print (per library comment from Paulo) but "supporting the mission" is what drives the size of the group; this is not linked to the size of the user base.

Joseph: This seems to be the case for UK-TUG too.

Ross Moore: Some form of shareware approach might highlight the need for developer support.

Tom: SageMath has gone this route; they could not operate without income. Similarly for MatLab. Other examples were given. Open question: What were their original license/copyright situations?

Boris: The legal/moral right could be problematic, as TUG doesn't own the tools.

Suggestion: Post a "Support us" link on downloading, similar to Ubuntu.

Paulo C.: Current members tend to expect others to join for the same reasons ("support"), i.e., the need to provide help particularly to new users, e.g., through workshops.

Tom: There are two types of members, people and institutions. Among the latter, there are 10 universities, one publisher, four commercial entities. Is this a possibility to raise money?

Boris has explored this, with very little response. Persuasion works best as an employee.

Other business: None.

The tikzducks Package

Susanne Raab

Abstract

Ti*k*Zducks is a funny little package to draw rubber ducks in Ti*k*Z. The following article will give a short overview of the package and show some examples of how the Ti*k*Zducks package can be used.

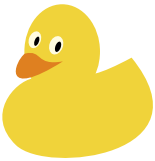
1 Introduc(k)tion

The host country of TUG'18, Brazil, is also the home country of *Hevea brasiliensis*, commonly known as rubber tree. This tree is the source of natural latex, which (or some of its synthetic replacements) can be used to produce rubber ducks.

This raises the question if rubber ducks can also be made of LaTeX? Yes, they can — using the Ti*k*Zducks package! This package originates from a post on TeX.StackExchange [2] which showed that the TeX community a) has a great sense of humour and b) seemed to be in desperate need of a package to easily add ducks to their documents. The package is now available on CTAN (`ctan.org/pkg/tikzducks`) and included in both TeX Live and MiKTeX.

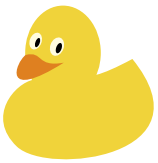
The basic usage is fairly simple:

```
\documentclass{standalone}
\usepackage{tikzducks}
\begin{document}
\begin{tikzpicture}
  \duck
\end{tikzpicture}
\end{document}
```



Instead of loading it as a package, there is also a Ti*k*Z library with the same functionality:

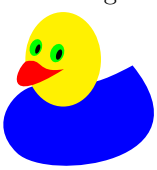
```
\documentclass[tikz]{standalone}
\usetikzlibrary{ducks}
\begin{document}
\begin{tikzpicture}
  \draw (0,0) pic {duck};
\end{tikzpicture}
\end{document}
```



2 Modularity

Drawing a simple yellow duck is by far not the only thing which this package is capable of. It comes with many accessories and the ability to adjust colours to customise the ducks. Both can be controlled with an intuitive key-value interface. For example, the colour of the various body parts can be changed:

```
\duck[body=blue,head=yellow,
      bill=red,eye=green]
```



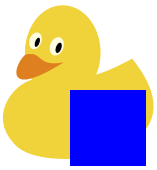
Or accessories can be added:

```
\duck[cap=blue,cricket]
```

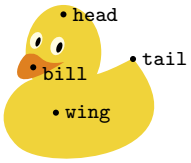
As can be seen in the above example, the colour of additional components can usually be changed in the same way as for the body parts. A complete list of all the available options would exceed the scope of this *TUGboat* article; please see the package documentation [3, Section 2].

The customisation of the Ti*k*Zducks is not limited to the predefined accessories included in the package. In the end, the ducks are just shapes in a `tikzpicture`. This means that the essentially infinite amount of possibilities of Ti*k*Z are available for further customisation. A trivial example:

```
\duck
\fill (2,0) rectangle (1,1);
```



To ease customisation, the Ti*k*Zducks package predefines some coordinates at prominent positions:



3 Football ducks

TUG'18 is taking place in the football nation Brazil, so it is only appropriate that the Ti*k*Zducks can not just play football (`\duck[football]`), but also have customisable jerseys. The special macro `\stripes` adds a highly adjustable stripe pattern which allows emulating jerseys from nearly every team. The following example shows how to reproduce the jerseys of the Brazil national team:

```
\duck[tshirt=Green,jacket=Gold,
  football,stripes={
  \stripes[color=Blue,
  rotate=-75,width=0.5,
  distance=1.5]}]
```

4 Duckify a person

A nice application of the Ti*k*Zducks is to "duckify" persons. In the following a short tutorial will give an idea how this can be done based on a photograph.

As an example, let's use this photo of Don Knuth (image by J. Appelbaum [1]):

As a first step, suitable colours have to be determined. For all non-artists, this process can be simplified with a colour picker tool, which is included in many image manipulation programs (e.g., `gimp`) or as standalone applications. The following colours are extracted from the example image:

sweater: RGB = (95, 42, 50)
shirt: RGB = (241, 238, 254)
skin: RGB = (229, 175, 166)

If colours derived from a photo are used to fill large shapes of solid colour, they are sometimes perceived as too dark. This problem can be solved by mixing them with white to make them a bit lighter. With slightly lightened colours and some suitable accessories the following "duckified" version of Don Knuth can be drawn:

```
\definecolor{jacket}{RGB}{95,42,50}
\definecolor{skin}{RGB}{229,175,166}
\definecolor{shirt}{RGB}{241,238,254}
\duck[%
  body=skin!50!white,
  bill=skin!80!gray,
  tshirt=shirt, jacket=jacket,
  recedinghair=lightgray!80!brown,
  squareglasses=brown!30!lightgray,
  eyebrow, book=\TeX,
  bookcolour=black!40!brown
]
```

5 Examples

As mentioned above, see the package documentation for all of Ti*k*Zducks's myriad possibilities, but as a teaser, here are a few examples:

From top left to bottom right: Frenchman duck, crowned duck, duck scout, bee duck, Royal Swan Upper, Albert Einstein, witch duck and David Hilbert.

To see more examples, there are some collections available online

- `github.com/samcarter8/tikzducks/tree/master/duckpond`
- `tex.stackexchange.com/questions/387047`

to which users can also contribute their creations. There are even several videos produced by Ulrike and Gert Fischer, Carla Maggi and Paulo Cereda, featuring the Ti*k*Zducks:

- The great Ti*k*Zducks Christmas Extravaganza, `vimeo.com/246256860`
- International Pizza Day, `vimeo.com/254643482`
- Happy Groundhog Day, `vimeo.com/252719006`
- Aquarela with Ti*k*Zducks, `vimeo.com/270727100`

6 Suggestions or problems

The package source is hosted at `https://github.com/samcarter8/tikzducks`, along with a tracker for bug reports, feature requests or contributions. For questions related to the package there is a special {`tikzducks`} tag available at TEX.StackExchange.

7 Acknowledgments

I would like to thank the TEX Users Group for the possibility to present the Ti*k*Zducks package at their annual meeting and especially Paulo Cereda for the vivid presentation he gave in Rio de Janeiro!

The author of the Ti*k*Zducks package is grateful for the valuable contributions and suggestions received from Ulrike Fischer, Carla Maggi, Enrico Gregorio, Andrew Stacey and many others.

References

[1] J. Appelbaum. Licensed under CC BY-SA. `https://en.wikipedia.org/wiki/File:KnuthAtOpenContentAlliance.jpg`

[2] How can we draw a duck?, 2017. `tex.stackexchange.com/q/346695`

[3] samcarter. *The Ti*k*Zducks package, Version 0.7*, 2018. `ctan.org/pkg/tikzducks`

◇ Susanne Raab
`Susanne.Raab (at) fau (dot) de`

A rollback concept for packages and classes

Frank Mittelbach

Abstract

In 2015 a rollback concept for the LaTeX kernel was introduced. Providing this feature allowed us to make corrections to the software (which more or less didn't happen for nearly two decades) while continuing to maintain backward compatibility to the highest degree.

In this paper we explain how we have now extended this concept to the world of packages and classes, which was not covered initially. As classes and extension packages have different requirements compared to the kernel, the approach is different (and simplified). This should make it easy for package developers to apply it to their packages and authors to use when necessary.

Contents

1 Introduction 107

2 Typical scenarios 108

3 The document interface 108
- 3.1 Global rollback 108
- 3.2 Individual rollback 108
- 3.3 Specifying a version instead of a date 109
- 3.4 Erroneous input 109
- 3.5 Advice for early adopters 109

4 The package/class interface 110

5 Special considerations for developers 110
- 5.1 Early adopters 111
- 5.2 New major release in beta 111
- 5.3 Two major releases in use 111
- 5.4 Fine grained control (if needed) . . . 112
- 5.5 Using `l3build` for source management 112

6 Command summary 112
- 6.1 Document interface, for users 112
- 6.2 Package and class interface, for developers 112

1 Introduction

In 2015 we introduced a rollback concept for the LaTeX kernel that enables a user to request a kernel rollback to its state at a given date by using the latexrelease package [1]. For example,

```
\RequirePackage[2016-01-01]{latexrelease}
```

would result in undoing all kernel modifications (corrections or extensions) released between the first of January 2016 and the current date.[1] Undoing means reinstalling the definitions current at the requested date and normally also removing new commands from TeX's memory so that `\newcommand` and similar declarations do not fall over because a name is already declared.

This mechanism helps in correctly processing older documents that contain workarounds for issues with an older kernel, issues that have since been fixed in a way that would make the old document fail, or produce different output, when processed with the newer, fixed kernel.

If necessary, the latexrelease package also allows for rolling the kernel forward without installing a new format. For example, if the current installation is dated 2016-04-01 but you have a document that requires a kernel with date 2018-01-01, then this can be achieved by starting it with

```
\RequirePackage[2018-01-01]{latexrelease}
```

provided you have a version of the latexrelease package that knows about the kernel changes between the date of your kernel and the requested date. Getting this version of the package is simple as the latest version can always be downloaded from CTAN. Thus you will be able to process your document correctly even when updating your complete installation is not advisable or impossible for one or another reason.

However, rolling back the kernel state is only doing half of the job: the LaTeX universe consists of many add-on packages and those were not affected by a kernel rollback request. We are therefore now extending the concept by providing a much simpler method for use in packages and classes, one that we think will be straightforward for developers and also easy for document authors to use.

Unlike the method used by the kernel, which tracks every change individually and is able to roll back the code to precisely the state it had on any given day, the new method for packages and classes is intended to cover only major change points, e.g., the introduction of major new features or (incompatible) changes in syntax or interfaces.

As we will have only a few rollback points per package or class, the different releases are all stored in separate files. In the main file it therefore only needs a single declaration per release to enable rollback. The downside is, of course, that for each release the whole package code is stored, instead of managing the differences between releases. This is one of the

[1] There are a few exceptions as some modifications are kept: for example, the ability to accept date strings in ISO format (e.g., 2016-01-01) in addition to the older LaTeX convention (e.g., 2016/01/01). These are not rolled back because removing such a feature would result in unnecessary failures.

reasons why this approach should be used only for major changes, i.e., at most a handful in the lifetime of a package.

From a technical perspective it is also possible to use the method introduced with latexrelease in package and class files, i.e., to mark up modifications using the commands `\IncludeInRelease` and `\EndIncludeInRelease` — the package's documentation [1] gives some advice on how to apply it in a package scenario — but the use of these commands in package code is cumbersome and results in fairly unreadable code, especially when there are many minor changes. This is an acceptable price to pay for fairly stable code, such as the kernel itself, since it offers complete control over the rollback to any date, but it is not truly practical in package or class development and so, to our knowledge, it has therefore never been used up to now. Section 5.4 gives some advice on how to achieve fine-grain control in a somewhat simpler manner.

2 Typical scenarios

A typical example, for which such a rollback functionality would have provided a major benefit (and will do so for packages in the future), is the caption package by Axel Sommerfeldt. This package started out under the name of caption with a certain user interface. Over time it became clear that there were some deficiencies in the user interface; to rectify these without making older documents fail, Axel introduced caption2. At a later point the syntax of that package itself was superseded, resulting in caption3 and then, finally, that got renamed back to caption. So now older documents using caption will fail whilst documents from the intermediate period will require caption2 (which is listed as superseded on CTAN but is still distributed in the major distributions). So users accustomed to copying their document preamble from one document to the next are probably still continuing to use it without noticing that they are in fact using a version with defective and limited interfaces.

Another example is the fixltx2e package that for many years contained fixes to the LaTeX kernel. In 2015 these were integrated into the kernel so that today this package is an empty shell, only telling the user that it is no longer needed. However, if you process an old document (from before 2015) that loads fixltx2e then of course the fixes originally provided by this package (like the corrections to the floats algorithm) would get lost as they are now neither in the kernel nor in the "empty" fixltx2e package if that doesn't roll back as well — fortunately it does and always did, so in reality it isn't quite an empty shell.

A somewhat different example is the amsmath package, which for nearly a decade didn't see any corrections even though several problems have been found in it over the years. If such bugs finally get corrected, then that would affect many of the documents written since 2000, since their authors may have manually worked around one or another of the deficiencies. Of course, as with the caption package, one could introduce an amsmath2, amsmath3, ... package, but that puts the burden on the user to always select the latest version (instead of automatically using the latest version unless an earlier one is really needed).

3 The document interface

By default LaTeX will automatically use the current version of any class or package — and prior to offering the new rollback concept it always did that unless the package or class had its own scheme for providing versioning, either using alternative names or by hand-coded options to select a version.

3.1 Global rollback

With the new rollback concept all the user has to do (if he or she wants their document processed with a specific version of the kernel and packages) is to add the latexrelease package at the beginning of the document and specify a desired date as the package option, e.g., just as in the first example:

```
\RequirePackage[2018-01-01]{latexrelease}
```

This will roll back the kernel to its state on that day (as described earlier) and for each package and the document class it will check if there are alternate releases available and select the most appropriate release of that package or class in relation to the given date.

3.2 Individual rollback

There is further fine-grain adjustment possible: both `\documentclass` as well as `\usepackage` have a second (little known) optional argument that up to now was used to allow the specification of a "minimal date". For example, by declaring

```
\usepackage[colaction]
           {multicol}[2018-01-01]
```

you specify that multicol is expected to be no older than the beginning of 2018. If only an older version is found, then processing such a document results in a warning message:

```
LaTeX Warning: You have requested, on input
   line 12, version '2018-01-01' of package
   multicol, but only version
   '2017/04/11 v1.8q multicolumn formatting
   (FMi)' is available.
```

The idea behind this approach is that packages seldom change syntax in an incompatible way, but more often add new features: with such a declaration you can indicate that you need a version that provides certain new features.

The new rollback concept now extends the use of this optional argument by letting you additionally supply a target date for the rollback. This is done by prefixing a date string with an equal sign. For example,

```
\usepackage{multicol}[=2017-06-01]
```

would request a release of multicol that corresponds to its version in June 2017.

So assuming that at some point in the future there is a major rewrite of this package that changes the way columns are balanced, the above would request a fallback to what right now is the current version from 2017-04-11. The old use of this optional argument is still available because presence or absence of the = determines how the date will be interpreted.

The same mechanism is available for document classes via the `\documentclass` declaration, and for `\RequirePackage` if that is ever needed.

3.3 Specifying a version instead of a date

Specifying a rollback date is most appropriate if you want to ensure that the behavior of the processing engine (i.e., the kernel and all packages) corresponds to that specific date. In fact, once you are finished with editing a document, you can preserve it for posterity by adding this line:

```
\RequirePackage[⟨today's-date⟩]
               {latexrelease}
```

This would mean that it will be processed a little more slowly (since the kernel may get rolled back and each package gets checked for alternate versions), but it would have the advantage that processing it a long time in the future will probably still work without the need to add that line later.

However, in a case such as the caption package or, say, the longtable package, that might eventually see a major new release after several years, it would be nice to allow the specification of a "named" release instead of a date: for example, a user might want to explicitly use version 4 rather than 5 of longtable when these versions have incompatible syntax, or produce different results.

This is also now possible if the developer declares "named" releases for a package or class: one can then request a named version simply by using this second optional argument with the "name" prefixed by an equal sign. For example, if there is a new version of longtable and the old (now current) version is labeled "`v4`", then all that is necessary to select that old version is

```
\usepackage{longtable}[=v4]
```

Note that there is no need to know that the new version is dated 2018-04-01 (nor to request a date before that) to get the old version back.

The version "name" is an arbitrary string at the discretion of the package author — but note that it must not resemble a date specification, i.e., it must not contain hyphens or slashes, since these will confuse the parsing routine.[2]

3.4 Erroneous input

The user interface is fairly simple and to keep the processing speed high the syntax checking is therefore rather light. Basically the standard date parsing from the kernel is used, which is rather unforgiving if it finds unexpected data.

Basically any string containing a hyphen or a slash will trigger the date parsing which then expects two hyphens (for the case of an ISO date) or two slashes (otherwise) and other than these separators, only digits. If it does find anything else, chances are that you will get a "`Missing \begin{document}`" error or, perhaps even more puzzling, a strange selection being made. For example, `2011/02` may mean to us February 2011 but for the parsing routine it is some day in the year 20 A.D. That is, it gets converted to the single number `201102`, so that, when this number is compared numerically to, say, `20000101`, it will be the smaller number, i.e., earlier, even though the latter is the numerical representation of January 1st 2000.

So, bottom line: do not misspell your dates and all is fine. That hasn't been a problem in the past, so hopefully it will be okay to continue with just this light checking. If not, then we may have to extend the checks made during parsing.

3.5 Advice for early adopters

If your document makes use of the new global rollback features, then it should be processable at any installation later than early 2015, when the latexrelease package was first introduced. If the installation is even older, then it needs upgrading or, at least, one has to add a current latexrelease package to the installation.

However, if your document uses the new concept for individual rollbacks of packages or classes (i.e.,

[2] Of course more sophisticated parsing could fix this, but we use fast and simple parsing that scans for slashes or hyphens with no further analysis.

with the =... syntax in the optional argument), then it is essential to use a LaTeX distribution from 2018 or later.[3] Earlier distributions will choke on the equal sign inside the argument as they will only expect to see a date specification there.

4 The package/class interface

The rollback mechanism for packages or classes is provided by putting, at the beginning of the file containing the code, a declaration section that informs the kernel about existing alternative releases.

These declarations have to come first and have to be ordered by date because the loading mechanism will evaluate them one by one and, once a suitable release is found, it will be loaded and then processing of the main package or class file will end. If there are no such declarations, or if the older releases are all ruled out for one reason or another, processing will continue as normal by reading all of the main file.

The old releases are stored in separate files, one for each release, and we suggest using a scheme such as ⟨*package-name*⟩-⟨*date*⟩.sty as this is easy to understand and will sort nicely within a directory. However, any other scheme will do as well, as the name is part of the declaration.

The contents of this release file is simply the package or class file as used in the past. This means that before making a new version all you need to do is to make a verbatim copy of the current file and give it a new suitable name.[4]

This way it is also straightforward to include older releases after the fact, e.g., to take our famous caption example, Axel could provide the very first version of his package as caption-⟨*some-date*⟩.sty and caption2 as caption-⟨*another-date*⟩.sty in addition to adding the necessary declarations to the current release.

The necessary declarations in the main file are provided by the two commands, \DeclareRelease, and \DeclareCurrentRelease, that must be used in a *release selection* section at the beginning of the file. For each old release you can specify a ⟨*name*⟩, the ⟨*date*⟩ when it was first available and the ⟨*external-file*⟩ that contains the code.

[3] Alternatively you could try to roll the installation forward, by using a current latexrelease package together with a suitable date option.

[4] Instead of making a verbatim copy you may want to adjust the commentary added by docstrip at the top of the file. Though technically correct, it is a bit misleading if the file still contains the phrase "was generated from ...", given that it is now a frozen version representing a particular state in time, rather than being a generated one that can be regenerated any time as necessary.

```
\DeclareRelease
      {⟨name⟩}{⟨date⟩}{⟨external-file⟩}
```

Either the ⟨*name*⟩ or the ⟨*date*⟩ can be empty, but not both at the same time. Not specifying a ⟨*date*⟩ is mainly intended for providing "beta" versions that people can explicitly select but that should play no role in date rollbacks.

The current release also gets a declaration, but this time with only two arguments: a ⟨*name*⟩ (again possibly empty) and a ⟨*date*⟩ since the code for this release will be the rest of the current file:

```
\DeclareCurrentRelease{⟨name⟩}{⟨date⟩}
```

This declaration has to be the last one in sequence as it will end the *release selection* processing.

The order of the other releases has to be from the oldest to the newest since the loading mechanism compares every release declaration with the target rollback date and stops the moment it finds one that is newer than this target date. It will then select the one before, i.e., the last one that is at least as old as the target. Since the \DeclareRelease declarations with an empty ⟨*date*⟩ argument do not play a role in date rollbacks, they can be placed anywhere within the sequence.

As a typical example of a release section the start of the multicol package currently looks as follows because there was a major internal rewrite in April 2018. Note that because of some minor fixes afterwards the actual package date is already June.

```
\NeedsTeXFormat{LaTeX2e}[2018-04-01]
\DeclareRelease{}{2017-04-11}
                 {multicol-2017-04-11.sty}
\DeclareCurrentRelease{}{2018-04-01}
\ProvidesPackage{multicol}[2018/06/26 v1.8t
             multicolumn formatting (FMi)]
```

If the rollback target is not a date but a name, the mechanism works in the same way with the exception that a release is selected only if the name matches. If none of the names is a match, then the mechanism will raise an error and continue by using the current release.

5 Special considerations for developers

While loading an older release of a package or class, both types of release declarations are made no-ops, so that, in case the files containing the code also have such declarations, they will not be looked at or acted upon. This makes it possible to simply move the code from an old release into a new file without the need to touch it at all. Of course, removing those declarations doesn't hurt and will make loading a tiny fraction faster.

As mentioned earlier, best practice for release names is to append the release date to the package or class name, but the ⟨*external file*⟩ argument also allows other naming schemes.

You may have wondered why you have to make a declaration for the current release, given that later on there will be a `\Provides...` declaration that also contains a date and a version string and thus could signal the end of the release declaration section. The reason is as follows: if you want to give your current release a name, then it is best practice to make that name something simple like `v4` (and keep it that way) even though your current package is technically already at `v4.2c` and is listed that way in the `\ProvidesPackage` declaration. For the same reason (given that not every minor change will be provided as a separate version to which people can roll back), the ⟨*date*⟩ in `\DeclareCurrentRelease` reflects when that major release was first introduced. Thus, after a while that date may well be earlier than the current package date.

5.1 Early adopters

For one or two years after the introduction of this new method, there is a danger that people with older installations will pick up an individual package from, say, CTAN that contains release declarations with which their kernel (from 2017 or earlier) is unable to cope. It may therefore be a good idea for developers to additionally add the following lines at the top of packages or classes when using the new rollback feature:

```
\providecommand\DeclareRelease[3]{}
\providecommand\DeclareCurrentRelease[2]{}
```

This way the declarations will be bypassed in case the kernel doesn't know how to deal with them.

As an alternative one could add a statement that requires a minimal kernel version, i.e.:

```
\NeedsTeXFormat{LaTeX2e}[2018-04-01]
```

so that users get a clear error message that they need to update their installation if they want to use the current file.

5.2 New major release in beta

If you are working on a new major release of your package or class, you may want to get it out into the open so that people can try it and provide feedback. In that case the current release is still the official release which should be selected by default, and the "beta" version should only be selected if explicitly requested. To achieve that you could add

`\DeclareRelease{beta}{}{`⟨*external-file*⟩`}`

before

`\DeclareCurrentRelease{}{`⟨*some-date*⟩`}`

so that testers can explicitly access your new version by asking for it via

`\usepackage[`⟨*options*⟩`]{`⟨*package*⟩`}[=beta]`

while everyone loading the package without the extra optional argument would get the current release.

5.3 Two major releases in use

One special scenario for which this method is only partially suitable is the case where we have two major releases that are in continuing parallel use and that are both under active maintenance (i.e., receive bug fixes and other updates once in a while). In that case it is necessary to make one version the primary release and allow the other (and its updates) to be accessed only via names: a date rollback can obviously work for only one line of development.

For example, if both `v4` and `v5` of package foo are in use and you consider `v5` as being the go-forward version (even though you are still fixing bugs in the `v4` code), then you can deploy a strategy as in the following example:

```
% last v4 only release:
 \DeclareRelease{}{2017-06-23}
                  {foo-2017-06-23.sty}
% first v5 release:
 \DeclareRelease{}{2017-08-01}
                  {foo-2017-08-01.sty}
% patch to v4 after v5 got introduced:
 \DeclareRelease{v4.1}{}
                  {foo-v4-2017-09-20.sty}
% patch to v5:
 \DeclareRelease{}{2017-08-25}
                  {foo-2017-08-25.sty}
% another patch to v4:
 \DeclareRelease{v4.2}{}
                   {foo-v4-2017-10-01.sty}
% nickname for the latest v4 if you want
% users to have simple access via a name:
 \DeclareRelease{v4}{}
                  {foo-v4-2017-10-01.sty}
% current v5 with further patches:
 \DeclareCurrentRelease{v5}{2018-01-01}
```

This way users can use `\usepackage{foo}[=v4]` to get the latest v4 release or use the more detailed release names such as `[v4.1]`. This means that if package foo is requested at version v4 (or one of its sub-releases), it will not change even if there is a general rollback request via latexrelease.

Normally, this should be just fine, but if you really require automatic date rollback functionality on both major versions, because the two are really equal in rank, then you are essentially saying they are independent works with some common root. In that case you should give them two separate names,

e.g., call the older version foo-v4 when you introduce version 5 of foo and from that point on manage the history independently.[5]

5.4 Fine grained control (if needed)

As mentioned earlier, the interface is deliberately designed to be simple and easy to use. As a price, each rollback point is (by default) a separate file. The idea behind this is that there is not much point in managing each and every small change as a rollback point, but only those that possibly alter the behavior of a package within the document so that, when processing older documents, it is important to be able to get back to an earlier state.

However, if you find yourself in a situation where you have many rollback files with only minor differences, and you consider this unsatisfactory, then here is one other command at your disposal that you can use to combine several files into a single file. Within a file corresponding to a `\DeclareRelease` declaration you can use

```
\IfTargetDateBefore{⟨date⟩}
    {⟨before-date-code⟩}{⟨after-or-at-date-code⟩}
```

This must be used after the *release selection* section (if present) and has the following effect: If the user requested, say, `[=2017-06-01]` then the mechanism first selects the file that is supposed to be current on that date, i.e., the release that was introduced on that date or is the last one that was introduced before that date. Now, if in this file we have a statement like the above, then the ⟨*date*⟩ is compared to `2017-06-01` and depending on the outcome either ⟨*before-date-code*⟩ or ⟨*after-or-at-date-code*⟩ is executed.

This way a single external file can hold rollback information for several patches on distinct dates, but of course, the burden is then on the developer to add the appropriate declarations, which is a little more work than just copying and renaming files.

The alternative is to use `\IncludeInRelease` and `\EndIncludeInRelease`. The latexrelease package documentation [1] gives some advice on how to apply those commands.

5.5 Using `l3build` for source management

If you use `l3build` [2] for managing your sources, then it is necessary to ensure that the files for the old releases are copied into the distribution. To support this, the default configuration for `l3build` specifies

```
sourcefiles = {"*.dtx", ".ins",
                 "*-????-??-??.sty"}
```

i.e., all `.dtx` and `.ins` files, together with all `.sty` files matching the naming convention suggested in this article, are automatically included in the build.

If you prefer a different naming convention you have to adjust this setting in the `build.lua` file of your project. Otherwise you are ready to go without any adjustments.

6 Command summary

6.1 Document interface, for users

For a global rollback of kernel and packages, use

```
\RequirePackage[⟨target-date⟩]{latexrelease}
```

at the beginning of your document.

To request a rollback for a single package or class, use the second optional argument with the date preceded by an equal sign, i.e.,

```
\documentclass[⟨options⟩]{⟨class⟩}[=⟨date⟩]
\usepackage [⟨options⟩]{⟨package⟩}[=⟨date⟩]
```

6.2 Package and class interface, for developers

To declare an old or special release, use

```
\DeclareRelease
        {⟨name⟩}{⟨date⟩}{⟨external-file⟩}
```

Leave the ⟨*name*⟩ argument empty if rollback should be only via dates. Leave the ⟨*date*⟩ empty if this special release should be accessible only via its name.

Always finish this *release selection* section with a declaration for the current release:

```
\DeclareCurrentRelease{⟨name⟩}{⟨date⟩}
```

In this declaration you must provide a ⟨*date*⟩ but the ⟨*name*⟩ can be left empty (which is the usual case).

Within a release file (but after the *release selection* section), you can specify conditional code to be selected based on a requested rollback date by using:

```
\IfTargetDateBefore{⟨date⟩}
    {⟨before-date-code⟩}{⟨after-or-at-date-code⟩}
```

References

[1] The LaTeX Team. *The* `latexrelease` *package*, April 2015. Available at `https://www.latex-project.org/help/documentation`.

[2] The LaTeX Team. *The* `l3build` *package — Checking and building packages*, March 2018. The manual `l3build.pdf` should be part of your installation. Run "`texdoc l3build`" to find it. See also `https://ctan.org/pkg/l3build`.

◇ Frank Mittelbach
`https://www.latex-project.org`

[5] While in rare cases this might be the best approach, try to avoid it as long term management will be problematic, to say the least.

Font loading in LaTeX using the fontspec package: Recent updates

Will Robertson

1 Introduction

The fontspec package (ctan.org/pkg/fontspec) is, I am astonished to say, close to 15 years old. I honestly don't know where all the time has gone. The interface that fontspec provides was originally very simple: load a font. The font feature interface followed quickly after, and was originally based around macOS's AAT font technology, which is largely obsolete these days. Today, fontspec is targeted mainly towards the use of OpenType fonts, although the legacy AAT code is still functional. Another font technology, Graphite, can be used when running the XeTeX engine but the fontspec interface is extremely minimal (and rather undocumented).

This article will discuss some of its more recent updates, possible future interfaces, and what I now consider to be some 'best practices' for fontspec use.

2 Font loading

There are a number of ways to load or set up fonts in fontspec:

- `\fontspec`
- `\setmainfont`
- `\newfontfamily`
- `\defaultfontfeatures`

As the package has grown, the best way to combine these possibilities is probably not so clear, especially to new users.

Originally, fontspec was written for the XeTeX engine on Mac OS X, with fonts being accessed via the operating system, which provided automated interfaces for selecting, e.g., 'italic' and 'bold'. Thus, early on, users became accustomed to the idea of using the human-oriented font names known to the system, as in writing

```
\setmainfont{Hoefler Text}
```

in their document, and then everything 'just worked' without additional configuration. This feature later influenced luaotfload (for LuaTeX) to provide a similar mechanism, by scanning through known font directories and constructing its own database of font files, their 'logical' names, and the relationships between shapes within a particular family. And thus it is that, right now, you can write

```
\setmainfont{TeX Gyre Pagella}
```

in a LuaLaTeX document and the correct font will be chosen and bold and italic styles will 'just work' since the TeX Gyre fonts are included in the standard TeX distributions.

However, loading font families in these ways has confusing edge cases, and it increases document portability problems, and it can be slow. Over time I have moved away from using this feature and I think long term it would be better to de-emphasise its use. I now recommend all users to load their fonts by filename.

The most straightforward means to do this is like so:

```
\setmainfont{texgyrepagella-regular.otf}[
 ItalicFont     = texgyrepagella-italic.otf    ,
 BoldFont       = texgyrepagella-bold.otf      ,
 BoldItalicFont = texgyrepagella-bolditalic.otf,
]
```

But here there is a lot of duplication and a more streamlined way is:

```
\setmainfont{texgyrepagella}[
 Extension      = .otf        ,
 UprightFont    = *-regular   ,
 ItalicFont     = *-italic    ,
 BoldFont       = *-bold      ,
 BoldItalicFont = *-bolditalic,
]
```

Whereas in the first case the mandatory argument `{texgyrepagella-regular.otf}` was a direct reference to the 'regular upright' font file, in the second case `{texgyrepagella}` is merely a shorthand name from which the filenames are constructed.

2.1 The case against loading by font name

I claimed above that there were some problems with loading fonts by their external name instead of their filename. Let's discuss these in more detail.

Edge cases. Modern font families aren't the font families as we once knew them. It's now common to purchase families with literally tens of styles, with weights from extra light to ultra black and a multitude of styles. (I recently purchased Gill Sans Nova and it arrived as 43 separate `.otf` files.) Now that we have OpenType variable fonts, families of arbitrary complexity may start to appear. Software such as luaotfload needs to use heuristics to establish the relationships between fonts, and there can be no general solution to this task. Therefore, sometimes it fails to select the correct 'bold' font in particular circumstances; therefore, sometimes I get bug reports.

Document portability. If you are loading a font by name, it has to be installed somehow on that local computer. Well, of course any font that you load has to be installed; but if it is by name it's possible for it to be installed in a(ny) number of different locations,

and the way that the font loading heuristics work, there could be a(ny) number of valid names that the font can be referred to. When failure happens, is it because you haven't installed it correctly, or have other software changes made the names no longer match, or has a database not been updated when it should have, or is it a XeTeX vs LuaTeX difference, or . . . ?

Whereas if a font is loaded by filename it is generally clear, or at least discoverable, where to look and what to fix if the font cannot be found. It's also easier to set up a multiple-computer environment where all your fonts are stored, say, in Dropbox or on a secure network drive, and there is no 'font installation' process anymore. As an example, I store my 'local' `texmf` tree in Dropbox and have a shell alias set up as

```
alias texmf="sudo tlmgr conf texmf \
               TEXMFHOME '~/Dropbox/texmf'"
```

This allows me to run `texmf` on a new (or freshly updated) computer and as long as I'm connected to Dropbox I'll be able to load all the fonts I have stored in that tree.

Slow. This is an obvious one. When luaotfload scans through your font directories, it can take quite some time. Loading fonts by filename simply doesn't have this problem. Another aspect of the 'slow' problem is that from an operating system perspective it can get slower (on older computers especially) as one installs tens or hundreds of additional fonts. (Microsoft Word's WYSIWYG font menu is a good example of this.)

3 The interface for font features

The 'plain' approach to selecting OpenType font features requires the use of raw feature tags, like `+lnum` to activate 'lining numbers' or `+dlig` to activate 'discretionary ligatures'.

```
\font\x="[EBGaramond12-Regular.otf]:+lnum;+dlig"
```

OpenType has quite a number of standard feature tags, but they are not organised into a hierarchy. In the early days of the fontspec package, its feature loading interface was modelled on Apple's AAT font technology. This competitor to OpenType, quietly fading in popularity, used a keyval approach to font features instead, which mapped more naturally to a user interface. The fontspec interface to the plain example above is:

```
\fontspec{EBGaramond12-Regular.otf}[
  Numbers   = Lining        ,
  Ligatures = Discretionary ,
]
```

UltraLight Light Book Medium Semibold **Bold**
Heavy ExtraBold UltraBold
CnUltraLight CnLight CnBook CnMedium CnSemibold **CnBold**
CnHeavy CnExtraBold CnUltraBold

Figure 1: Upright 'normal' and 'condensed' weights of Gill Sans Nova.

Deco-Regular
Shadowed-Light
Shadowed-Medium
SHADOWED-OUTLN
INLINE-COND
INLINE-EXTRALT
INLINE-LIGHT
INLINE-REGULAR
INLINE-BOLD

Figure 2: Additional 'unusual' fonts in the Gill Sans Nova family.

It is also possible to change the font features for the currently-selected font using `\addfontfeatures`. A relatively recent improvement to fontspec dramatically improved the reliability and internal logic for such commands. Now, writing

```
\addfontfeatures{Numbers = OldStyle}
```

will explicitly deactivate `Numbers = Lining` before adding the new `+onum` OpenType tag. OpenType features are now provided consistently with `Off` and `Reset` variants (e.g., `Numbers = OldStyleOff` and `Numbers = OldStyleReset`) which interact properly with the feature loading logic to either forcibly deactivate a feature or to reset that particular fontspec feature to its default state.

4 Typical example

Let's now introduce a commercial font, and use this in an example of setting up a custom font from scratch; I'll use 'Gill Sans Nova' as it comes with a range of weights and styles. This font family has upright and italic fonts in both 'normal' and 'condensed' widths with a large range of weights (see Figures 1 and 2).

When using Gill Sans Nova myself, I usually want the Medium and Bold weights paired, so I created a file in my local `texmf` directory named `gill-sans-nova.fontspec`, which looks like this:

```
\defaultfontfeatures[gill-sans-nova]{
 UprightFont    = GillSansNova-Medium.otf      ,
 ItalicFont     = GillSansNova-MediumItalic.otf,
 BoldFont       = GillSansNova-Bold.otf        ,
 BoldItalicFont = GillSansNova-BoldItalic.otf  ,
}
```

This file allows me to write, without any additional options, `\setmainfont{gill-sans-nova}` or other fontspec font selection command. Additionally, I could write:

```
\newfontfamily\bookfont{gill-sans-nova}
```

and then use `\bookfont` in the setup of the typography of the document.

To use a different selection of fonts from the family, I can create another `.fontspec` file and load that in the same way.

The `\defaultfontfeatures` command isn't *required* to be in a `.fontspec` file; it could be in a class or package file, or even just pasted into a document preamble. But this approach keeps your preamble as tidy as possible if you have a series of font family definitions you will (probably) reuse between documents.

4.1 Feature possibility: font collections?

In the example above, `\newfontfamily` was briefly mentioned for creating a macro that switches the font to a pre-defined family. This is akin to extending the selections provided in LaTeX with its `\rmfamily`, `\sffamily`, and `\ttfamily` definitions.

In other contexts it may be desirable to define a single command to replace all of the standard `rm`/`sf`/`tt` fonts. One interface we could imagine here might look like

```
\setmainfont{pagella}% default 'collection'
\setmainfont{aldus}[Collection={examples}]
```

to support a book with a series of 'examples', which are typeset in a different style than the main text. The document might look like:

```
this is the \textsf{main} text
% uses the default fonts

\selectfontcollection{examples}
this is some \textsf{example} text
% uses fonts from the "examples" collection
```

Over the next few months (or years, if my plans go awry), I'll experiment with implementing some interfaces along these lines for user testing. Please keep an eye out; I would certainly welcome additional feedback.

5 Additional NFSS declarations

The standard '`...Font`' variants that could be loaded within fontspec have thus far been restricted to the relatively standard set of:

`UprightFont` / `ItalicFont` / `BoldFont` / `BoldItalicFont` / `SlantedFont` / `BoldSlantedFont`

(And for each of these variants both normal and small caps shapes are allowed.)

However, LaTeX's font selection scheme allows complex mappings along what it terms the 'shape' and the 'series' axes. These can be assigned using fontspec's `FontFace` option:

```
\defaultfontfeature+[gill-sans-nova]{
 FontFace = {uu}{n}{GillSansNova-UltraLight.otf},
 FontFace = {ll}{n}{GillSansNova-Light.otf     },
 FontFace = {hh}{n}{GillSansNova-Heavy.otf     },
 FontFace = {xx}{n}{GillSansNova-ExtraBold.otf },
}
```

Now, for fonts in which only a range of weights need to be defined, as here, the fontspec interface to this is a little awkward. I have been discussing the idea of a more friendly syntax with a number of users recently. On the other hand, for truly arbitrary font shapes that cannot be categorised in a standard way (e.g., the 'Deco' Gill Sans font shown in Figure 2), the more flexible `FontFace` option provides the most generic interface.

6 'Inner' emphasis and 'Strong' emphasis

By default, writing `\emph{\emph{abc}}` produces an upright 'abc' as LaTeX knows that italic text needs to switch (back) to upright text for emphasis. Some typesetters prefer instead to emphasise within italic text with small caps, and LaTeX 2_ε provides the `\eminnershape` interface for defining the behaviour of nested emphasis beyond a single level.

In fontspec, I became interested in the idea of supporting arbitrary levels of nesting. For emphasis this is presumably of limited utility, but later in this section it will become more clear why this is useful. Therefore, fontspec now supports arbitrary nesting using (say)

```
\emfontdeclare{\itshape        ,
               \upshape\scshape,
               \itshape        ,
}
```

which will ask emphasised text to switch to italic, then upright small caps, then italic small caps, as shown in Figure 3. Each command sequence separated by commas is applied successively.

Just above, we saw that fontspec provides the `FontFace` option to load a range of font weights in

Figure 3: An example of nested emphasis.

Figure 4: An example of nested `\strong` emphases.

a single family. Inspired by HTML's `<strong>` tag, I have added `\strong` to fontspec as a sibling to `\emph`. This is the command which makes it sensible to support arbitrary nesting, given the large range of weights seen in modern font families.

To follow from the previous Gill Sans Nova example, to setup `\strong` with nesting I can write:

```
\strongfontdeclare{
  \bfseries                  ,
  \fontseries{hh}\selectfont ,
  \fontseries{xx}\selectfont ,
}
```

This will produce the results shown in Figure 4.

Note there is no `\weaken` command to change weights in the opposite direction! More seriously, I can imagine generalising the interfaces here to a pair of code-level interfaces such as `\fontseriesup` and `\fontseriesdown` to change weights along a scale.

These interfaces will be subject to change as improved methods for loading additional bold weights are developed. At the moment the `\emfontdeclare` and `\strongfontdeclare` commands are global for all fonts; there is no current way to have font-specific declarations there.

7 Custom encodings

We all know that fonts aren't perfect, and while many problems can be solved by choosing a better font, this is not always possible. The fontspec package, in concert with recent LaTeX 2_ε changes around the `TU` font encoding, now allows per-font customisation for symbols and accent support. As an example, say I'm creating a poster with some Sanskrit text but I'm using a Western font; I can choose to redefine the underdot accent `\d` by loading two fonts like so:

```
\newfontfamily\sanskritfont{CharisSIL}
\newfontfamily\titlefont{Posterama}[
  NFSSEncoding=fakedotaccent
]
```

Of course, this needs some extra setup beforehand:

```
\DeclareUnicodeEncoding{fakedotaccent}{
  \input{tuenc.def}
  \EncodingCommand{\d}[1]{%
    \hmode@bgroup
      \o@lign{\relax#1\crcr\hidewidth
      \ltx@sh@ft{-1ex}.\hidewidth}%
    \egroup
  }
}
```

Then later in the document, no additional work is needed:

```
...{\titlefont    KALITA\d M}...
...{\sanskritfont KALITA\d M}...
```

The first uses the 'fake' accent; the second the real Unicode glyph.

There is a huge caveat to this, which is that LaTeX can't intercept Unicode accents. So if you have text that is written literally as '`KALITAṂ`' in the source, LaTeX can't correct this for you, and you'll end up with whatever the font gives you. This limitation could be addressed by adding a pre-processing stage to the LaTeX typesetting, but there is no built-in mechanism to do this.

8 Conclusion

While the fontspec package has undergone extensive development over the years, the core concepts are the same: load fonts in a flexible way. But I think it's fair to say that as the package has grown and the interfaces have become more complex, the interface is not as clear as it could be. Will I ever re-write fontspec entirely? Probably not. But with LaTeX3 interfaces to consider, it's probable that a slimmed-down version of fontspec will need to make an appearance somewhere or another.

◇ Will Robertson
School of Mechanical Engineering
The University of Adelaide, SA
Australia
will.robertson (at) adelaide dot edu dot au
wspr.io/fontspec

Supporting color and graphics in expl3

Joseph Wright

1 Introduction

The expl3 language has grown over the past decade to cover a wide range of programming tasks [4]. However, at present there are a number of areas where expl3 offers little or no 'core' support and which will need functionality at this level. Here, I'll be focussing on one in particular: color and graphics support.

In the classical LaTeX 2$_\varepsilon$ setup, the `picture` environment along with the packages graphics [5] and color provide the basis for this area. To allow driver-dependent operations, a set of definition files is loaded by graphics to map user operations to driver-specific instructions. Nowadays, these are managed by the LaTeX team in the bundle graphics-def [2].

In addition to this core support, a number of well-established contributed packages offer significant additional features. Particularly notable here are xcolor [1], which allows user-friendly mixing of colors, and Ti*k*Z/pgf [6], an extremely rich and versatile system for the programmatic creation of graphics.

Here, I will look at recent efforts to begin providing a similar level of overall functionality *via* expl3. Central to these efforts is the availability of a fast, expandable and accurate software floating-point unit (FPU) within expl3. This provides a base on which many graphics-related functions can build: calculations are a core part of many image-related functions.

2 The driver layer

Unlike the LaTeX 2$_\varepsilon$ situation, where the graphics and color driver code is managed (somewhat) separately from the kernel, the expl3 versions are part of the core distribution. Development of the driver code in expl3 has been informed by recent efforts to standardise the LaTeX 2$_\varepsilon$ versions, and *vice versa*.

As new features are added to expl3 which require driver support, the driver layer is being adjusted to match. This means that unlike in LaTeX 2$_\varepsilon$, for expl3 there should be a single set of definitive driver files, supported by the team and usable by (and documented for) others.

3 Colo(u)r

The LaTeX 2$_\varepsilon$ (required) package color provides a base interface for using pre-defined colors. However, one of the most common ways to use a color is to describe it as a mix of base colors: red, green and blue, or cyan, magenta, yellow and black. The xcolor package provides a convenient 'expression' interface for creating mixtures: `\color{red!50!blue}`.

Supporting this mixing, conversion between different color models, and other features such as spot colors, are all (largely) covered in the experimental l3color package [3]. Using the LaTeX3 FPU makes much of the core support very easy to implement: the various pieces of mathematics can be expressed directly, rather than requiring complex dimension shuffling.

At present, the nature of input in l3color is limited to the 'simple' color expressions defined by xcolor: feedback on what is helpful to end users would be very welcome.

4 Image inclusion

At present, expl3 support for image inclusion is only ready at the driver level. Implementing a code-level set of `\image_...` functions is on the 'to do' list, and is likely straightforward.

5 Drawing

Whilst the `picture` environment of the LaTeX kernel does provide a way to create simple graphical elements, today perhaps the most powerful tool for this task is Ti*k*Z/pgf. Reimplementing all of the latter may seem excessive, but there are several reasons to explore this. First, a core aim of expl3/LaTeX3 work is to eventually provide a full set of features for supporting document preparation, certainly providing code-level tools for all common tasks. Coupled to this, an expl3 implementation will have API consistency with the rest of the code: mixing Ti*k*Z and expl3 can be tricky. We are also able to use existing expl3 tools in the implementation and usage. Finally, there is the potential offered by the LaTeX3 FPU: this avoids using dimensions for floating point work, and so also avoids the `Dimension too big` issue that comes up from time-to-time using Ti*k*Z.

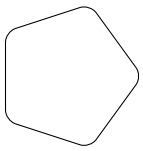
Much like expl3, pgf is divided into different layers: these line up as show in Table 1. There is good alignment, and thus in many ways it is simply a case of re-creating the macros with new names. Of course, there is more to do than that: for example, the use of the LaTeX3 FPU means that co-ordinate expressions are processed expandably by l3draw, with a knock-on effect in usage. However, as far as possible the interfaces in l3draw retain the same arguments as those in pgf.

6 Examples

At the time of writing, l3draw is very much a work in progress. However, the core idea of constructing paths is fully implemented. For example, a simple geometric shape including smoothing joins:

Table 1: Comparison of Ti*k*Z/pgf and l3draw concepts

Layer	Ti*k*Z/pgf	l3draw
System	`\pgfsys@moveto`	`\driver_draw_moveto:nn`
Base	`\pgfpathmoveto`	`\draw_path_moveto:n`
Interface	`\draw`	—



```
\draw_begin:
  \draw_path_corner_arc:nn { 4pt } { 4pt }
  \draw_path_moveto:n
    { \draw_point_polar:nn { 0 } { 1cm } }
  \int_step_inline:nnnn { 72 } { 72 } { 359 }
    {
      \draw_path_lineto:n
        {
          \draw_point_polar:nn { #1 } { 1cm }
        }
    }
  \draw_path_close:
  \draw_path_use_clear:n { stroke }
\draw_end:
```

The new code also integrates with existing ideas such as coffins. Here, we draw a line to the center of typeset text:


```
\draw_begin:
  \draw_path_moveto:n { 0cm , 0cm }
  \draw_path_lineto:n { 0cm , 1cm }
  \draw_path_use_clear:n { stroke }
  \hcoffin_set:Nn \l_tmpa_coffin
    { This~is~text. }
  \draw_coffin_use:Nnn \l_tmpa_coffin
    { hc } { vc }
\draw_end:
```

We can also exploit the expandable nature of the FPU:

22.72949518869545pt,-17.11517943480897pt

```
\tl_set:Nx \l_tmpa_tl
  {
    \draw_point_intersect_circles:nnnn
      { (0,0) } { 1cm }
      { (sqrt(2),sqrt(3)) } { 1cm }
      { 1 }
  }
\tl_to_str:N \l_tmpa_tl
```

Thus, l3draw is ready for application in expl3 contexts which require drawing. Over time, we expect to cover essentially the entire API provided by pgf's core, plus probably node handling (loaded by pgf but not technically part of the core of the bundle).

References

[1] U. Kern. Extending LaTeX's color facilities: the xcolor package, 2016. ctan.org/pkg/xcolor

[2] LaTeX Project. Color and graphics option files, 2018. ctan.org/pkg/graphics-def

[3] LaTeX Project. Experimental LaTeX3 concepts, 2018. ctan.org/pkg/l3experimental

[4] LaTeX Project. The expl3 package and LaTeX3 programming, 2018. ctan.org/pkg/expl3

[5] LaTeX Project. The graphics bundle, 2018. ctan.org/pkg/graphics

[6] T. Tantau and C. Feuersänger. Ti*k*Z and pgf, 2015. ctan.org/pkg/pgf

◇ Joseph Wright
LaTeX Project
joseph dot wright (at)
morningstar2.co.uk

siunitx: Past, present and future

Joseph Wright

Abstract

The siunitx package provides a powerful toolkit for typesetting numbers and units in LaTeX. By incorporating detail about the agreed rules for presenting scientific data, siunitx enables authors to concentrate on the meaning of their input and leave the package to deal with the formatting. Here, I look at the background to the package, what led me from version 1 to version 2, and why version 3 is now under development.

1 Introduction

Typesetting units naturally lends itself, in TeX, to using macro support. The formal rules for SI units [1] link unit names with unit symbols, and it's not surprising that several authors have created packages that tackle some or all of the subtleties involved. I've detailed some of these issues and packages before, when I last looked at siunitx for *TUGboat* [6]. Here, I'll briefly recap some of those key points, then explore what is driving efforts toward a third version of the package.

2 Early days

Before siunitx, there were a variety of units-related packages, including one called SIunits [3], which deals with providing semantic macros for units. My involvement started when I followed up an apparently-innocuous post to `comp.text.tex` from Stefan Pinnow in November 2007, reporting a straightforward bug

> `I want to report that \reciprocal, \rpsquare, \rpcubic, etc. output is written as "-1" instead of a $-1$, when the package option "textstyle" is used. I tried to contact Mr. Heldoorn, but he didn't answer until now. Does anyone have an idea what to do?`

Being young(ish) and foolish, after looking at the bug itself and finding that SIunits was no longer maintained, I volunteered to pick up the package. Even more foolishly, I then followed up with the following in November 2007:

> `As some of you may have noticed, following a recent bug report concerning the SIunits package, I have taken over as the package maintainer. I have uploaded a bug fix for the specific issue to CTAN, and so hopefully it will appear within a day or two.`
>
> `It has been suggested by the maintainer of the SIstyle package that integration of the two be would worth considering. Other suggestions have also been made in the newsgroup and by private mail. I am therefore planning to review the existing situation and see what improvements are needed/desirable. As well as SIunits and SIstyle, I am going to look at numprint, units, unitsdef and hepunits for inspiration/points to consider/etc.`
>
> `So far, I have some outline ideas, for example: ...`

I soon had quite a list, and work on the new package began in earnest, and by February 2008 the first testing release was out. After a little more work, and a rename, the first official version of siunitx hit CTAN on the 16th of April 2008 [7].

3 Key features

The core features of siunitx have been present from that first version, and are pretty well-known. I'd summarise them as

- Automatic, semantic formatting of quantities (numbers with units)
- Parsing and manipulation of numbers
- Control of printing of numbers, units and quantities
- Alignment of numbers in tables
- Unified key–value interface for controlling options

For me, the package has always been about *units*, and the fundamental idea that input such as

`\joule\per\mole\per\kelvin`

can give $\mathrm{J\,mol^{-1}\,K^{-1}}$ or $\frac{\mathrm{J}}{\mathrm{mol\,K}}$ or $\mathrm{J/(mol\,K)}$, depending on the settings in force. This idea has been there since day one, and the code has carried through more-or-less unchanged.

4 From version one to version two

Version one of siunitx worked well for delivering on those key features. Releases progressed rapidly, culminating in version 1.4c in February 2010. However, adding new features was a problem: internally it was a bit of a mess. For example, if you look at the old code you'll see:

- Internals *other than unit parsing* taken from existing packages and somewhat haphazard

- Sub-optimal key–value choices
- Essentially no internal API
- Poor self-coded loops
- ...

Around this time, Will Robertson contacted me to ask what I thought of the LaTeX3 language expl3 [4]. This was before I joined the LaTeX team, and expl3 looked a bit different than today, but the central ideas were all there. I liked the ideas, but at the time was a bit wary of loading an external library (and thus a dependency). So I started by picking out the ideas and re-coding them in my development setup for version two. It soon became clear that I needed a *lot* of the ideas, and I realised that I'd be much better off just requiring expl3.

Work on the second version of siunitx took me into expl3 programming, and asking the team for lots of features. In particular, I wanted to have key–value support built-in, rather than needing to use another package to do that. So I wrote some code, which I called keys3, to solve the problem. It turned out that was my application to join the team: it's today l3keys in expl3 itself!

The rewrite gave me a chance to significantly revise internal API aspects, and to significantly improve performance. It also came of course with new features for users, and completely new names for the key–value interface. In the v2.0 release, I didn't include backward-compatibility: I soon learned, and that's been there since a few days after the second generation release.

5 From version two to version three

Version two retains most of the features that version one had, but as well as the good ones, it turns out it keeps some of the bad ones too! In particular

- Assumptions about fonts: OpenType, *etc.*
- No code-level API expl3
- Internals still too messy
- Testing the PDF documentation only
- Monolithic source
- Still too slow

5.1 Font control

The font assumptions carry all the way through from SIstyle [2], and which I adjusted only slightly. The approach currently used is

1. Detect current font type using LaTeX internal data
2. Insert everything inside `\text`
3. Apply `\ensuremath` inside the box
4. Perhaps apply `\text` again (for text mode output)
5. Force the font with *e.g.* `\mathrm` or `\rmfamily`

That requires a lot of work, and more importantly is unreliable: it's not always easy to get the right font 'inside' the output section. It also fails very badly with OpenType math mode fonts, where the ideas in classical TeX about math families simply don't apply. So there is a new approach: for version three

1. Detect current font type using LaTeX internal data
2. Set any aspects *that are needed*
3. Only use an `\mbox` if math version has to be altered

This 'minimal change' approach is much faster than the current one, and is much better at respecting font changes. I'm still finalising compatibility for current edge-case setups, but I believe the new code is much preferable overall.

5.2 The code API and testing

The development of siunitx version two very much parallels that of expl3 as a truly usable language: in the time I've been using it, expl3 has gone from a set of clever experiments to a well-established approach to coding in TeX. But keeping all of siunitx up-to-date with ideas has been tricky.

The biggest issue is that when I wrote the current release code, there were just user commands such as `\num` and internal implementation. However, it's now clear that each user command should have a *documented* code-level interface. Moreover, these interfaces should all be properly tested: the team have created l3build precisely for that [5]. Combining these ideas, the new code will take the input

```
\siunitx_unit_format:nN
  { \joule \per \mole }
  \l_tmpa_tl
\tl_show:N \l_tmpa_tl
```

and create as output

```
> \l_tmpa_tl=
  \mathrm {J}\,\mathrm {mol}^{-1}
```

Notice that this is easy to test, and highlights another new idea: the parsing step should produce the same results as a user typing in the formatting 'by hand'.

5.3 First alpha release

At the time of writing, the development of version three has reached the first alpha stage: the code is usable but there are real gaps. Currently, all of the following are working:

- Core functionality:
  - Unit parsing and formatting
  - Real number formatting
  - Tabular columns
- Existing API: `\num`, `\SI`, `\si`, `S`-column
- New (experimental) document API: `\unit`, `\qty`

There are some big areas left to do, such as multi-part numbers, ranges, lists and most importantly the compatibility layer for dealing with existing documents. However, that is all relatively manageable, and I expect to be done around the end of the year. So 2019 should see siunitx reach version 3.0.0, and I hope retain its place as *the* units package for LaTeX.

References

[1] Bureau International des Poids et Mesures. The international system of units (SI), 2010. bipm.org/en/measurement-units/

[2] D. Els. The SIstyle package, 2008. ctan.org/pkg/sistyle

[3] M. Heldoorn and J. Wright. The SIunits package: Consistent application of SI units, 2007. ctan.org/pkg/siunits

[4] LaTeX Project. The expl3 package and LaTeX3 programming, 2018. ctan.org/pkg/expl3

[5] LaTeX Project. l3build: Checking and building packages, 2018. ctan.org/pkg/l3build

[6] J. Wright. siunitx: A comprehensive (SI) units package. *TUGboat* 32(1):95–98, 2011. tug.org/TUGboat/tb32-1/tb100wright-siunitx.pdf

[7] J. Wright. siunitx — a comprehensive (SI) units package, 2018. ctan.org/pkg/siunitx

◇ Joseph Wright
Morning Star
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2.co.uk

Appendix: demos

Simple number formatting:

123	`\num{123}` \\
1234	`\num{1234}` \\
12 345	`\num{12345}` \\
0.123	`\num{0.123}` \\
0.1234	`\num{0,1234}` \\
0.123 45	`\num{.12345}` \\
3.45×10^{-4}	`\num{3.45d-4}` \\
-10^{10}	`\num{-e10}`

Angles:

10°	`\ang{10}` \\
12.3°	`\ang{12.3}` \\
4.5°	`\ang{4,5}` \\
1°2′3″	`\ang{1;2;3}` \\
1″	`\ang{;;1}` \\
10°	`\ang{+10;;}` \\
−0°1′	`\ang{-0;1;}`

Units as macros:

$$\mathrm{kg\,m\,s^{-2}}$$
$$\mathrm{g\,cm^{-3}}$$
$$\mathrm{V^{2}\,lm^{3}\,F^{-1}}$$
$$\mathrm{m^{2}\,Gy^{-1}\,lx^{3}}$$
$$\mathrm{H\,s}$$

```
\si{\kilo\gram\metre\per\square\second} \\
\si{\gram\per\cubic\centi\metre}        \\
\si{\square\volt\cubic\lumen\per\farad} \\
\si{\metre\squared\per\gray\cubic\lux}  \\
\si{\henry\second}
```

Quantities:

$$1.23\,\mathrm{J\,mol^{-1}\,K^{-1}}$$
$$0.23 \times 10^{7}\,\mathrm{cd}$$
$$1.99/\mathrm{kg}$$
$$1.345\,\tfrac{\mathrm{C}}{\mathrm{mol}}$$

```
\SI[mode = text]{1.23}{J.mol^{-1}.K^{-1}} \\
\SI{.23e7}{\candela} \\
\SI[per-mode = symbol]
  {1.99}{\per\kilogram} \\
\SI[per-mode = fraction]
  {1,345}{\coulomb\per\mole}
```

Arara — TEX automation made easy

Paulo Roberto Massa Cereda

Abstract

This article covers a bit of history behind arara, the cool TEX automation tool, from the earlier stages of development to the new 4.0 series. We also highlight some noteworthy features of our tool.

1 Introduction

Writing software is easy. Writing *good* software is extremely difficult. I was working on a Catholic songbook with 1200+ songs and several indices and cross-references. The compilation steps required to achieve the final result were getting out of hand.

At some point, I realized I *knew* all the steps I had to reproduce beforehand, I only had to find a way to automate them! Inspired by the way compilers work (i.e., read a source file, ignore all comments and process the rest), I could exploit TEX comments to include special indications on what to do on the document. Since engines do ignore comments, no side effects would arise, at least document-wise.

It was a cold afternoon. I sat in front of my computer and decided to work on this new tool. It was a matter of time to reach preliminary yet promising results. I mentioned this effort in the chat room of the TEX community at StackExchange and some friends asked me to make a public release out of it, as other users could benefit from this new tool.

However, a name was needed for the tool. In the chat room, we used to have a lot of fun with palindromes (especially palindromic reputations in arbitrary bases), so I took that aspect as inspiration. Then I thought of a very beautiful, colourful bird of the Brazilian fauna: the macaw, or as we like to call it, the arara. The name was immediately adopted!

Once the name was chosen, I needed a logo. Since I am a Fedora Linux user, I was always a fan of their default typeface, which is quite round! The choice was made: the humble arara tool became arara! (But we'll use the more subdued arara in regular text.) My life was about to change.

2 A bit of history

A lot of things have happened since version 1.0, released in 2012, to the new version 4.0, released in 2018. This section presents a bit of history of arara, including challenges in each version.

2.1 The first version

There is a famous quote along the lines of "If at first you do not succeed, call it version 1.0." The first version of arara was also the first public release, dated April 2012. Nothing much was there, besides the core concepts that still exist today:

- *Rules:* a rule is a formal description of how arara handles a certain task. It tells the tool how to do something.
- *Directives:* A directive is a special comment inserted in the source file in which you indicate how arara should behave.

Back then, we could write directives in our document and have the tool process them as expected, like the following example:

```
% arara: pdftex
Hello world!
\bye
```

Amusingly, the first version offered only a log output as an additional feature. There was no verbose mode. The log file was a gathering of streams (error and output) from the sequence of commands specified through directives. And that was it.

2.2 The second version

The first version had a serious drawback: compilation feedback was not in real time and, consequently, no user input was allowed. For the second version, real time feedback was introduced when the tool was executed in verbose mode.

```
$ arara -v mydoc.tex
... [real time feedback] ...
```

Two other features were included in this version: a flag to set an overall execution timeout, in milliseconds, as a means to prevent a potentially infinite execution, and a special variable in the rule context for handling cross-platform operations.

2.3 The third version

So far, arara was only a tiny project with a very restricted user base. However, for version 3.0, a qualitative goal was reached: the tool became international, with localised messages in English, Brazilian Portuguese, German, Italian, Spanish, French, Turkish and Russian. Further, new features such as configuration file support and rule methods brought arara to new heights. As a direct consequence, the lines of code virtually doubled from previous releases.

```
$ arara --help -L es
...
 -h,--help     imprime el mensaje de ayuda
 -l,--log      genera el registro de la salida
...
```

When the counter stopped at version 3.0, Brent Longborough, Marco Daniel and I decided it was time for arara to graduate and finally be released in TEX Live. Then things really changed in my life. The

tool was a success! Given the worldwide coverage of that TEX distribution, arara silently became part of the daily typographic tool belt of many users. But then, the inevitable happened: a lot of bugs emerged from the dark depths of my humble code.

2.4 Critical and blocker bugs

Suddenly, several questions about arara were posted in the TEX community at StackExchange and I was not able to provide a consistent, definitive answer for many of them! It was very tricky to track the bugs to their sources, and some of them were really nasty. For instance, a simple scenario of a file with spaces in the name was more than enough to make the poor tool cry for help for apparently no reason:

```
$ arara "My PhD thesis.tex"
```

Likewise, the issues page of the project repository hosted at GitHub had a plethora of reports, and little could I do about them. I delved into the code of third party libraries, but the root of all evil seemed to lie in my own sources.

2.5 Nightingale

In all seriousness, I was about to give up. My code was not awful, but there were a couple of critical and blocking bugs. Something very drastic had to be done in order to put arara back on track. Then, proceeding on faith, I decided to rewrite the tool entirely from scratch. In order to achieve this goal, I created a sandbox and started working on the new code. And this new project got a proper name: nightingale.

It was the right thing to do. Nicola Talbot helped me with the new version, writing code, fixing bugs and suggesting new features. She was writing a book about LATEX for administrative work at the time and was extensively using arara in the code examples. Her writing indirectly became my writing as well, as I progressively improved the code and added new features to match her suggestions.

2.6 The fourth version

At some point, nightingale had to say farewell and gave most of its features to the bigger, older bird in the nest. It is worth mentioning that nightingale still lives in my repository at GitHub for those who are bold enough to try it. From 1500+ lines of code in version 3.0, arara 4.0 tripled that number: a whopping 4500+ lines of code! And, most important: all critical and blocking bugs were completely fixed.

However, although the code was ready for production, the user manual was far from being finished. In fact, the documentation had to be written entirely from scratch. Then another saga started: find proper time and effort to document a great yet complex tool in all details, from user to developer perspectives.

It took me a lot of dedication to write the user manual and try to cover as much detail as possible for every feature, old and new, and the tool itself. Some of the internals had to be changed, so more explanations were needed. Documenting a tool is almost as difficult as writing code for it!

3 New features

This section highlights some noteworthy features found in the new version 4.0 of arara. For additional information, please refer to our user manual.

3.1 REPL work flow

In version 4.0, arara employs a REPL (read-evaluate-print loop) work flow for rules and directives. In previous versions, directives were extracted, their corresponding rules were analyzed, commands were built and added to a queue before any proper execution or evaluation. I decided to change this work flow, so now arara evaluates each rule on demand, i.e., there is no *a priori* checking. A rule will always reflect the current state, including potential side effects from previously executed rules.

3.2 Multiline directives

Sometimes, directives can span several source lines, particularly those with several parameters. From arara 4.0 on, we can split a directive into multiple lines by using the `arara: -->` mark on each line which should comprise the directive. We call it a multiline directive. Let us see an example:

```
% arara: pdflatex: {
% arara: --> shell: yes,
% arara: --> synctex: yes
% arara: --> }
```

It is important to observe that there is no need for them to be on contiguous lines in the source file, i.e., provided that the syntax for parameterized directives holds for the line composition, lines can be distributed all over the code. The log file (when enabled) will contain a list of all line numbers that made up a directive.

3.3 Directive conditionals

arara 4.0 provides logical expressions, written in the MVEL language, and special operators processed at runtime in order to determine whether and how a directive should be processed. This feature is named *directive conditional*, or simply *conditional* for short. The following list describes all conditional operators available in the directive context.

- `if`: The associated MVEL expression is evaluated beforehand, and the directive is interpreted if, and only if, the result of such evaluation is true. This directive, when the conditional holds true, is executed at most once.

```
% arara: pdflatex if missing('pdf')
% arara: --> || changed('tex')
```

- `unless`: Same as `if` but the condition test is inverted.

```
% arara: pdflatex unless unchanged('tex')
% arara: --> && exists('pdf')
```

- `until`: The directive is interpreted the first time, then the associated MVEL expression evaluation is done. As long as the result holds false, the directive is reinterpreted. There is no guarantee of halting.

```
% arara: pdflatex until !found('log',
% arara: --> 'undefined references')
```

- `while`: Same as `until` but the condition test is inverted.

```
% arara: pdflatex while missing('pdf')
% arara: --> || found('log', 'undefined
% arara: --> references')
```

Although there is no conceptual guarantee for proper halting of unbounded loops, we have provided a practical solution to potentially infinite iterations: arara has a predefined maximum number of loops. The default value is 10, but it can be overridden either in the configuration file or on the command line.

3.4 Directive extraction only in the header

The `--header` command line option changes the mechanics of how arara extracts the directives from the code. The tool always reads the entire file and extracts every single directive found throughout the code. However, by activating this switch, arara will extract all directives from the beginning of the file until it reaches a line that is not empty and is not a comment (hence the option name). Consider the following example:

```
% arara: pdftex
Hello world.
\bye
% arara: pdftex
```

When running arara without the `--header` option, two directives will be extracted (on lines 1 and 4). However, if executed with this switch, the tool will only extract one directive (from line 1), as it will stop the extraction process as soon as it reaches line 2.

3.5 Dry-run execution

The `--dry-run` command line option makes arara go through all the motions of running tasks and subtasks, but with no actual calls. This is useful for testing the sequence of underlying system commands to be performed on a file.

```
[DR] (PDFLaTeX) PDFLaTeX engine
---------------------------------------
Authors: Marco Daniel, Paulo Cereda
About to run: [ pdflatex, hello.tex ]
```

Note that by the rule, authors are displayed (so they can be blamed in case anything goes wrong), as well as the system command to be executed. It is an interesting approach to see everything that will happen to your document and in which order. It is important to observe, though, that conditionals are not evaluated in this mode.

3.6 Local configuration files

From version 4.0 on, arara provides support for local configuration files. In this approach, a configuration file can be located in the working directory associated with the current execution. This directory can also be interpreted as the one relative to the processed file. This approach offers a project-based solution for complex work flows, e.g., a thesis or a book. However, arara must be executed within the working directory, or the local configuration file lookup will fail. Observe that this approach has the highest lookup priority, which means that it will always supersede a global configuration.

3.7 File hashing

arara 4.0 features four methods for file hashing in the rule and directive scopes, presented as follows. The file base name refers to the file name without the associated extension.

- `changed(extension)`: checks if the file base name concatenated with the provided extension has changed its checksum from last verification.
- `changed(file)`: the very same idea as the previous method, but with a proper Java `File` object instead.
- `unchanged(extension)`: checks if the file base name concatenated with the provided extension is unchanged from last verification. It is the opposite of the `changed(...)` method.
- `unchanged(file)`: the very same idea as the previous method, but with a proper Java `File` object instead.

The value is stored in a database file named `arara.xml` as a pair containing the full path of the provided file and its corresponding CRC-32 hash (the

database is created as needed). If the entry already exists, the value is updated, or created otherwise.

3.8 Dialog boxes

A *dialog box* is a graphical control element, typically a small window, that communicates information to the user and prompts them for a response. arara 4.0 provides UI methods related to such interactions. As good practice, make sure to provide descriptive messages to be placed in dialog boxes in order to ease and enhance the user experience.

3.9 Session

Rules are designed under the *encapsulation* notion, such that direct access to the internal workings of such structures is restricted. However, as a means of supporting framework awareness, arara provides a mechanism for data sharing across rule contexts, implemented as a `Session` object. In practical terms, this particular object is a global, persistent map composed of keys and values available throughout the entire execution.

3.10 Redesigned user interface

For arara 4.0, we redesigned the interface in order to look more pleasant to the eye; after all, we work with TEX and friends. Please note that the output here is truncated to respect the column width.

```
  __ _ _ __ __ _ _ __ __ _
 / _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
 \__,_|_|  \__,_|_|  \__,_|

Processing 'doc.tex' (size: 307 bytes, last
modified: 05/29/2018 08:57:30), please wait.

(PDFLaTeX) PDFLaTeX engine ..... SUCCESS
(PDFLaTeX) PDFLaTeX engine ..... SUCCESS

Total: 1.45 seconds
```

First of all, we have the nice application logo, displayed using ASCII art. The entire layout is based on monospaced font spacing, usually used in terminal prompts. Hopefully you follow the conventional use of a monospaced font in your terminal, otherwise the visual effect will not be so pleasant. First and foremost, arara displays details about the file being processed, including size and modification status:

```
Processing 'doc.tex' (size: 307 bytes, last
modified: 05/29/2018 08:57:30), please wait.
```

The list of tasks was also redesigned to be fully justified, and each entry displays both task and subtask names (the former being displayed enclosed in parentheses), besides the usual execution result:

```
(PDFLaTeX) PDFLaTeX engine ..... SUCCESS
(PDFLaTeX) PDFLaTeX engine ..... SUCCESS
```

If a task fails, arara will halt the entire execution at once and immediately report back to the user. This is an example of what a failed task looks like:

```
(PDFLaTeX) PDFLaTeX engine ..... FAILURE
```

Also, observe that our tool displays the execution time before terminating, in seconds. The execution time has a very simple precision, as it is meant to be easily readable, and should not be considered for command profiling.

```
Total: 1.45 seconds
```

The tool has two execution modes: *silent*, which is the default, and *verbose*, which prints as much information about tasks as possible:

- When in silent mode, arara will simply display the task and subtask names, as well as the execution result. Nothing more is added to the output.
- When executed in verbose mode, arara will display the underlying system command output as well, when applied. In version 4.0 of our tool, this mode was also entirely redesigned in order to avoid unnecessary clutter, so it would be easier to spot each task.

It is important to observe that, in verbose mode, arara can offer proper interaction if the system command requires user intervention. However, in silent mode the tool will simply discard this requirement and the command will almost surely fail.

4 The future

Now that arara 4.0 is officially released and already available in CTAN and TEX Live, it is time to plan the future. Our repository already has suggestions for new features and improvements. The work on arara 5.0 has begun! If you have any feedback about our tool, please drop us a note.

Also, if you believe your custom rule is comprehensive enough and deserves to be in the official pack, please contact us. We will be more than happy to discuss the inclusion of your rule in forthcoming updates. Happy TEXing with arara!

◇ Paulo Roberto Massa Cereda
Analândia, São Paulo, Brazil
cereda dot paulo (at) gmail dot com
github.com/cereda/arara

The Canvas learning management system and LaTeXML

Will Robertson

1 Introduction

In 2017 The University of Adelaide adopted *Canvas* (Instructure, Utah, USA) for its learning management system (LMS). Unlike our previous LMS, Canvas provides a programming interface to get data from and send data to its servers. In this paper I will discuss the system I have started developing to prepare coursework material using LaTeX, processed via LaTeXML, and uploaded into Canvas in both HTML and PDF formats.

1.1 What is a learning management system?

An LMS is an online system that organises students into classes and allows them to access course notes, assignments, lecture recordings, discussion boards, and so on. They usually also include collaborative features to help teamwork activities. The LMS is also how the lecturer or coordinator for the course communicates with the students, posts grades, and otherwise does their job of running the course.

Each course tends to have a fairly typical structure, with links to the syllabus, assignment submissions, grades, etc. Actual course content in Canvas is organised into modules with individual pages written in HTML via a rich text editor. The amount of technical content in each course will depend on the coordinator; most academics I know still rely largely on printed lecture notes.

1.2 Requirements for the project

As the coordinator for the mechanical engineering honours project course, I maintain a large collection of course content that is provided as reference material. After many years as a LaTeX user, I am most comfortable with the idea of a source document that is portable, version controllable, and easy to edit. For me, using any web interface to fix typos or make small changes requires a non-negligible mental effort due to having to log in, navigate to an appropriate page, and click through to an editing mode. This project was motivated by my desire for a more efficient and effective means to keep course material up-to-date.

In the past, our honours project course documentation was a single monolithic Word document, which was difficult to maintain and awkward to extract information from. With a slow move to online documentation and incremental changes over time, this comprehensive document fell out of sync with the live content and had to be dropped. However, the single reference document had specific utility in being able to be easily distributed, and being searchable; its loss was not disastrous but not ideal.

The new LMS provided the opportunity to rethink the means by which the information needed in this course was documented and communicated to students and supervisors. This project was motivated by two main requirements: (1) improving the care and maintenance of the (somewhat extensive, and growing, amount of) content, and (2) to allow the generation of a comprehensive PDF reference document for the entire course.

When I started this work, I didn't know exactly what I wanted except that I knew I needed at least a good combination of the following:

One source, multiple outputs. My need for a better system arose from wanting to be able to keep online content up-to-date, while also having this content collected in a single offline location. And it soon became clear in my thinking that it would also need to be compilable into a single PDF document for distribution via alternate means.

Macros. I knew there would be a fair amount of information that I would want to re-use and keep consistent through the document (e.g., the names of academic and professional staff members; due dates for assessment). This precluded writing in HTML or a 'plain text' format like Markdown.

Reliable and easy to set up. As much as I might like to program my own document preparation system, using this was to be part of my day job and I don't want to have to dive into the weeds if code rot occurs and the system breaks down.

LaTeX syntax. I admit, I stick with what I know. Although I could have jumped into any number of competing technologies here, I knew I would be most comfortable if I was writing in the system with which I was most accustomed. More objectively, LaTeX is a mature format with a variety of possibilities among third party support tools for creating HTML.

2 The authoring interface

After some quick trials to establish that I could programmatically send HTML content to Canvas, it was time to decide how to generate the HTML in the first place.

2.1 Which HTML converter to choose?

As discussed above, I didn't want to do anything home-grown (and hence fragile), nor inflexible such as Markdown or raw HTML. I was aware of a number of

'competing' technologies to write in LaTeX or LaTeX-like syntaxes which could be converted to XML and hence to HTML. The most actively developed of these tools, and their implementation language, appear to be:

- lwarp [1] — Lua
- LaTeXML [2, 3] — Perl
- HeVeA [8] — OCaml
- Tralics [4] — C++
- TeX4ht [5] — C and TeX
- GELLMU [6, 7] — Emacs Lisp

(Not an exhaustive list; apologies for any oversights.) To be honest I didn't thoroughly evaluate each on their pragmatic merits; I was passingly familiar with LaTeXML's philosophy, had heard it was robust, and wanted to see if it fit the bill. It did, largely speaking. Of the tools listed above, only TeX4ht and lwarp are included in TeX Live. While LaTeXML required 'manual' installation, I had no troubles doing so.

2.2 LaTeXML overview

The LaTeXML program would be better explained by someone who knows a lot more than I do about Perl, XML, and friends. My user-level understanding is that LaTeXML reimplements an extensive subset of TeX in Perl, so that input documents are literally processed as LaTeX syntax, but not by the LaTeX program. The LaTeXML parser then intercepts package loading and inserts its own understanding of the various syntaxes introduced by different packages. If needed, it is possible to write custom support for packages that it doesn't cover out of the box.

LaTeXML does an excellent job emulating TeX, and it covers an impressive array of both TeX and LaTeX programming constructs.[1] Therefore, including in my preamble a construct like

```
\newcommand\honourscoord{Will Robertson}
```

simply worked out of the box with LaTeXML; indeed, simple LaTeX 2ε programming using counters and so on worked without a hitch.

To run LaTeXML on my document involves the somewhat complex command:

```
latexml tex/$FILENAME.tex | latexmlpost - \
  --xsltparameter=SIMPLIFY_HTML:true \
  --sourcedirectory=tex \
  --format=html5 \
  --destination=html/$FILENAME.html \
  --splitat=chapter \
  --splitnaming=label
```

This setup ensures that for each 'chapter' of my LaTeX document a separate self-contained HTML file is created, which is the starting point for getting my content into Canvas.

Using a degree of consistency in the naming and structure of my document, each of my source LaTeX files with an `\input` for each chapter is therefore converted into a similarly-named HTML file. The structure of the source document is as follows:

```
\documentclass{report}
  ...
\begin{document}
  \title{...}\author{...}\date{...}
  \maketitle\tableofcontents
  \input{../texdata/course-data.tex}
\part{Introduction}\label{part-intro}
  \input{../pages/introduction}
  \input{../pages/course-schedule}
  \input{../pages/week-planner}
  ...
```

The file `course-data.tex` is the source for various data (such as names of staff members, weightings of assessment, due dates, etc.) in basic LaTeX data structures.

Each `.tex` file contains one chapter, with a convention that the `\label` of each chapter matches its file name:

```
% file: pages/introduction.tex
\chapter{Introduction}
\label{introduction}
...
```

This is because LaTeXML does not convert file by file; rather, with the `--splitat=chapter` option, the output HTML is split by chapter.

3 The Canvas programming interface

Canvas has a so-called 'REST API'[2] that was easy enough for me to get started with, with a little trial and error. Initially, I used `curl` commands wrapped into Bash scripts like this:

```
curl -X GET -H "$CANVASAUTH" $CANVASCOURSE/$1
```

where `$CANVASAUTH` is set up in my `.bash_profile` as a secret 'token' to avoid needing a manually-input password, and `$CANVASCOURSE` is defined essentially as the institution-specific URL to the Canvas course. Finally, `$1` is the parameter to send through to the Canvas API, such as `assignments` or `rubrics` or `users`, etc. The parameters passed in the `curl` argument `$1` can include additional options, such as

```
assignments?search_term=charter
```

would return just the one assignment for my honours project students called their 'Project Charter'.

The Canvas API can also be used to send or upload data to a course using `PUT` and `POST`. This

[1] Even David Carlisle's `xii.tex` can be successfully run through LaTeXML.

[2] `https://canvas.instructure.com/doc/api/`

can range from relatively small and simple pieces of information, to entire 'content pages' in HTML, to linked files that students can access and download (see Section A).

The results from any communication are sent back from the Canvas API in JSON format, for which there are a number of useful standard tools. For Bash scripts I use `jq`, a nice tool which is designed in the philosophy of Unix tools such as `awk` and `sed`.

After getting comfortable with this Bash script approach of sending and receiving data using `curl` and friends, I more recently started programming small Lua interfaces for more advanced operations. The main impetus for this switch was that the Canvas API will not send arbitrarily large amounts of information in one request. For example, if I request a list of students, it will send me just ten and expect me to ask again with `page=2` as an option. Then repeat until no data is returned. This level of iteration was beyond what I wanted to invest in a Bash script.

To write the Lua scripts I needed a number of third-party utilities. For Lua there are in fact very many tools to convert JSON data into Lua's 'table' data structure. I have been using the library json-lua (installed via luarocks) and it has been fine. After installing an SSL library (namely luasec, since unencrypted HTTP wouldn't cut it), I was able to convert the `curl` command above into an equivalent statement in Lua:

```
local http  = require("ssl.https")
local ltn12 = require("ltn12")
local body, code, hdrs, status = http.request{
  method = "GET",
  url = canvas_url .. req .. "?" .. opt,
  headers = {
   ["authorization"] = "Bearer "..canvas_token,
   ["content-type"]  = "application/json"
  },
  sink = ltn12.sink.table(canvas_result),
}
```

where `canvas_result` is the name of the table where the data will be stored. It is then 'decoded' from JSON using json-lua.

4 The generated HTML page

For pages delivered via Canvas, I am not writing self-contained HTML files; rather, Canvas constructs the page within its main interface, and within the page includes what I call 'snippets' of HTML to display the actual course content.

The HTML that is generated by LaTeXML is not intended to be used for 'snippets' to be transferred into a separate system, but the good thing about HTML is that this isn't really a problem, since any additional markup in the HTML is silently ignored.

LaTeXML runs a two-stage process, where the latexml program itself generates generic XML from the LaTeX input, and then latexmlpost converts this generic XML into one of several output formats. In theory I could write my own XSL stylesheet to format the information in the generic XML document in a customised way. Instead, I simply postprocess the HTML output from latexmlpost; because LaTeXML creates highly structured and machine friendly HTML, the output from latexmlpost makes this an easy process with some ad hoc tools (with plans for more robust Lua processing in the future).

According to the options passed to latexmlpost, each chapter is converted into its own HTML file; this means I have a one-for-one correspondence between files `\included` as chapters in the main document. Each of these chapters has a standard structure, with 'top matter' that is not needed:[3]

```
<!DOCTYPE html><html>
<head>
 <title>
  ⟨Title of chapter⟩
 </title>
 <meta http-equiv="Content-Type"
       content="text/html; charset=UTF-8">
 <link rel="stylesheet" href="LaTeXML.css"
       type="text/css">
  [...]
</head>
<body>
```

The chapters end with 'bottom matter' which is also not needed in my application:

```
<footer class="ltx_page_footer">
  [...]
</footer>
</div>
</body>
</html>
```

Finally, the 'content' of each HTML file has a structure along the following lines:

```
<div class="ltx_page_main">
<header class="ltx_page_header">
  [...]
</header>
<div class="ltx_page_content">
 <section class="ltx_chapter ltx_authors_1line">
  <h1 class="ltx_title ltx_title_chapter">
   <span class="ltx_tag ltx_tag_chapter">
    ⟨Chapter number⟩
   </span>
   ⟨Title of chapter again⟩
```

[3] All of the HTML examples have been reformatted for ease of reading.

```
  </h1>
  <div class="ltx_date ltx_role_creation"></div>
  <section id="S1" class="ltx_section">
   ⟨Contents of chapter⟩
  </section>
 </section>
</div>
```

The un-greyed text is the part of the HTML that is retained for upload into Canvas. A single line of `awk` is used to extract everything contained between the outer `<section>` tags:

```
awk '/\<section.*\>/,/\<\/section\>/' \
  html/$BASE >> snip/$BASE
```

The LMS doesn't do anything in particular with the `<section>` tags, but it doesn't mind them, either. Similarly, all of the CSS structure (the `class` and `id` tags) is not used by Canvas, but also, thankfully, doesn't cause any problems.

It is worth noting that while plain LaTeX and HTML share some superficial similarities in the types of document structures they can produce, in some cases LaTeXML really has to work hard to replicate certain LaTeX structures in HTML+CSS. The example that I ran into was related to lists. In order to cope with LaTeX syntax such as:

```
\begin{enumerate}
  \item      aaa
  \item[1b.] bbb
  \item      ccc
\end{enumerate}
```

the HTML generated by latexmlpost looks like:

```
<ol id="I1" class="ltx_enumerate">
 <li id="I1.i1" class="ltx_item"
                style="list-style-type:none;">
  <span class="ltx_tag ltx_tag_enumerate">
   1.
  </span>
  <div id="I1.i1.p1" class="ltx_para">
   <p class="ltx_p">aaa</p>
  </div>
 </li>
 <li ...>...</li>
 <li ...>...</li>
</ol>
```

(The second and third items have identical structure and are elided.) This is carefully structured HTML source intended to be styled with specific CSS provided by LaTeXML. However, without latexml's custom CSS files to control its layout, this HTML produces output something like this:

1. aaa

1b. bbb

2. ccc

While understandable to a reader, this is not ideal from a formatting perspective. LaTeXML's default to match as much of LaTeX's functionality as possible is notable in the overall design of LaTeXML, but for my needs it still needed a workaround. In an early stage of this project, I used more cumbersome regex code to adapt the LaTeXML output to something that didn't need CSS, but the developer of LaTeXML kindly added the `--xsltparameter=SIMPLIFY_HTML:true` option to account for my use case here.

5 Future work

- I do not yet have an automated approach to linking images and files. This has been okay for the time being, since for a small number of items doing a manual upload is no real imposition. In the long run, it would be nice to have this automated; compiling the main document could create a list of files, and a Lua script could check which files were already present in Canvas and only upload those missing. (And possibly even delete any existing files no longer used in the document.)
- What about mathematics? Luckily, for this course I don't need to include mathematical content. LaTeXML, naturally, can produce appropriate mathematical output in a variety of modes (MathML, etc.). Currently Canvas is in the middle of having its support for mathematical content improved and I'm holding off considering this further until their platform has stabilised.
- In time I will transition away from Bash scripts entirely to make the system more portable and robust. Since a TeX platform is required to typeset the PDF documentation, Lua is the natural choice as a scripting language. I explicitly do not wish to develop a full-featured Canvas interface in Lua, but I hope this system will become general enough that other users could deploy it for their courses.
- Currently these documents use a largely 'one-way' communication in that the LaTeX source is compiled and delivered to Canvas. However, for certain types of information (assignment rubrics in particular), the best source of this information is within the LMS itself. Therefore, I will be

building data processors to typeset information from Canvas within the PDF documentation (with a simple link in the online version).

6 Benefits of scripting Canvas

This is unrelated to the LaTeX side of things, but coming to terms with Canvas's programming interface has opened the door for me to perform quite a number of additional tasks that were previously impossible with our older LMS.

For example, our honours project reports are assessed by their academic supervisors, and directing supervisors to each report and following up in a timely manner were both difficult tasks to automate. Using a Lua script I now dynamically create a list of project reports that have not yet been marked. This allows me to automatically construct personalised emails for each project supervisor to remind them when marks are due with direct links to their assessments to mark.

As with LaTeX itself, once you have an interface that can be programmed it opens the door to extending the ways in which one uses the system.

7 Conclusion

I have satisfied the following use cases in this work:

- A typo fix or quick addition can be done by editing the source in a text editor, synced via the cloud, and uploaded with a one-line command. No need to open a browser, log in, and click through.
- Information can be 'programmed' using macros for greater consistency. This is straightforward in LaTeX and basically unheard of in a web-based editing approach.
- HTML and PDF output are kept in sync at all times; the PDF provides an archivable document for the entire course.

Although I haven't (yet) produced the most elegant system, I have created a solution without too much elbow grease beyond standard LaTeX ecosystem tools that does quite a bit more than I think anyone would otherwise consider possible. The choice of LaTeXML worked well, although few design decisions rely on it; switching to another tool for the HTML conversion would be possible with a little additional work.

The ability to develop these solutions is truly a testament to the flexibility of LaTeX, and an example of why I think it will remain relevant indefinitely. Once you program your first document, you can never go back to manually keeping track of all the bits and pieces.

References

[1] B. Dunn. Producing HTML directly from LaTeX — the lwarp package. *TUGboat* 38(1), 2017. tug.org/TUGboat/tb38-1/tb118dunn-lwarp.pdf

[2] D. Ginev and B. R. Miller. LaTeXML 2012 — A Year of LaTeXML, 2014. nist.gov/publications/latexml-2012-year-latexml

[3] D. Ginev, B. R. Miller, and S. Oprea. E-books and Graphics with LaTeXML, 2014. arxiv.org/pdf/1404.6547v1

[4] J. Grimm. Tralics, a LaTeX to XML translator. *TUGboat* 24(3), 2003. tug.org/TUGboat/tb24-3/grimm.pdf

[5] E. M. Gurari. TeX4ht: HTML production. *TUGboat* 25(1), 2004. tug.org/TUGboat/tb25-1/gurari.pdf

[6] W. F. Hammond. GELLMU: A bridge for authors from LaTeX to XML. *TUGboat* 22(3), 2001. tug.org/TUGboat/tb22-3/tb72hammond.pdf

[7] W. F. Hammond. Dual presentation with math from one source using GELLMU. *TUGboat* 28(3), 2007. tug.org/TUGboat/tb28-3/tb90hammond.pdf

[8] HeVeA. hevea.inria.fr

A Uploading a file to Canvas via its API

Evidently in a fit of late night fervour I concocted the following monstrosity for uploading a file to Canvas using a Bash function and few helper commands:

```
curl -X POST  -H "$CANVASAUTH" \
                  "$CANVASCOURSE/files" \
     -F "name=$1" \
     -F "parent_folder_path=upload" > tmp.json ;
URL=‘cat tmp.json | jq ’.upload_url’‘ ;
KEYS=‘cat tmp.json | jq ’.upload_params’ |\
      jq -r -j "to_entries | \
        map(\"-F \(.key)=\(.value|tostring)\
            \")|.[]"‘ ;
echo curl -D response.tmp\
                $URL $KEYS -F file=@$1 | bash ;
LOC=‘sed -n -e ’s/Location: \
                     \(.*\)/\1/p’ response.tmp‘;
LOC=${LOC%$’\r’}
curl -X POST -H "$CANVASAUTH" "$LOC" | jq ;
```

I think it's fair to say that a Lua implementation would be rather more maintainable.

◇ Will Robertson
School of Mechanical Engineering
The University of Adelaide, SA
Australia
will.robertson (at) adelaide dot edu dot au
https://gitlab.adelaide.edu.au/wspr/canvas-tools

Implementing PDF standards for mathematical publishing

Ross Moore

The author (hereafter, simply 'Ross') asserts the desirability[1] of having mathematical documents — journal articles, research reports, monographs, courseware, etc. — be produced conforming to modern PDF standards; in particular, validating for PDF/UA (Universal Accessibility) and PDF/A-2a (or PDF/A-3a) for both Archivability and Accessibility. These are published standards, respectively as ISO 14289-1:2012 (slight revision in 2014) [13], ISO 19005-2:2011 [8] and ISO 19005-3:2012 [9], all based on ISO 32000-1 (PDF 1.7) [4]. Ross has demonstrated the feasibility of using LaTeX to build documents that conform to these standards and can provide example documents [21].

With the cooperation of most academic publishers, Ross asserts that this can be achieved within five years, along with just a little education of authors to provide the minimal extra information required in producing such documents from LaTeX source. This involves use of LaTeX coding supporting 'Tagged PDF', mostly already written for much of 'standard' LaTeX, many commonly-used packages, and extendable to the document classes required by academic publishers that accept LaTeX source from authors.

Here is a summary of how the envisioned timeline of 5 years would be achieved, outlining the main tasks to be undertaken both by publishers and by Ross himself.

Year 1 *Publishers*: Implement full support for PDF/A-2u (or PDF/A-3u) — which doesn't require 'Tagged PDF'.

TWG:[2] extend support for 'Tagged PDF' to cover all features of the class files used by publishers, and to more LaTeX packages used by authors along with such classes.

Year 2 *Publishers*: provide free access to PDFs of example articles (from back issues), produced by Ross during year 1, to visually-disabled academics and researchers, for feedback on the quality of the Accessibility features; technical editors learn the extra requirements in the production of 'Tagged PDF' documents, compliant with both PDF/UA and PDF/A-2a (or PDF/A-3a). Also seek feedback from libraries on the switch to using PDF/A.

TWG: provide instruction to technical (and other) editors on the extra requirements; continue to process more examples from back issues, solving issues that may arise in supporting special kinds of content.

Year 3 *Publishers*: start to produce 'Tagged PDF' versions of current issues in a small number of journals; educate all editors in the extra requirements for producing 'Tagged PDF'.

Ross: implement any extra features, arising from the user feedback; work in support of production staff to ensure the 'Tagged PDF' articles are produced smoothly; start to develop instructional materials, suitable for both authors and editors.

Year 4 *Publishers*: extend use of 'Tagged PDF' to more journals; continue to receive feedback, passing on recommendations to Ross for implementation; fully develop instructional materials for authors.

TWG: continue to work on the LaTeX coding, to produce a stable, modularised system that editors and (later) authors will be able to use; extensions to other document classes, such as books and monographs, etc.

Year 5 *Publishers and Ross*: make the new LaTeX packages or class files publicly available (e.g., via the CTAN network).

Publishers: make available instructional materials for these new resources.

TWG: continue to act on feedback from authors and editors, expecting to produce regular updates to the software, as required.

1 Details

We now describe some of the details involved with the tasks listed above.

1.1 Archivability

To achieve archivability, the main issues for PDFs created with LaTeX, in the standards ISO 19005-2:2011 [8] and ISO 19005-3:2012 [9], are the following:

(a) inclusion of metadata in the XML-based XMP format [3];

(b) specification of a color profile (usually 'RGB' or 'CMYK') for the document, and ensuring that all embedded images and any color-changing LaTeX commands used in the production of the PDF conform to the color-space described by the profile;

[1] Libraries at academic institutions, particularly in Germany, are requiring academic theses to be submitted conforming to a PDF/A standard. Governments in several countries have 'accessibility' requirements for electronic publications. Notable here is the U.S. GSA Government-wide Section 508 Accessibility Program [2].

[2] 'TWG' here denotes a TeX working group, consisting of developer/programmers from existing TUG working groups, initially with Ross as the lead programmer. This will include members of the LaTeX3 team.

(c) characters in all fonts used within the document have a mapping into the Unicode collection of character codes;

(d) interword spaces are present in the PDF output.

These are all requirements for all flavours of level and conformance for PDF/A. Thus to build up conformance with PDF/A-2a (the 'accessible' flavour), one can start first with PDF/A-2b or PDF/A-2u which do not require the extra complications needed for accessibility. Put simply, deal with the other issues first; then add the 'accessibility' part later. Furthermore, none of these affects the actual typesetting, so a PDF/A document can replace a non-PDF/A one at any stage in the production process, in particular to become a preferred online format. PDF/A-3 is appropriate when the final PDF is to contain attachments in formats other than PDF; e.g., movies or runnable code for mathematics engines.

The LaTeX package pdfx [22] has coding to deal with all of the issues (a)–(d) listed above, when used with pdfTeX as the processing engine. Ross is currently the principal developer for the pdfx package, so is familiar with its LaTeX coding, and how to extend it to cope with any issues that may arise.

1.2 Metadata

In a production environment, metadata will normally be stored in a database, separate from the LaTeX source but clearly related to it. Using pdfx, a file (with suffix .xmpdata) is used with the LaTeX job. This can be generated as a database report. Publishers will need to develop appropriate work flows.

The XMP metadata [3] is organised as an XML formatted file, which is included uncompressed into a PDF as a single object. Using pdfx this XML file is constructed using a template, with TeX macros expanding to provide the values for specific Metadata fields. Values are passed to these macros using the .xmpdata file.

This setup is easily expanded to include extra fields, using the concept of 'PDF/A Extension Schema'. The standard templates which come with pdfx have examples of this, for including Metadata fields from PRISM specifications [23]. It is not hard to add new fields, and define extra TeX macros within the coding for pdfx.sty. If there is a need for this, in particular for information that ultimately needs to be supplied by authors, then Ross can extend the pdfx package to cope.

1.3 Color profiles

For PDF files distributed via the Internet, the appropriate color profile will normally be based on the 'RGB' color space. Consider embedded images, such as photographs, scientific graphics or other kinds of colored diagram. These will all need to specify their colors via the RGB space — but this may not be the space used when a graphic image supplied by the author was created. There will be a need for conversions to be made.

Publishers already have to deal with such issues, but with strict conformance to a PDF/A standard, the amount of work required in this area can be expected to increase. It could be useful to setup an online 'color-conversion' service. This would allow authors to convert images prior to submission of a paper for publication. The resulting image could then be used by the author during preparation of their manuscript, and also be already available on the publisher's system, converted into the required format (perhaps with extra metadata added).

Some publishers may also want a 'CMYK' version of images for paper-printing, in color. This might be done via a separate LaTeX run, with the .xmpdata file specifying a CMYK profile, which need not be the one included with the pdfx package distribution [22]. Note that pdfx loads the xcolor package, with options corresponding to the chosen color space ('RGB' or 'CMYK'). This forces all LaTeX color commands to use the specified space, irrespective of how a color was originally specified by the author. Thus there is no need to adjust an author's LaTeX preamble, or other coding, to ensure compliance with a PDF/A standard.

1.4 Font mappings to Unicode

The pdfTeX software already has a feature to generate mappings of font characters to valid Unicode code-points, and the pdfx package enables this feature. However, some rarely-used characters can be mapped into invalid code-points; e.g., incorrectly mapped into the 'Private Use' area, or where there is no obvious corresponding code-point. When this occurs, the `\pdfglyphtounicode` command allows the problem to be overcome, for each troublesome font character. Such instances should be reported to Ross, to insert such command usage into pdfx.sty, to avoid the particular issue recurring.

1.5 Interword spaces

Normally TeX (and hence LaTeX) does not insert space characters into the PDF output that it generates. However, since 2014 (and earlier in an experimental branch) the pdfTeX software has had the ability to insert extra spaces, using a heuristic method to determine when sufficient white space occurs between characters. This happens on output

only, so has no effect whatsoever on the typesetting. That is, the visual page remains unchanged. These spaces do contribute to Copy/Paste and occur within the output stream fed to screen-readers and 'Assistive Technology'.

This feature was added, at Ross' request, specifically to meet the requirement for PDF/A compliance. It is turned on automatically when using the pdfx package.

1.6 Validation

When producing a PDF document that is supposed to be conform to a particular standard, it is important to check this conformance with validator software. For PDF/A there are a number of programs that can do this. Best is probably the 'Preflight' utility that is built-in to the 'Acrobat Pro' application from Adobe Systems Inc. [1]. This can also be obtained as stand-alone software named pdfaPilot [10]. There's another validator at pdftools.com [11].

Production editors in particular, and eventually all editors, will need to gain experience using such validation software. Starting with PDF/A-2u, the errors are likely to be associated with items (a)–(d) above. Ross has had more than 5 years of experience creating documents conforming to PDF/A, in all its flavours. He can help interpret the error reports that may arise using Adobe's Preflight.

Later, as we move towards accessibility, via 'Tagged PDF', which has much stricter requirements on what must be tagged, and how that tagging is actually done — some of the errors will have an obvious cause associated with poorly-constructed LaTeX coding in an author's manuscript. Others may not be so clear. Since TeX was designed well before PDFs were conceived, let alone any tagging, there are no internal checks related to it. Macros can be written in LaTeX to catch some tagging-related errors, but for most errors that are not also errors in TeX, mostly you will not know that there is a problem until the resulting PDF has been checked with a validator. Again Ross' experience is paramount here, to diagnosing the problem and devising appropriate internal LaTeX coding to properly handle the situation.

2 Tagging for accessibility

Producing a 'Tagged PDF' document requires tagging both structure and content. With 'structure tagging' viewed as providing a tree-like description of blocks the information contained within the document, then 'content tagging' supplies the leaf nodes for this tree. That is, a span of content 'hangs off' the structure tree, much as a leaf hangs from a branch. Detailed indexing of all structure types allows for rich navigation facilities; e.g., finding all section headings, list items, all inline or displayed mathematical expressions.

It is this embedded structure and indexing that is the key to 'Accessibility' within a PDF, since it allows the content to be studied in ways that are independent of visual appearance. A primary requirement of PDF/UA, is for *all* content to be 'tagged', either as leaf nodes of the structure tree, or as an 'Artifact' — such as page numbers and running headers and footers. Content such as embedded figures and mathematical formulae must be accompanied by 'alternative text', providing an easily spoken description of what the formula is about.

Precise (and some not-so-precise) rules on how a PDF file needs to be constructed for PDF/UA is detailed in the 'Matterhorn Protocol' [15]. This document contains a collection of 31 checkpoints, comprised of 136 'Failure Conditions', to test whether a given PDF is compliant. Most of these can be automated, and such checking is available with the validation software mentioned above. Furthermore, Adobe's 'Acrobat Pro' software [1] has a suite of 32 checks built in to its 'Accessibility Tool', most of which are the same as in the Matterhorn Protocol; there are differences, but no incompatibilities. Some of the latter tests can only be checked by a human; e.g., sufficient color contrast, correct reading order, hyperlinks point to a sensible (and correct) target, etc. In all, these rules embody all of the WCAG guidelines [24] for documents delivered via the internet, insofar as these can be sensibly applied to a PDF document — most can.

It is desirable for PDFs produced using LaTeX to satisfy *all* the automated checks of both the Matterhorn Protocol [15] and Acrobat Pro's Accessibility Tool [1]. This has been the case with Ross' earlier work [18, 19, 20], and remains so with the 'Tagged PDF' document examples [21] produced more recently. Furthermore, the idea of 'alternative text' can be extended beyond figures and formulæ; indeed, any structure item or span of content can have a speakable alternative. This is useful for the 'Accessible Text' view, as read by screen-readers that do not have the ability to follow structure, but only content. In [20] Ross introduced the notion of 'access-tag', which is a span of content containing a single very, very small space, which can be equipped with arbitrary 'alternative text'. It can hold words such as "start of enumerated list", "end of quotation", etc., as appropriate to the start and end of blocks of special content.

2.1 Tagged PDF with LaTeX

To date, the tagging required for many aspects of a document's structure have been implemented by Ross, within example documents [21]. These include:

> document title, normal paragraphing, section headings (numbered and unnumbered), lists and list-items, font-changes within a paragraph, italic corrections, inline mathematics, most amsmath displayed environments, footnotes, cross-referencing, hyperlinks, table-of-contents, citations, bibliography, floats, theorem-like environments, some verbatim environments, quotations, centering, abstract, included images, ...

The title page of a document depends very much on the particular document class being used. A significant amount of work is needed to correctly tag all the different pieces of information that may occur. Thus a large part of the work required to support a new document class is devoted to just getting the first page(s) correct. All of this has been done in such a way that the pagination and full-page layout are exactly the same as would occur with no tagging; that is, all the internal 'glue' calculations performed by the TeX engine result in the same numerical values for all glue settings.

Other typical document elements that currently are not yet properly supported include:

> tabular environments, two-column format, color-changing commands, language switches, index and indexing, some displayed math-environments, line-numbering, commenting, and most user-supplied packages that are not merely built from environments in the LaTeX base distribution.

Thus there is still plenty of work to be done, some of it easy, but much of it not. The main source of difficulty lies in the ways that different environments can interact with each other. There are many situations in LaTeX where one environment or structure is not complete until the next has begun. So it is not just a matter of wrapping start and finish tags around every piece of supplied content. Instead one needs to understand the subtleties of how different environments and other structures actually start and finish, within the context established by surrounding material.

2.2 Export to other formats

When creating a 'Tagged PDF' document from LaTeX source, the created tags use the name of the LaTeX environment presenting its content. These names are essentially arbitrary, so must be 'mapped' to standard PDF structure types. Similarly for the tagging of content. The format provides a 'Role Map' feature for precisely this purpose. This is important for exporting the PDF's content into other formats: XML, HTML, plain text, accessible text, Microsoft Word, Excel spreadsheet, PowerPoint slides, etc.

Upon exporting to XML using Acrobat Pro DC [1], the structure tags use the user-defined (i.e., LaTeX) names; that is, no DTD is assumed. Furthermore all the metadata is included in the resulting XML file. On the other hand, export to HTML uses tags based on the role-mapped structure. There is also a 'Class Map' feature that allows style attributes to be added to tags upon export. With proper use of the 'Role Map' and 'Class Map' a 'Tagged PDF' version of a document can be truly 'Universal', much as in the mathematical (categorical) sense. That is, good quality alternative formats of a document's content can (at least in principle) be obtained from the PDF file, using 'Export' options, just as a functor can be factored through a 'Universal Object'. To make this even better, especially when mathematical content is involved, may require collaboration with Adobe and other PDF browser distributors.

3 Future directions

In July 2017, a new version of the PDF specification was released, namely PDF 2.0 [6]. This is 'Tagged PDF', with only a few changes and recommendations from PDF 1.7 [5, 4], and a few new features. There should soon be a release of PDF/UA-2. The main difference will be that mathematics must be described structurally, using MathML tagging [14]. Ross has done some preliminary work [18, 19, 20] which used a non-standard pdfTeX engine, not generally available. It is too soon to directly incorporate this work, but it should be a long-term aim to do so. First we need to get editors and authors used to using 'Tagged PDF' and creating 'Accessible' documents. Generating MathML descriptions, and their inclusion into published articles, can be a goal to be realised perhaps seven or more years hence.

References

[1] Acrobat DC; Adobe Systems Inc. acrobat.adobe.com/au/en/acrobat.html

[2] General Services Administration, U.S. Government-wide Section 508 Accessibility Program. section508.gov

[3] ISO 16684-1:2012; Graphic technology — Extensible metadata platform (XMP) specification — Part 1: Data model, serialization and core properties; Technical

Committee ISO/TC 130 Graphic technology, (February 2012). Reviewed in 2017. iso.org/standard/57421.html

[4] ISO 32000-1:2008; Document management — Portable document format (PDF 1.7); Technical Committee ISO/TC 171/SC 2 (July 2008). Also available as [5]. iso.org/iso/catalogue_detail?csnumber=51502

[5] PDF Reference 1.7; Adobe Systems Inc.; November 2006. Also available as [4]. adobe.com/devnet/pdf/pdf_reference.html

[6] ISO 32000-2:2017; Document management — Portable document format — Part 2: PDF 2.0; Technical Committee ISO/TC 171/SC 2 (July 2017). iso.org/standard/63534.html

[7] ISO 19005-1:2005; Document Management — Electronic document file format for long term preservation — Part 1: Use of PDF1.4 (PDF/A-1); Technical Committee ISO/TC 171/SC 2 (Sept. 2005). Revisions via Corrigenda: ISO 19005-1:2005/Cor 1:2007 (March 2007); ISO 19005-1:2005/Cor 2:2011 (Dec. 2011). iso.org/iso/catalogue_detail?csnumber=38920

[8] ISO 19005-2:2011; Document Management — Electronic document file format for long term preservation — Part 2: Use of ISO 32000-1 (PDF/A-2); Technical Committee ISO/TC 171/SC 2 (June 2011). iso.org/iso/catalogue_detail?csnumber=50655

[9] ISO 19005-3:2012; Document Management — Electronic document file format for long term preservation — Part 3: Use of ISO 32000-1 with support for embedded files (PDF/A-3); Technical Committee ISO/TC 171/SC 2 (October 2012). iso.org/iso/catalogue_detail?csnumber=57229

[10] pdfaPilot; Callas Software GmbH. callassoftware.com/en/products/pdfapilot

[11] 3-Heights PDF Validator; pdftools.com, Premium PDF Technology. pdf-tools.com/pdf20/en/products/pdf-converter-validation/

[12] ISO 14289-1:2012; Document management applications — Electronic document file format enhancement for accessibility — Part 1: Use of ISO 32000-1 (PDF/UA-1). Technical Committee ISO/TC 171/SC 2 (August 2012). Corrected version (December 2014). iso.org/iso.catalogue_detail?csnumber=64599

[13] ISO 14289-1:2014, Document management applications — Electronic document file format enhancement for accessibility — Part 1: Use of ISO 32000-1 (PDF/UA-1). International Standards Organisation, 2014. iso.org/standard/64599.html

[14] ISO/IEC 40314:2016, Information technology — Mathematical Markup Language (MathML), Version 3.0, 2nd Edition; Technical Committee: ISO/IEC JTC 1 Information technology, (February 2016). iso.org/standard/58439.html

[15] The Matterhorn Protocol (version 1.02); PDF Association, 2014. pdfa.org/publication/the-matterhorn-protocol-1/.

[16] PDF/UA-1 Technical Implementation Guide: Understanding ISO 32000-1 (PDF 1.7); AIIM, aiim.org/Global/AIIM_Widgets/Community_Widgets/Technical-Implementation-Guide-32000-1

[17] PDF/UA in a Nutshell; PDF Association, 2012. pdfa.org/download/pdfua-in-a-nutshell

[18] Moore, Ross; *Ongoing efforts to generate "tagged PDF" using pdfTEX*, in *DML 2009, Towards a Digital Mathematics Library, Proceedings*, Petr Sojka (editor), Muni Press, Masaryk University, 2009. ISBN 978-80-20-4781-5. Reprinted in: *TUGboat* Vol.30, No.2 (2009), pp. 170–175. tug.org/TUGboat/tb30-2/tb95moore.pdf

[19] Moore, Ross; Tagged Mathematics in PDFs for Accessibility and other purposes, in CICM-WS-WiP 2013, Workshops and Work in Progress at CICM, CEUR Workshops Proceedings. ceur-ws.org/Vol-1010/paper-01.pdf

[20] Moore, Ross; PDF/A-3u as an archival format for accessible mathematics, in S.M. Watt et al., eds.: CICM 2014, Springer LNAI 8543, pp. 184–199, 2014. springer.com/computer/theoretical+computer+science/book/978-3-319-08433-6

[21] Moore, Ross; Examples of Tagged PDF documents built using LaTeX. maths.mq.edu.au/%7Eross/TaggedPDF/

[22] pdfx — PDF/X and PDF/A support for pdfTEX; Moore, Ross, C.V. Radhakrishnan, Hàn Thế Thành, Selinger, Peter. ctan.org/pkg/pdfx

[23] PRISM: Publishing Requirements for Industry Standard Metadata; Idealliance. idealliance.org/prism-metadata.

[24] Web Content Accessibility Guidelines (WCAG) 2.0 — W3C Recommendation 11 December 2008; Web Accessibility Initiative (WAI) of the World Wide Web Consortium. w3.org/TR/2008/REC-WCAG20-20081211

◇ Ross Moore
maths.mq.edu.au/~ross/TaggedPDF

FreeType_MF_Module: A module for using METAFONT directly inside the FreeType rasterizer

Jaeyoung Choi, Ammar Ul Hassan, Geunho Jeong

Abstract

METAFONT is a font description language which generates bitmap fonts for the use by the TEX system, printer drivers, and related programs. One advantage of METAFONT over outline fonts is its capability for producing different font styles by changing parameter values defined in its font specification file. Another major advantage of using METAFONT is that it can produce various font styles like bold, italic, and bold-italic from one source file, unlike outline fonts, which require development of a separate font file for each style in one font family. These advantages are especially applicable when designing CJK (Chinese-Japanese-Korean) fonts, which require significant time and cost because of the large number of characters used in Hangeul (Korean character) and Hanja (Chinese character). However, to use METAFONT in current font systems, users need to convert it into its corresponding outline font. Furthermore, font rendering engines such as FreeType don't support METAFONT.

In this paper, we propose *FreeType_MF_Module* for the FreeType rasterizer. The proposed module enables direct usage of METAFONT just like any other font (outline or bitmap) supported in the FreeType rasterizer. Users of METAFONT don't need to pre-convert METAFONT fonts into corresponding outline fonts as FreeType_MF_Module automatically performs this. Furthermore, FreeType_MF_Module allows the user to easily generate multiple font styles from one METAFONT source file by changing parameters.

1 Introduction

In today's information society, much traditional pen and paper usage for communication between people has been increasingly replaced by computers and mobile devices. Text has become an effective source for gathering information and a means of communication between people. Although people commonly use smart devices these days with effective resources like media and sound, text generally plays the key role of interaction between user and device. Text is composed of characters, and these characters are physically built from specific font files in the digital environment's system.

Fonts are the graphical representation of text in a specific style and size. These fonts are mainly categorized in two types: outline fonts and bitmap fonts. Outline fonts are the most popular fonts for producing high-quality output used in digital environments. However, to create a new font style as an outline font, font designers have to design a new font with consequent extensive cost and time. This recreation of font files for each variant of a font can be especially painful for font designers in the case of CJK fonts, which require designing of thousands individual glyphs one by one. Compared to alphabetic scripts, CJK scripts have both many more characters and generally more complex shapes, expressed by combinations of radicals [3]. Thus it often takes more than a year to design a CJK font set.

A programmable font language, METAFONT, has been developed which does not have the above disadvantages of outline fonts. METAFONT is a programming language created by D.E. Knuth [1] that generates TEX-oriented bitmap fonts. A METAFONT source file is radically different from an outline font file: it consists of functions for drawing characters and has parameters for different font styles. By changing the parameters defined in a font specification file, various font styles can be easily generated. Therefore, a variety of font variants can be generated from one METAFONT source.

However, in practice users are unable to use METAFONT on modern systems, because current font engines like FreeType [2] do not provide any direct support of METAFONT. Unlike standard bitmap and outline fonts, METAFONT is expressed as a source code that is compiled to generate fonts. To use METAFONT in a general font engine like the FreeType rasterizer, users have to convert each metafont into its corresponding outline font. When it was developed in the 1980s, standard PC hardware was not fast enough to do real-time conversion of METAFONT source into a corresponding outline font. Current PC hardware, however, is fast enough to do such real-time conversion.

In this paper, a METAFONT module for the FreeType rasterizer (FreeType_MF_Module) is proposed. The proposed module enables direct use of METAFONT in FreeType, just like any other outline and bitmap font modules. With FreeType_MF_Module, users don't need to pre-convert a METAFONT font into its corresponding outline font before using it with the FreeType rasterizer, as FreeType_MF_Module automatically performs this. It allows users to easily generate variants of font styles by applying different parameter values. This module is directly installed in the FreeType rasterizer just like

its default font modules, thus minimizing any reliability and performance issues. We have tested our proposed module by generating different font styles with METAFONT and compared its performance with default FreeType modules and our previous research.

This paper is organized as follows. In Section 2, related research regarding font modules and libraries is discussed. The architecture of FreeType_MF_Module is explained in Section 3. Section 4 demonstrates how FreeType can support METAFONT, via some testing of FreeType_MF_Module. The performance of FreeType_MF_Module is also compared with FreeType default modules and other researches in this section. Section 5 gives concluding remarks.

2 Previous research and its problems

MFCONFIG [4] is a plug-in module for Fontconfig [5]. It enables the use of METAFONT on GNU/Linux and other Unix font systems. Figure 1 shows the architecture of the MFCONFIG module linked with Fontconfig.

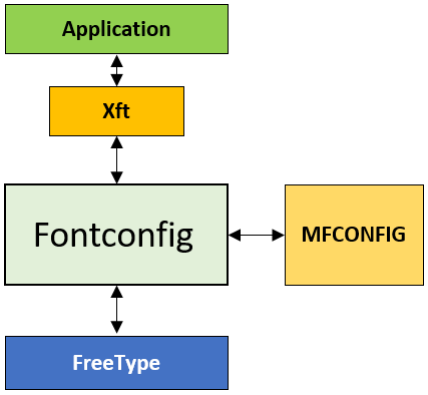


Figure 1: Basic architecture of MFCONFIG module

Although MFCONFIG does provide support for METAFONT in Fontconfig-based font systems, it has performance and dependency problems. Since MFCONFIG is plugged into a high-level font system, i.e., Fontconfig, and not at the low-level FreeType rasterizer, its performance is very slow compared to the font-specific driver modules supported by FreeType. Whenever the client application sends a METAFONT file request, Fontconfig communicates with MFCONFIG, performs operations, and then sends input to FreeType for rendering text. This whole process becomes slow because of the high-level operations before FreeType receives its input.

Other than the performance problem, MFCONFIG also has a dependency problem. As it works with the Fontconfig library, this means that in a font environment not using Fontconfig, this module cannot be used. Fontconfig is mainly used in the font system for GNU/Linux and some other Unix operating systems, so MFCONFIG cannot be supported in other environments, such as Windows and Mac OS X.

VFlib [6], a virtual font library, is a font rasterizer developed for supporting multilingual fonts. VFlib can process fonts which are represented in different font formats and outputs glyphs as bitmap images from various font files. VFlib supports many font formats like TrueType, Type 1, GF, and PK bitmaps [7], et al. It provides a unified API for accessing different font formats. A new module can be added in this font library for adding support for METAFONT but this library has its own drawbacks: as it supports many different font formats, and requires support from a database, it can be too heavy for embedded systems. It is also dependent on additional font libraries, such as the FreeType engine for TrueType font support and T1lib [8] for Type 1 font support, so it has its own dependency problems as well. Therefore, VFlib is not suitable for adding METAFONT support.

FreeType is a font rasterizer. It can produce high quality output for mainly two kinds of font formats, both outline and some bitmap formats. FreeType mainly supports font formats such as TrueType, Type 1, Windows, and OpenType fonts using the same API, independent of the font format. Although FreeType supports many different font formats, it doesn't provide any support for METAFONT directly. If there were a module for FreeType that directly supports METAFONT, users could take advantage of the METAFONT features above, e.g., generating variants of font styles by just changing parameter values. MFCONFIG's problems can also be resolved using such a module.

The proposed FreeType_MF_Module in this paper reuses the process for printing METAFONT from the MFCONFIG module. FreeType_MF_Module intends to solve the two problems of the MFCONFIG module. METAFONT can be used with any system having FreeType using the proposed module. As it is implemented like any other default FreeType module, it can be easily installed or uninstalled.

3 Implementation of FreeType_MF_Module in the FreeType rasterizer

3.1 FreeType_MF_Module as an internal module of FreeType

FreeType can support various font formats. Processing a font file corresponding to its format is done by an internal module in FreeType. This internal module is called a font driver. FreeType contains a configuration list of all driver modules installed, in a specific order. When FreeType receives a request

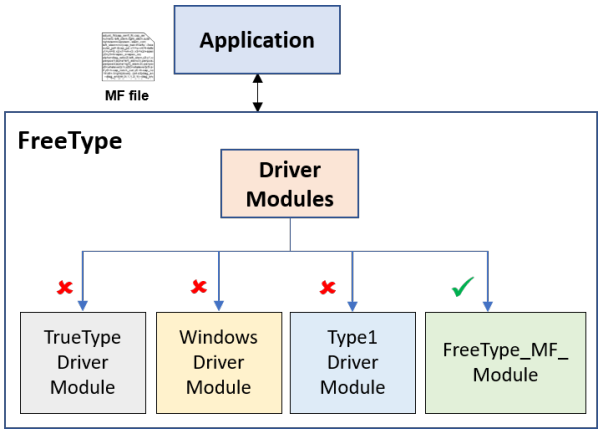


Figure 2: Process of selecting a module in FreeType

for a font file from an application, it passes this request to the driver module at the top of the list for processing. This module performs some internal operations to check if this font format can be processed or not. If this driver module supports the request, it performs all other operations to process the font file request. Otherwise the request is sent to the second driver module mentioned in the list. This process continues until a font driver is selected for processing the font file request. If no font driver can process the request, an error message is sent to the client application.

In our case, FreeType_MF_Module is directly installed inside FreeType just like its other internal modules. When the client application sends a request for a METAFONT file, FreeType_MF_Module receives this request and processes it. Figure 2 shows how FreeType will select a driver module for processing a METAFONT file request.

FreeType_MF_Module consists of three submodules: Linker module, Administrator module, and Transformation module.

3.2 Linker module

Linker module is the starting point of FreeType_MF_Module. It is mainly responsible for linking FreeType internal modules with FreeType_MF_Module. It is divided into two parts: inner meta interface and outer meta interface. The inner meta interface receives font file requests from internal modules and delivers it to the Administrator module for processing. After processing by the Administrator module, outer meta interface delivers the response to internal modules for further operations. The process of Linker module is shown in Figure 3.

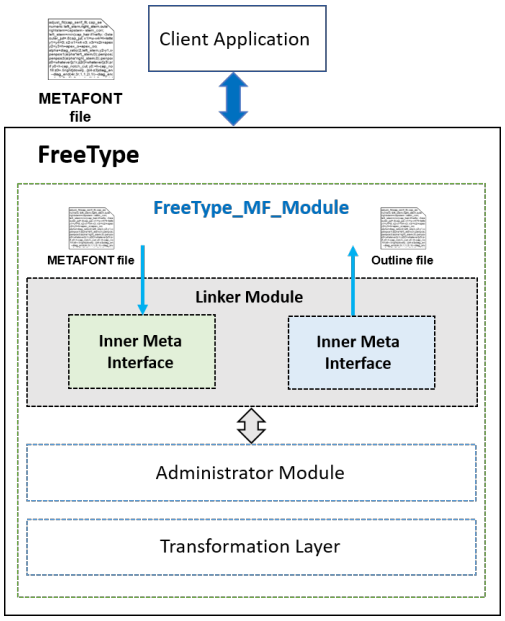


Figure 3: Linker module

3.3 Administrator module

The core functionality of FreeType_MF_Module is performed in the Administrator module. This module is divided into two layers: Search layer and Management layer.

The Search layer is responsible for finding all the installed METAFONT fonts in a table. This table contains a list of all the METAFONT fonts installed and how to fetch information related to them. The Search layer is implemented in Meta scanner and Meta table.

The Management layer mainly performs the following tasks:

1. Checking whether or not the requested font file is METAFONT.
2. Checking the cache to determine if the corresponding outline font for the METAFONT request is already stored. If yes, it sends the response directly from the cache. This functionality is implemented to achieve better performance and reusability.
3. If the outline font is not prepared in the cache, this request is sent to the Transformation layer. The outline font prepared by the Transformation layer is stored in the cache.
4. The response is sent back to FreeType internal modules by the Management layer.

The Management layer is implemented in three parts: Meta analyzer, Meta request, and Meta cache. Figure 4 shows the Administrator module and its sublayers.

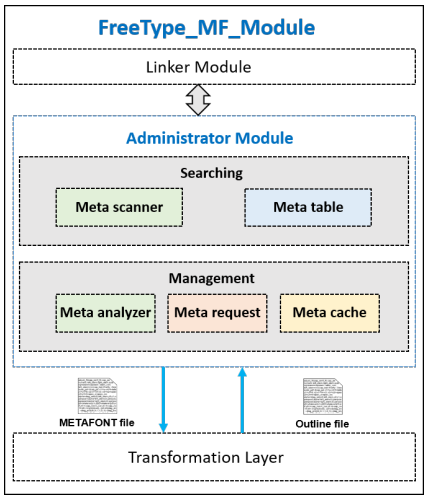


Figure 4: Administrator module

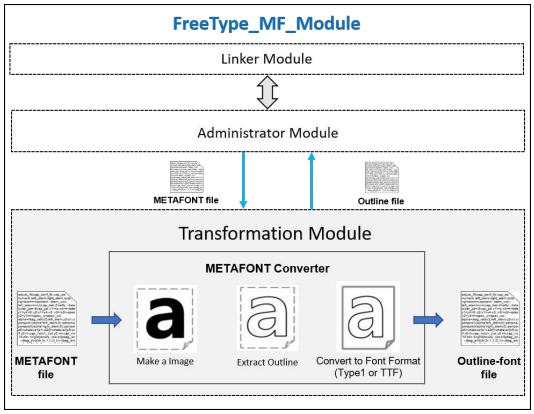


Figure 5: Transformation module

3.4 Transformation module

The Transformation module is mainly responsible for converting the METAFONT file into its corresponding outline font file. If the outline font file for a requested METAFONT file doesn't exist in the table then the Administrator module sends the request to the Transformation module. This module processes the request and returns the corresponding outline font file to the Administrator module. Figure 5 shows how the Transformation module converts METAFONT files into corresponding outline files.

3.5 METAFONT support in FreeType using FreeType_MF_Module

The overall architecture of FreeType_MF_Module is shown in Figure 6. As seen there, FreeType_MF_Module is an internal module of FreeType which is responsible for processing METAFONT file requests.

- First an application sends a font file request to FreeType (step 1).
- If all other driver modules fail to process this font file request, the request is sent to FreeType_MF_Module through the Linker module. Inner meta interface delivers this request to the Administrator module (step 2).
- A Meta request in the Administrator module receives all the information in this font file request and sends it to the Meta Analyzer to check if this font file is METAFONT or not (step 3). If this font file is not METAFONT this request is sent back to FreeType (step 3a). If this request is METAFONT, The Meta analyzer checks if this METAFONT file is installed or not by scanning the Meta table. If not found in the Meta table, an error is sent back to FreeType internal modules (step 3b).
- There can be a scenario in which the METAFONT font is installed but its corresponding outline font is not stored in the cache. In this case, the Meta cache is scanned to check if the corresponding outline file is stored in it (step 4). If it is already stored in the Meta cache with the same style parameters as requested, it is directly sent to FreeType (step 4a). If it is not stored in Meta cache, the request is sent to the Transformation layer (step 4b).
- The Transformation layer converts the METAFONT file into its corresponding outline font by applying the requested style parameters (step 5).
- An outline font is returned from the Transformation module to the Administrator module where the Meta cache is updated for future reuse (step 6).
- The outer Meta interface returns this outline font to core FreeType for further processing (step 7).

Lastly, FreeType renders this outline font that was made from the requested METAFONT with the style parameter values.

The FreeType_MF_Module is perfectly compatible with the standard FreeType rasterizer. FreeType_MF_Module provides direct support of METAFONT in FreeType rasterizer just like its default Type1 driver module, TrueType driver module, etc. The module manages the METAFONT font and its conversion to the corresponding outline font. Client applications can request any style parameters of METAFONT; FreeType_MF_Module processes them and the result is displayed on the screen as usual. As it is directly implemented inside the FreeType rasterizer, it has no dependency problems as discussed in Section 2. FreeType_MF_Module can easily generate multiple font families like bold, italic, and bold-italic depending on the style parameter values passed to it.

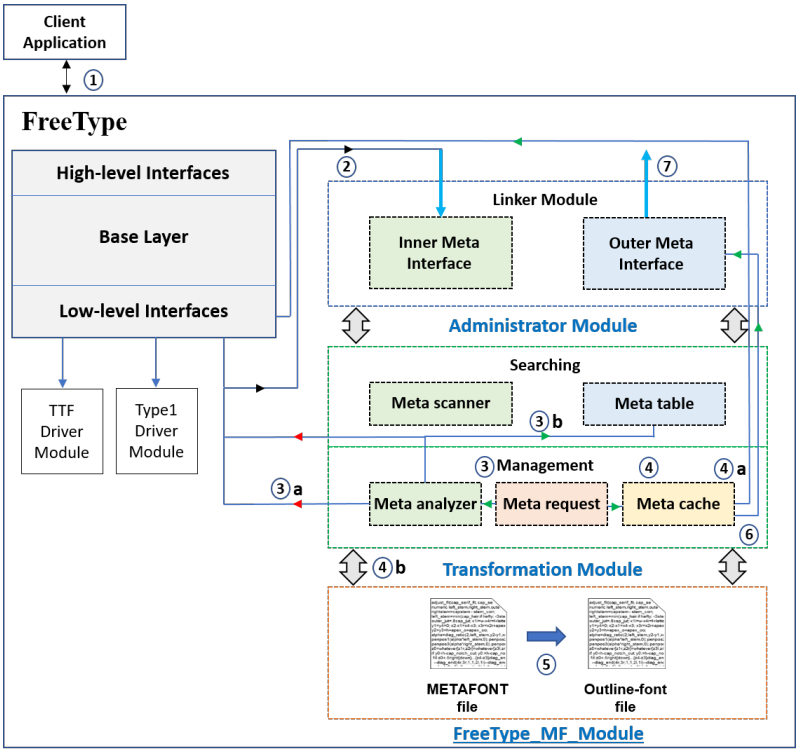


Figure 6: FreeType_MF_Module architecture

Table 1: FreeSerif font family

Style	Styled Output	Font files
Normal	FreeSerif	FreeSerif.ttf
Bold	**FreeSerifBold**	FreeSerifBold.ttf
Italic	*FreeSerifItalic*	FreeSerifItalic.ttf
Bold + Italic	***FreeSerifBoldItalic***	FreeSerifBoldItalic.ttf

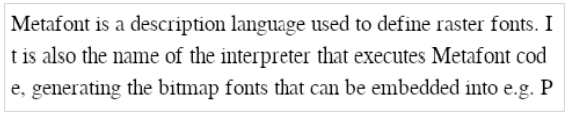
Table 2: Various font styles with Computer Modern

Style	Styled Output	Parameter values
Normal	ComputerModern	Default values of stem, hair, curve, slant
Bold	**ComputerModern**	stem+20, hair+20, curve+20, slant default
Italic	*ComputerModern*	Default values of stem, hair, curve, slant= 0.4
Bold + Italic	***ComputerModern***	stem+20, hair+20, curve+20, slant = 0.4

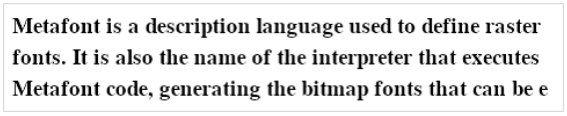
4 Experiment and performance evaluation of FreeType_MF_Module

For the experiment of using FreeType_MF_Module to generate different font styles from METAFONT source, the authors used a font viewer application in GNU/Linux. This application directly uses FreeType to render fonts. It takes a font file and text as input and displays the styled text on the screen using the X Windows System. For testing, the authors have used all four styles of FreeSerif font family as TrueType fonts, i.e., normal, bold, italic, bold-italic, comparing with the Computer Modern fonts in METAFONT.

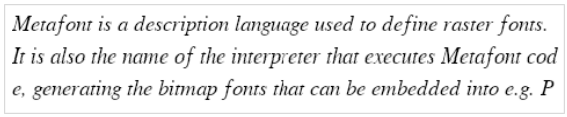
Table 1 shows the FreeSerif font family in four different styles. These styles are generated by using four different font files. Table 2 shows Computer Modern in the same four styles, made using different parameter values. These styles are made from one single METAFONT source file. The parameter values which are modified for generating these font styles are hair, stem, curve, and slant. The three parameters hair, curve, and stem are related with the bold style. Increasing their value increases the boldness of text. These parameter values are different for lowercase and uppercase characters. The slant parameter is related to the italic style. As shown in Table 2, for the normal style the default values of all four parameters are used. For bold style, the values used are stem+20, hair+20, curve+20, and slant parameter default value. Default values of stem, hair, curve, and slant= 0.4 are used for italic style. Whereas,



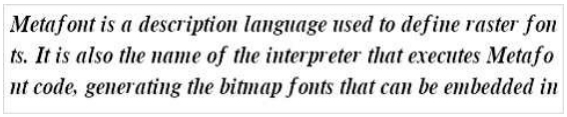
(a) FreeSerif normal style



(b) FreeSerif bold style

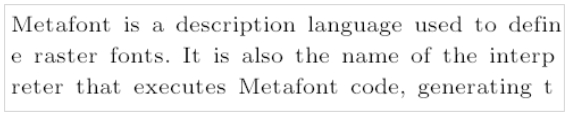


(c) FreeSerif italic style

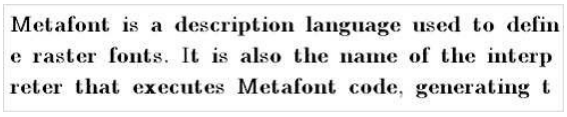


(d) FreeSerif bold-italic style

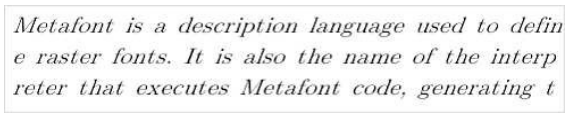
Figure 7: Dataset rendered in FreeSerif (enlarged from screen resolution)



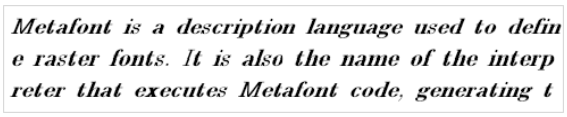
(a) default values of stem, hair, curve, slant



(b) stem+20, hair+20, curve+20, slant default



(c) default values of stem, hair, curve; slant = 0.4



(d) stem+20, hair+20, curve+20, slant = 0.4

Figure 8: Dataset rendered in Computer Modern (enlarged from screen resolution)

stem+20, hair+20, curve+20, slant = 0.4 values are used for bold-italic style. Similarly, many other font styles can be generated with this single METAFONT source file by changing parameter values.

To test the performance of FreeType_MF_Module compared to FreeType default driver modules and the MFCONFIG module, another experiment was performed using the same font viewer application. All four font files of FreeSerif in Table 1 were used for testing the TrueType driver module of FreeType; Computer Modern source files were used with four different parameter values to generate four different styles in Table 2. For the text input, a sample dataset was used which consisted of 2,000 words and over 8,000 characters, including space characters. The average time in milliseconds between the font style request from application and the successful display of styled text on the screen was computed and compared.

Figure 7 shows the result of printing four FreeSerif fonts and Figure 8 shows the result of four Computer Modern METAFONT fonts. Table 3 shows the average time to print the dataset using the FreeSerif font by the TrueType driver module, the Computer Modern font by FreeType_MF_Module, and Computer Modern with the MFCONFIG module. The default TrueType driver module in FreeType takes 3ms to 7ms to print the dataset with all four FreeSerif families. FreeType_MF_Module takes 4ms to 10ms to print this dataset with Computer Modern, whereas the MFCONFIG module took 50ms to 120ms to display a similar size dataset.

The performance of FreeType_MF_Module is comparatively slower than default FreeType driver module because it takes extra time to convert a METAFONT into its corresponding outline font by applying the style parameters. On the other hand, FreeType_MF_Module has very good performance relative to the MFCONFIG module, because it is directly implemented inside the FreeType rasterizer and not dependent on other font libraries like Fontconfig and Xft [9] etc. Hence, we can conclude that FreeType_MF_Module in the FreeType rasterizer can provide direct support of METAFONT usefully in practice, in almost real time on a modern PC.

Performance can be further improved by optimizing the METAFONT converter in the Transformation layer. Currently, the METAFONT converter works with the mftrace and autotrace programs. Future work will consider proposed module optimization and direct usage of TEX bitmap fonts like GF and PK which are not supported by the FreeType rasterizer.

FreeType_MF_Module is a suitable module for providing users with parameterized font support on the screen by applying style parameters directly to the METAFONT font. To reiterate, users don't need to pre-convert METAFONT fonts into outlines before

Table 3: Average time to display dataset with the TrueType driver, FreeType_MF_Module, and MFCONFIG

Style	TrueType driver Avg. Time	FreeType_MF_Module Avg. Time	MFCONFIG Avg. Time
Normal	4.5 ms (3-6)	6 ms (5-8)	70 ms (50-80)
Bold	4 ms (4-6)	7 ms (6-9)	85 ms (70-100)
Italic	4 ms (4-6)	6 ms (4-7)	105 ms (70-110)
Bold + Italic	5 ms (5-7)	8 ms (6-10)	100 ms (90-120)

using with FreeType, as FreeType_MF_Module automatically performs this step. Users see no difference between METAFONT fonts and TrueType fonts using FreeType with this module. FreeType_MF_Module has also overcome the performance and dependency problems of the MFCONFIG module.

5 Conclusion

In this paper, we have proposed a module, named FreeType_MF_Module, enabling direct support of METAFONT in the FreeType rasterizer. Outline fonts such as TrueType and Type 1 do not allow users to easily change font styles. For every different font style in outline fonts, a new font file is created, which can be a time consuming and costly process for CJK fonts consisting of large numbers of complex characters. METAFONT fonts do not have this disadvantage.

FreeType supports many different font formats, including TrueType, Type 1, Windows fonts, etc., but does not provide any support for METAFONT. The proposed module is installed directly inside FreeType and can be used like any other internal module to support METAFONT fonts. The authors have reported on experiments demonstrating that a variety of styled fonts can be generated from METAFONT by adjusting parameter values in a single METAFONT source file by FreeType_MF_Module in almost real time.

Acknowledgement

This work was supported by an Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIT) (No. R0117-17-0001, Technology Development Project for Information, Communication, and Broadcast).

References

[1] Donald E. Knuth, *Computers and Typesetting,* Volume C: *The METAFONTbook*. Addison-Wesley, 1996.

[2] David Turner, Robert Wilhelm, Werner Lemberg, *FreeType*. `freetype.org`

[3] Jinpyung Kim, et al. *Basic Study of Hangul Font,* Seoul: Korea Publishing, Research Institute, 1988.

[4] Jaeyoung Choi, Sungmin Kim, Hojin Lee, Geunho Jeong, *MFCONFIG: A METAFONT plug-in module for the FreeType rasterizer*, *TUGboat* 37:2, pp.163–170, 2016. `tug.org/TUGboat/tb37-2/tb116choi.pdf`

[5] *Fontconfig*. `freedesktop.org/wiki/Software/fontconfig`

[6] Hirotsugu Kakugawa, *VFlib — a general font library that supports multiple font formats,* EuroTEX conference, March 1998. `www-masu.ist.osaka-u.ac.jp/~kakugawa/VFlib/vflib35.ps`

[7] Tomas Rokicki, *The GFtoPK processor Version 2.4,* 06 January 2014. `ctan.org/pkg/mfware`

[8] Rainer Menzner, *A Library for Generating Character Bitmaps from Adobe Type 1 Fonts*. `inferiorproducts.com/docs/userdocs/t1lib/t1lib_doc.pdf`

[9] Keith Packard, *The Xft font library: Architecture and Users Guide*, Proceedings of the 5th annual conference on Linux Showcase Conference, 2001. `keithp.com/keithp/talks/xtc2001/xft.pdf`

◇ Jaeyoung Choi
Ammar Ul Hassan
Geunho Jeong
369 Sangdo-Ro, Dongjak-Gu
Seoul 06978, Korea
choi@ssu.ac.kr
ammar@ssu.ac.kr
ghjeong@gensolsoft.com

WeTEX and Hegelian contradictions in classical mathematics

S. K. Venkatesan

Abstract

We consider the contradiction between WYSIWYG and LaTeX markup, in order to demonstrate a Hegelian synthesis (WeTEX, a KaTEX-based JavaScript implementation) of this contradiction. We then briefly consider other contradictions such as form vs. content in mathematics typesetting. After that we move on to the Hegelian contradictions in classical mathematics, starting from Zeno's paradox, leading to the hierarchy of infinities, continuum hypothesis, and finally the problem of algorithmic complexity classes.

"Reason has always existed, but not always in a reasonable form" — Karl Marx.

1 Introduction

Contradictions are mills through which reality ebbs and flows. The right and left hand sides exist as opposing manifestations. The fact that the perfect symmetry of the left and right is broken is what establishes their distinction. If Earth were a perfect sphere, then there would not be rivers and valleys that bring life to it. However, since the radius of the earth is almost a constant (to more than 99% accuracy), we are qualified to call it a sphere.

Contradictions are also a great source of changes to society and transitions from one form to another. Goliath was powerful when combat was direct physical conflict with hand-held weapons. When the giant Goliath calls David to come near him to fight, it is clear that only at that range can he overcome David. David, at the same time, keeps his distance to deny him that opportunity and defeats him with his long-distance weapon. Inside the atomic nucleus such a conflict between the Goliath of nuclear forces that operate only at short distances releases an enormous amount of energy when they come in conflict with electrical repulsion that operates at long distances, when the size of the nucleus becomes sufficiently large, as in the case of the uranium or plutonium nucleus. At the other end of the spectrum, when gravitational forces crush electrical repulsion in hydrogen ions, the Goliathian nuclear forces take over, releasing the enormous energy that powers suns.

In this article we first consider WeTEX, a KaTEX-based JavaScript implementation of an equation editor that resolves the conflict between the WYSIWYG (What You See Is What You Get) paradigm and the LaTeX macros, a WYSIWYM (What You See Is What You Mean) system. We see how both of these contradictory approaches work at different user parametric ranges and use cases. The synthesis of these opposing paradigms produces a new application, WeTEX.

In the third section we consider Zeno's paradox, countable infinity and uncountable infinities of higher order. We also show how these Hegelian contradictions lead to a hierarchy of infinities.

In the fourth section we consider the Cantor set in the context of the continuum hypothesis and in the fifth and final section we consider a class of recursive algorithms for constructing the Cantor set and its complexity.

2 WeTEX — a synthesis between WYSIWYG and WYSIWYM

LaTeX markup solved the problem of typesetting documents using a system of macros involving braces and backslashes. WYSIWYG systems, on the other hand, use a system of graphical menu objects with place holders for inserting text and symbols. This allows the user to directly visualize the output instantly with instant gratification. On the other hand, it is not easy to search for symbols in a character palette in a WYSIWYG system, so LaTeX-like backslashed named entities are more convenient. Autocompletion prompts can further improve productivity in authoring such macro entities in LaTeX, as some LaTeX editors provide.

The WeTEX system was created as a proof of concept for a hybrid system. At present we deal only with authoring equations. We have used KaTEX's JavaScript rendering engine to implement this math editor. The KaTEX source code is available in a Github repository, `github.com/Khan/KaTeX`. WeTEX's open source (GPL-licensed) code is available in the Github repository `github.com/Sukii/WeTeX`.

In addition to the standard LaTeX (KaTEX flavor) macros, WeTEX defines some additional macros and keyboard shortcuts, described in Table 1.

In addition to typesetting equations we will also be making an attempt to make the mathematics computable, wherever possible. A preliminary attempt at integration with Maxima [1], an open source GPL licensed maths computing library, has been made at `mathml.in`. Here the contradiction is between form and content, as we will be attempting to walk on two legs, quite similar to the literate programming approach for LaTeX macro packages where one has to make a fine balance between the code and documentation. One the one hand the attempt of presentation aspect is beautiful typography, as an endeavor of a metal-based art form developed for many centuries in Europe. However, all this beauty has to be rooted in objectivity and realism, especially

Table 1: WeTEX macros and shortcut keys

WeTEX input	*Shortcut key*	*Description*	*Sample output*
`_`	shift `_`	Subscript	A_d
`^`	shift `^`	Superscript	A^d
`\t[]`	ctrl `-`	Tensor	A^{ijk}_{abc}
`\f[]`	shift `%`	Fraction	$\frac{a}{d}$
`\r[]`	shift `!`	Square root	$\sqrt{456}$
`\x[]`	shift `@`	Unicode text	Unicode
`\q[]`	shift `#`	Cube root	$\sqrt[3]{082}$
`\d[]`	ctrl `d`	Differential derivative	$\frac{\mathrm{d}X}{\mathrm{d}Z}$
`\p[]`	ctrl `p`	Partial derivative	$\frac{\partial X}{\partial Z}$
`\i[]`	alt `i`	Integration	$\int_X^Z Y\ \mathrm{d}X$
`\s[]`	alt `s`	Summation	$\sum_X^Y Z$
`(())`	ctrl `(`	Left & right parentheses	$\left(\frac{a}{z}\right)$
`[[]]`	ctrl `[`	Left & right brackets	$\left[\frac{a}{z}\right]$
`{{}}`	ctrl `{`	Left & right braces	$\left\{\frac{a}{z}\right\}$
`\8`	ctrl `8`	Infinity symbol	∞
`\0`	ctrl `.`	Centered single dot	$\cdot$
`shift ~`		Similar	$\sim$
`ctrl =`		Equivalent	$\equiv$

the functional requirement of mathematics and its development rooted in industrial society, publishing books and journals both in electronic and print forms. Modern electronic devices also now allow us to develop newer forms that can interact more closely with users both in form and content, allowing much broader dissemination of content, forcing us to think about accessibility requirements of content transcending the current form. However the spirit of accessibility is not only a technical requirement, but also requires that we change the form (pun unintended) of the content also, making it accessible to a broader audience.

3 Classical Hegelian hierarchy of infinities

We will begin with the classical Zeno's paradox that is usually explained in terms of Achilles and the tortoise [2].

3.1 Zeno's paradox

However, we will illustrate this in a much simpler way using the concept of recursive decimals:

$$0.99999\cdots = 1\,.$$

We know from this that there is an infinite sequence of numbers, 0.9, 0.99, 0.999, ..., which are bounded above by the finite value 1.0. The finite value 1.0 can be expressed by the infinite sequence of decimals represented by the above equation. Of course, there are many such infinite sequences that approach 1.0, each with its own rhyme and rhythm. The fact that the infinite is contained in the finite value of 1.0 is indeed a paradox in classical logic, but not in paraconsistency logic [3], which allows such an inconsistency to exist without great consternation. In classical logic A and −A cannot exist as truth together. It would create a blow-up. Consider the classical Boolean logic statement:

$$\text{(A or B) and (}-\text{A or B) is true}\,.$$

In classical logic we can conclude from this statement that B is true. This is considered a blow-up, as in classical logic an unrelated statement B would be true if both A and −A were to be true. Paraconsistency logic [3] mitigates this blow-up. The name "paraconsistency" was coined by the Peruvian philosopher Francisco Miró Quesada [4].

3.2 Hegelian hierarchy of infinities in classical mathematics

Let us now consider the cardinality of a set as the number of elements in a set, for example, if

$$A = \{\text{dog, cat, horse, car, bus, train, aeroplane}\}$$

then $\#(A) = 7$. Now consider the set of all subsets of A, symbolically written as the set 2^{A}. This notation for the power-set is justified by the relation

$$\#(2^{\mathrm{A}}) = \sum_{i=0}^{\#(\mathrm{A})} \binom{n}{i} = 2^{\#(\mathrm{A})}\,.$$

We would like to point out that this process can be continued indefinitely, i.e., we can construct the power-set of a power-set or, in plain language, that there can be a set of all subsets of the set of all subsets of a set, etc., i.e.,

$$\mathrm{A}, 2^{\mathrm{A}}, 2^{2^{\mathrm{A}}}, \ldots\,.$$

Now let us consider the set of natural numbers,

$$\mathrm{N} = \{1, 2, 3, \ldots\}$$

The cardinality of natural numbers is denoted by aleph,

$$\aleph = \#(N)\,.$$

Now consider the cardinality of the set of all real numbers between 0 and 1, i.e.,

$$c = \#([0, 1])]\,.$$

Just as the number of drops of water in a glass of water cannot be counted like a bunch of bananas, we intuitively know that the set of real numbers in [0,1] cannot be counted like natural numbers. We will prove this now.

Let us represent all real numbers between 0 and 1 in terms of their binary representation in some counting order,

$$\begin{aligned} a_1 &= 0.01010\cdots \\ a_2 &= 0.110010\cdots \\ a_3 &= 0.101011\cdots \\ &\cdots\,. \end{aligned}$$

By a diagonalization process discovered by Cantor we can construct a real number,

$$b = 0.100\cdots ,$$

where b is obtained by reversing the 0s and 1s of the i-th digit of a_i, such that

$$b \notin \{a_1, a_2, a_3, \ldots\} ,$$

leading to a contradiction, thus proving that it is not countable. However, this is only a qualitative result. We will now prove a quantitative result, that

$$c = 2^{\aleph} .$$

In order to prove this, consider the binary representation of a real number in (0,1),

$$x = \{0.010011001\cdots\} .$$

We now obtain the corresponding subset of natural numbers by considering all the index positions of "1" in the above binary representation, i.e.,

$$\mathrm{S} = \{2, 5, 6, 9, \ldots\} .$$

Similarly, for every subset of natural numbers we can construct a real number in (0,1). This implies a bijective mapping between the power-set of natural numbers and the set (0,1), proving the result. QED

So finally we also have the result that we can construct an infinite hierarchy of infinities, i.e.,

$$\aleph, 2^{\aleph}, 2^{2^{\aleph}}, \ldots .$$

Although here we are only discussing about numbers and the mathematics of set theory, these results have much greater implication in computer science as there is a corresponding categorical mapping at higher levels that maps these domains to similar problems there. At an abstract level, decision problems can be considered as function mappings,

$$f :\ A \to \{0, 1\} ,$$

where $A \subset \mathrm{N}$.

So in essence a decision problem can be mapped to a real number. However, at the same time it can be shown that the set of algorithms or procedural programs is countable, as a Turing machine can be reduced to a natural number, a binary state of the computer. Putting the two facts together, we get the result that not all decision problems can be solved accurately by a computer as real numbers are uncountable. However, rational numbers are countable (as they are like two-dimensional natural numbers, they can be counted in a zig-zag way starting from the top right corner) and they are dense in real numbers (which means that a sequence of rational numbers can sufficiently approximate any given real number). So every decision problem can be solved approximately by a computer, although the degree of approximation varies depending on the decision problem and the computer algorithm.

4 The Cantor set and the continuum hypothesis

The continuum hypothesis states that there are no cardinal numbers between $\aleph$ and $2^{\aleph}$. We will now argue against this, but the argument has to be considered from the point of view that there are better measures that distinguish between different shades of infinities than just counting bananas, namely, by weighing them.

Let us now consider the Cantor set, $\mathscr{C}$, which can be obtained by recursively removing the middle one-third (but keeping the end points during the removal) of the set of real numbers in [0,1]. Consider the sequence of sets, $C_1 = [0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$, $C_2 = [0, \frac{1}{9}] \cup [\frac{2}{9}, \frac{1}{3}] \cup [\frac{2}{3}, \frac{7}{9}] \cup [\frac{8}{9}, 1]$, ..., defining the Cantor set to be

$$\mathscr{C} = \bigcap_{k=1}^{\infty} C_k .$$

It can also be symbolically written as a geometric set following the relation

$$3 \times \mathscr{C}(0) = \mathscr{C}(0) \cup \mathscr{C}(2/3) .$$

Or to put it more simply, when we scale the Cantor set by 3, we get two Cantor sets, i.e.,

$$3^D = 2 ,$$

where $0 < D < 1$ is the scaling dimension of the Cantor set and from this relation we obtain $D = \log 2 / \log 3 \approx 0.63$.

For example, we can see that natural numbers are 0-dimensional points in Euclidean space that don't scale, while the interval [0,1] scales linearly, a two-dimensional square scales quadratically, a three-dimensional cube grows to the cubic power, etc.

We can also prove easily that the Cantor set is uncountable. Consider the ternary representation of a real number in (0,1), i.e.,

$$x = 0.0102201\cdots .$$

The points in the Cantor sets will not have digit "1" in them, i.e.,

$$x = 0.002022002\cdots$$

Similar to the proof involving the binary representation of real numbers we can show that the Cantor set, $\mathscr{C}$, is uncountable. Using a similar procedure we can also show that

$$\#(\mathscr{C}) = 2^{\aleph} .$$

One then wonders how this is a counter example to the continuum hypothesis? The answer lies not in counting bananas but weighing bananas, as there is more geometric information in the Cantor set that is lost in counting rather than weighing them. It is this extra geometric information that is captured by the scaling dimension, which distinguishes it as a

fractal object existing between the 0-dimension and 1-dimension. This quantitative aspect becomes clear as we deal with algorithms that generate the Cantor set and their complexity in the next section.

5 Algorithmic complexity of the Cantor set and the power of iterative functional formulations

Let us now consider the iterative algorithm for constructing a Cantor set by considering the n-digit ternary representation of real number between 0 and 1, such as

$$x = 0.\underbrace{012102002\cdots1020}_{n} .$$

The Cantor set follows the recursive relation,

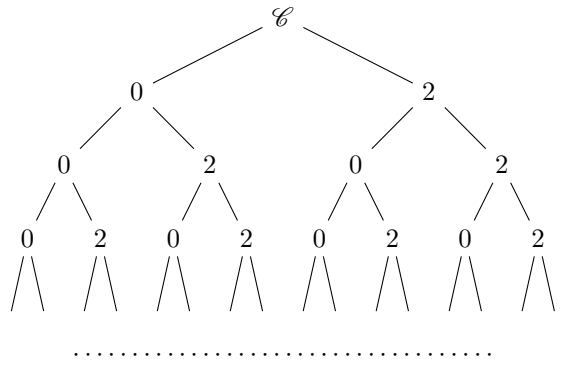
$$\mathscr{C} = \frac{1}{3}\mathscr{C} \cup \left(\frac{1}{3}\mathscr{C} + \frac{2}{3}\right) .$$

This can be used to define a recursive push-down from the left,

$$x \to \begin{pmatrix} \frac{1}{3}x \\ \frac{1}{3}x + \frac{2}{3} \end{pmatrix},$$

which can be represented using a constant push of 0s and 2s from the left as in the tree-like representation shown in Figure 1.

Figure 1: Binary tree formulation of the Cantor set



To construct the real numbers between 0 and 1 in an n-digit ternary representation, we need a decision tree with 3^n steps, while to construct the Cantor set, we need a tree of 2^n steps. As there are no further information or constraints that can be retrieved either from geometry or from its ternary representation, so this is the minimal complexity that can be achieved in order to compute the points in the Cantor set, $\mathscr{C}$. This then proves that the Cantor set computation problem cannot be solved in polynomial time.

However, we have discovered how a single algorithmic step in recursive functional formulation requires 2^n operations in the state machine. Checking the results (2^n states of n-digits) of this output to see if these n-digits are distinct points in $\mathscr{C}$ requires equal or more effort. However, if we are using a stateless function to generate the output then it is enough to test only a few points in the Cantor set to check for the veracity of 2^n values (just as in a cooked pot it is enough to test a few particles of rice to see if it is cooked). So this effectively reduces the problem to a NP-class problem. This shows the power of recursive functions and functional formulations of the lambda calculus. The power of neural networks in modelling data comes precisely from this iterative functional formulation.

The Cantor set is a simple example of a computable fractal. However, there are more complex (pun unintended) fractals that are not even computable. Penrose [5] conjectured that some Mandelbrot sets are not computable, and this has been confirmed [6]. However, here again, these Mandelbrot fractals are formulated in terms of complex functions, another example of iterative functional formulation, which in this case is not even computable in terms of state machines.

Finally, we would like to mention that the power of recursion was considered in the early 1960s by Noam Chomsky who quoted the famous phrase of Wilhelm von Humboldt, "infinite by finite means" and later on by Douglas Hofstadter in his popular work [7], where he also makes interesting references to Metafont.

References

[1] Maxima (1982). A Computer Algebra System. `maxima.sourceforge.net`

[2] Aristotle (350 BCE), Zeno's paradox, translated by R.P. Hardie and R.K. Gaye. `classics.mit.edu/Aristotle/physics.6.vi.html`

[3] Béziau, J.-Y. (1970). Future of Paraconsistency Logic. `wwwa.unine.ch/unilog/jyb/future-pl.pdf`

[4] New Directions in Paraconsistent Logic, 5th WCP (2014) Kolkata, India, February 2014, Béziau, J.-Y., Chakraborty, M., and Dutta, S., eds.. Springer-Verlag.

[5] Penrose, R. (1989). The Emperor's New Mind. Concerning Computers, Minds and The Laws of Physics. Oxford University Press, New York.

[6] Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull. Amer. Math. Soc., 21(1):1–46.

[7] Hofstadter, D. (1979). Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books.

◇ S. K. Venkatesan
TNQ Technologies, Chennai, India
`skvenkat (at) tnqsoftware dot co dot in`

TUG 2018 abstracts

Editor's note: Videos are available for nearly all of the talks; links and other information at `https://tug.org/tug2018/program.html`.

– – ∗ – –

Doris Behrendt
The General Data Protection Regulation (GDPR) in the European Union

On 25 May 2018 the GDPR was applied in the EU. In my position as treasurer of the German TEX user group DANTE e.V. I studied this regulation from the DANTE perspective and will talk about some aspects of this regulation, which are concerning us.

As some of you probably know, a lot of Europeans — including myself — are somewhat delicate about data processing and privacy. While the industry complains about the GDPR being a monster of bureaucracy, there are also some quite interesting legal bearings that come with it, e.g., it will also apply "to the processing of personal data of data subjects who are in the Union by a controller or processor not established in the Union, where the processing activities are related to ... the offering of ... services, irrespective of whether a payment of the data subject is required, to such data subjects in the Union ...".

This should be interesting especially to companies that are not based in the EU but are handling data of EU citizens, and by GDPR Article 83 (5) not complying could become expensive: "Infringements ... shall ... be subject to administrative fines up to 20,000,000 EUR, or in the case of an undertaking, up to 4% of the total worldwide annual turnover of the preceding financial year, whichever is higher ...".

You can imagine that this could become very interesting when the next Facebook or similar data scandal comes up.

S. Coriasco, D. Ahmetovic, T. Armano, C. Bernareggi, M. Berra, A. Capietto, N. Murru, A. Ruighi, E. Taranto
An automated method based on LATEX for the realization of accessible PDF documents containing formulae

Mathematical formulae contained in PDF documents generated using LATEX are usually not accessible with assistive technologies for visually impaired people, such as screen readers and braille displays.

To address this issue, we developed *Axessibility*, a LATEX package which allows creation of PDF documents in which the formulae can be read by these assistive technologies. *Axessibility* automatically generates hidden comments inside PDF documents corresponding to each formula (by means of the `/ActualText` PDF attribute). This actual text contains the LATEX code of the formula, and it is read by screen readers (JAWS, NVDA and VoiceOver). Moreover, we have created NVDA and JAWS dictionaries (in English and in Italian) that provide natural language reading for users that do not know LATEX.

While this package enables accessibility of mathematical formulae contained in PDF documents, it does not generate PDF/UA compatible documents.

Joachim Heinze
The unchanged changing world of mathematical publishing

1. A very short overview of the history of mathematical publishing with some Springer examples is given. *Numerische Mathematik* was the first of all SpringerNature journals ever, over all disciplines, to go online in 1994.

2. The change of the world of publishing: generating (scientists), composing (publishers and scientists) and disseminating (librarians and publishers) mathematical content in electronically form. TEX and "online visibility" are the buzzwords here.

3. Open access for all mathematical content? "New" initiatives like "Overlay Journals", based on arXiv, are briefly discussed, as well as the more recent Sci-Hub and ResearchGate initiatives.

4. Keep track of what has been published and cited. MathSciNet and zbMATH, the two big math review journals, in comparison to other initiatives, like Google Scholar, Scopus, and Web of Science. A new initative from China? MathSciDoc.

5. Recent developments in the dissemination of scientific information are discussed. Social media (Scholarly Collaboration Networks (SCN)) in scientific communication and some new initiatives such as "Sharedit" and "SciGraph" are briefly reflected upon. Artificial intelligence and some hope for the future will close the presentation.

Tom Hejda
`yoin` — Yet another package for automation of journal typesetting

A new LATEX package will be presented that allows combining journal, conference and similar papers into issues. The most important premises the package are built upon are (1) the papers themselves are independent documents to the extent that even different compilers can be used for different papers, and (2) the papers' page numbering is automated and there are tools for communicating metadata between the whole issue and the papers.

Please note that a preliminary version of the package will be presented and help from the community will very likely be sought at the conference.

Mico Loretan
Selective ligature suppression with the `selnolig` *package*
TEX has long provided straightforward methods for creating typographic ligatures. Until recently, though, suppressing inappropriate ligatures selectively could only be achieved by applying mark-up by hand to a document. `selnolig`, a LuaLATEX package, provides the machinery to perform selective ligature suppression in an automated way that requires minimal user involvement. The package also provides sets of ligature suppression rules for English and German language documents. The talk provides an overview of the package's design philosophy and main features, discusses some of its current limitations, and gives the outlook for further developments.

Frank Mittelbach
A quarter century of `doc`
In this talk I will re-examine my poor attempts at Literate Programming and how they have shaped (for better or worse) the LATEX world in the past decades.

It's about time to rethink some of the concepts invented back then — but can we still evolve?

Ross Moore
Authoring accessible 'Tagged PDF' documents using LATEX
Several ISO standards have emerged for what should be contained in PDF documents, to support applications such as 'archivability' (PDF/A) and 'accessibility' (PDF/UA). These involve the concept of 'tagging', both of content and structure, so that smart reader/browser-like software can adjust the view presented to a human reader, perhaps afflicted with some physical disability. In this talk we will look at a range of documents which are fully conformant with these modern standards, mostly containing at least some mathematical content, created directly in LATEX. The examples are available on the author's website, `web.science.mq.edu.au/~ross/TaggedPDF`.

The desirability of producing documents this way will be discussed, along with aspects of how much extra work is required of authors. Also on the above website, and published elsewhere in this issue (pp. 131–135), is a 'five-year plan' on how to modify the production of LATEX-based scientific publications to adopt such methods. This will involve cooperation between academic publishers and a TUG working group.

[Editor's note: Since the talk worked mostly from examples, showing non-printing aspects of what can be stored in, and extracted from PDF files, the printed description is not entirely sufficient; see the video at `youtube.com/watch?v=mPBtkCsChJw`.]

Eduardo Ochs
Dednat6: An extensible (semi-)preprocessor for LuaLATEX that understands diagrams in ASCII art
(LA)TEX treats lines starting with `%` as comments, and ignores them. This means that we can put anything we want in these `%` lines, even code to be processed by other programs besides TEX.

In this talk we describe a "semi-preprocessor", called `dednat6`, that makes blocks of lines starting with `%L` be executed as Lua code, treats blocks of lines starting with `%:` as 2D representations of derivation trees, and treats blocks of lines starting with `%D` as diagrams in which a 2D representation specifies where the nodes are to be placed and a stack-based language inspired by Forth is used to connect these nodes with arrows.

A predecessor of `dednat6`, called `dednat4`, was a preprocessor in the more usual sense: running "`dednat4.lua foo.tex`" on a shell would convert the trees and diagrams in `%:` and `%D`-blocks in `foo.tex` to `\def`s that LATEX can understand, and would put these `\def`s in a file `foo.dnt`. Then in `foo.tex` we put an `\input "foo.dnt"` to load those definitions. Dednat6 does something almost equivalent to that, but using LuaLATEX to avoid the needs for an external preprocessor and for an auxiliary `.dnt` file. Here is how; the workflow is unusual, so let's see it in detail.

Put a line
```
\directlua{dofile("loaddednat6.lua")}
```
in a file `bar.tex`. When we run "`lualatex bar.tex`" that line loads the `dednat6` library, initializes the global variable `tf` in the Lua interpreter with a `TexFile` object, and sets `tf.nline=1` to indicate that nothing in `bar.tex` has been processed with Dednat6 yet.

A (low-level) command like
```
\directlua{processlines(200, 300)}
```
in `bar.tex` would "process the lines 200 to 300 in `bar.tex` with `dednat6`", which means to take all the blocks of `%L`-lines, `%:`-lines, and `%D`-lines between the lines 200 to 300 in `bar.tex`, run them in the necessary interpreters, and then send the resulting LATEX code — usually `\def`s — to the `latex` interpreter.

The high-level macro `\pu` runs
```
\directlua(processuntil{tex.inputlineno})
```
which runs `processlines` on the source lines between `tf.nline=1` and the line where the current `\pu` is, and advances `tf.nline`. That is, it processes

with `dednat6` the lines in the current file between the previous `\pu` and the current one.

The strings `%L`, `%:`, and `%D` are called "heads" in `dednat6`, and it's easy to add support for new heads; this can even be done in a `%L` block.

With `dednat4`, all the `\def`s had to be loaded at once; in `dednat6` idioms like `{\pu ...}`, `$\pu ...$`, and `$$\pu ...$$` can be used to make the `\def`s between the last `\pu` and the current one be local.

Boris Veytsman
Stubborn leaders six years later
After six years the journal *Res Philosophica* changed the style of its table of contents. The new design requires the dotted line with the page number to follow the last line of the article title rather than the first one. The old design was described in a *TUGboat* article (33:3, pp. 316–318, 2012, `tug.org/TUGboat/tb33-3/tb105veytsman-leaders.pdf`).

We use this occasion to revisit the old code, discuss the new one and the fact that deceptively similar designs require completely different code.

Boris Veytsman
R+knitr+LaTeX workshop
The work of a research scientist involves keeping daily notebooks. Such a working notebook is a document with text, equations, calculations, figures, tables, code snippets which reflects the current state of the lab research. This workshop teaches how to maintain such notebooks in a TeX/R environment.

Prerequisites: please install the following on your computer — a TeX distribution (preferably either TeX Live or MiKTeX), R, with packages knitr, tikzDevice and Hmisc. For a front end, we can use either Rstudio or Emacs+AUCTeX+ESS.

After you have installed R, you can install knitr, tikzDevice and Hmisc using the R packaging system. Also, Vincent Goulet has constructed convenient Emacs distributions for Windows and Mac systems which include ESS+AUCTeX, available at `vigou3.github.io/emacs-modified-windows`.

Joseph Wright
Fly me to the moon: (La)TeX testing (and more) using Lua
Testing has been important to the LaTeX team since its inception, and over the years a sophisticated set of test files have been created for the kernel. Methods for running the tests have varied over the past quarter-century, following changes in the way the team work.

In recent years, the availability of Lua as a scripting language in all TeX systems has meant it has become the natural choice to support this work. With this as a driver, the team have developed the `l3build` package (`ctan.org/pkg/l3build`) for running tests automatically. Building on the core work, `l3build` has grown to provide a powerful approach to releasing packages (and the LaTeX kernel) reliably.

Here, I'll look at the background of our testing approach, before showing how and why Lua works for us here.

***MAPS* 48 (2018)**

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

Karl Berry, TeX Live Guide; pp. 3–45
[See `https://tug.org/texlive/doc.html`.]

Hans Hagen, Executing TeX; pp. 46–50
[Published in *TUGboat* 39:1.]

Hans Hagen, Variable fonts; pp. 51–58
[Published in *TUGboat* 38:2.]

Bogusław Jackowski, Piotr Pianowski, Piotr Strzelczyk, TeX Gyre text fonts revisited; pp. 59–65
[See DTK abstracts.]

Siep Kroonenberg, TLaunch, the TeX Live Launcher; pp. 66–69
[Published in *TUGboat* 38:2.]

Norbert Preining, `updmap` and `fmtutil` — past and future changes; pp. 70–76
[Published in *TUGboat* 38:2.]

NTG, Privacybeleid; pp. 77–80

Die TEXnische Komödie 2–3/2018

Die TEXnische Komödie is the journal of DANTE e.V., the German-language TEX user group (`dante.de`). (Non-technical items are omitted.)

Die TEXnische Komödie 2/2018

Jörg Bergs, Abseits der Wissenschaft: Setzen eines konzeptionellen Bildbands [Beyond science: Typesetting an illustrated book]; pp. 29–37

As the owner of one of the few analogue photo labs still in existence, I am using LATEX for our complete correspondence workflow and for our technical documentation, using standard packages such as KOMA-Script. For pure DTP tasks, however, we use QuarkXPress, as I consider LATEX to be suboptimal for such tasks. But I have now started to create an illustrated book using LATEX.

Thomas Hilarius Meyer, Liste der nicht benutzten Literatur ausgeben [Printing uncited references]; pp. 37–39

I suppose I am not the only one who has issues with the question of which references from the bib file have actually been used and which not. In this article I show a way of finding out.

Dominik Wagenführ, Vorstellung einer deutschsprachigen Bewerbungsvorlage [A German template for job applications]; pp. 40–53

Almost everyone, sooner or later, has to apply for a job. In this article I present my LATEX-based application template which is directed mainly to German TEX users. Components of the application are given in a single format making the whole application look consistent.

Gerd Neugebauer, CTAN quiz; pp. 53–55

Published in *TUGboat* 39:1.

Die TEXnische Komödie 3/2018

Bogusław Jackowski, Piotr Pianowski, Piotr Strzelczyk, TEX Gyre text fonts revisited; pp. 11–20

The collection of the TEX Gyre (TG for short) family of text fonts was first released by the GUST e-foundry in 2006–2009. Having finished this task, the GUST e-foundry team started to work on the math companion (in OpenType, OTF, format) for the TG text fonts. Work on the math companion was finished two years ago. It resulted in the broadening of the repertoire of glyphs that could be used not only in math mode but also in text mode in technical documents. Hans Hagen, indefatigably coming up with interesting ideas, proposed migrating the relevant glyphs to the text TG fonts. Needless to say, we seized on Hans's suggestion. The first step was to decide which glyphs are to be migrated (and/or improved). Obviously, the list of candidates grew and grew. All in all, about 1000 glyphs were designated to be added, mostly geometrical and math symbols. A math companion, so far, was provided only for serif fonts, thus the consistent enhancement of the repertoire of the sans-serif fonts was a working test for our fonts generator. We started with two fonts — the serif TG Pagella and the sans-serif TG Adventor. The results were satisfying. Now we are ready for the next step: to enhance similarly the rest of the TG family (TG Chorus which is hardly suitable for technical texts, needs an individualized approach). We believe, however, that we're over the hump. Below, we describe the most difficult and thus most interesting (to us) aspects of this stage of the TG project.

[To be published in the next issue of *TUGboat*.]

Robert Winkler, Akademisches Schreiben mit Markdown und Pandoc Scholar [Academic writing using Markdown and Pandoc Scholar]; pp. 21–28

In academia it is often required to create documents for different output formats. Besides scientific manuscripts there are webpages, CVs or reports to be created. Markdown is a simple text-based format that allows easy conversion with the help of Pandoc into different output formats, e.g., PDF, LATEX, OpenOffice XML, HTML, or EPUB. Pandoc Scholar is an extension for Pandoc, enriching it with semantic web annotations, e.g., Citation Typing Ontologies. A practical application is shown with the example of this article.

Christine Römer, Schnell und direkt von `.tex` zu `.epub` mit `tex4ebook` [Fast and direct from `.tex` to `.epub` using `tex4ebook`]; pp. 29–34

`tex4ebook` has been discussed already in DTK issue 4/2015; however we address it again in this article. Since the previous article mainly addressed issues in the output of mathematical content we now focus on its advantages regarding text output.

Editors, Im Netz gefunden [Found in the world wide web]; pp. 35–36

- Christian Justen: have `xcolor` typeset only the first line in a different color.
- Brian Dunn: centering a "framed box" between two columns.

[Received from Herbert Voß.]

Eutypon 38–39, October 2017

Eutypon is the journal of the Greek TEX Friends (`http://www.eutypon.gr`).

Apostolos Syropoulos, Diagrams with pgfplots; pp. 1–12

Most authors of science papers and books need to create plots and diagrams to display scientific data. The problem of course is which tool to use for this task. In general, people prefer to use a tool from a suite of tools which is familiar to them. The package pgfplots is the ideal package for people who use LATEX to prepare their documents. This paper describes the basic use of the package, i.e., how to create a simple diagram and how to place points and text on it. In addition, it presents some specialized features and how one can create bar charts and pie charts. (*Article in Greek with English abstract.*)

Dimitrios Filippou, Typesetting elements and other... chemicals; pp. 13–33

The International Union of Pure and Applied Chemistry (IUPAC) has produced several guidelines for the nomenclature of chemicals, and also for the appearance of chemical elements, compounds, physical/chemical variables, units, etc. TEX was made for typesetting mathematical formulæ. Nonetheless, with some effort, TEX's machine can be tweaked for typesetting chemical formulæ as well. Packages like chemmacros, mhchem, chemfig and xymtex, give with LATEX (and even with plain TEX) excellent results for documents with chemical symbols. (*Article in Greek with English abstract.*)

Dimitrios Filippou, TEXniques: Slanted black math symbols and other issues of unicode-math; pp. 35–37

In this regular column, it is shown how to obtain slanted (or italic) black math symbols with XELATEX and unicode-math, as well as some particular issues in the use of the same package. (*Article in Greek.*)

Dimitrios Filippou, Book presentations; pp. 39–40

The following books are presented:
(a) Keith Houston, *Shady Characters: Ampersands, Interrobangs and Other Typographical Curiosities*, 2nd edition, Penguin, UK 2015; and
(b) George Grätzer, *More Math into LATEX*, 5th edition, Springer, Cham, Switzerland.
(*Article in Greek.*)

[Received from Dimitrios Filippou and Apostolos Syropoulos.]

Don Knuth awarded Trotter Prize

Bart Childs, Rick Furuta

The Trotter Prize & Endowed Lecture Series on Information, Complexity and Inference is presented by the College of Science in collaboration with the Dwight Look College of Engineering at Texas A&M University (`science.tamu.edu`). It seeks to illuminate connections between science and religion.

Donald Knuth, Professor Emeritus of The Art of Computer Programming at Stanford University, and Michael Duff, Emeritus Professor of Theoretical Physics at Imperial College London, were awarded the prize April 17, 2018. Professor Knuth's lecture was entitled *Translating the Bible into Music*. Professor Duff's lecture was entitled *The Best of All Possible Worlds*.

A lively Q&A session was held the next day with the awardees in the Hawking Auditorium with titles "All Questions Answered" and "The Universe and Other General Questions", respectively.

Don's talk was about his composing *Fantasia Apocalyptica*. Don's home page (`www-cs-faculty.stanford.edu/~knuth`) is a good source for additional information about this work as well as his 80th birthday party.

Richard Furuta and I hosted the Knuth family for a TEXas barbecue at a well-known local restaurant, C&J's Barbecue. A good time was had by all. We also learned that Don "rarely passes up a chance to have peach cobbler." Don was accompanied by Jill and her sister and family. Doug Hensley, Professor of Mathematics, and my friend Barbara Schwartz also joined us. Barbara took the picture.

Rick and I were TUG officers 1985–89 and served as site coordinators for distributions of TEX systems.

◇ Bart Childs
Rick Furuta
Texas A&M University
College Station, TeXas 77843, USA
`bart , furuta (at) tamu.edu`

Hyphenation exception log

Barbara Beeton

This is a brief update of the list of words that TEX fails to hyphenate properly for U.S. English. The full list last appeared in *TUGboat* 16:1, starting on page 12, with periodic updates, most recently in *TUGboat* 39:1, p. 7.

A cumulative list appears in the CTAN collection, at `ctan.org/pkg/hyphenex`, or in TEX Live, in the file `tb0hyf` (`.tex` or `.pdf`). We are dispensing here with the details of how the list is organized; that information can be found in the previous update or on CTAN.

The heavy concentration of names in this update is due to the (un?)timely appearance of a 700-page book on a historical topic in the production queue at the AMS. Names often do not follow the pattern of ordinary English words, and are therefore not handled gracefully by the existing patterns. Worse still, they often appear in the first sentence of a paragraph, introducing the subject; and, once introduced, they are likely to occur again in the same work. Hence the desirability that they be dealt with once, in a local `\hyphenation` list.

British hyphenation

The patterns for British hyphenation — quite different from the U.S. patterns — were developed from a word list provided by the Oxford University Press based on their 1986 *Minidictionary of Spelling and Word Division*. As of 2014, the *New Oxford Spelling Dictionary* shows that many points of word division have changed. (The changes largely appear to be converging with U.S. practice.) The UK TEX Users Group has approached OUP to request access to the new word list, in order to update the patterns. There is apparently progress in this direction, at least in principle, although nothing definitive is known yet. Should the negotiations be successful, it will be reported here as soon as feasible.

The list — English words

`al-manac`	al-ma-nac
`al-manack`	al-ma-nack
`awarded`	awar-ded
`bankroll`	bank-roll
`courage`	cour-age
`coura-geous(ly)`	cou-ra-geous(-ly)
`dataflow`	data-flow
`dat-a-point`	data-point
`dataset`	data-set
`datatype`	data-type
`drosophila`	dro-soph-i-la
`ephemera`	ephem-era
`ephemeris(ides)`	ephem-eris(-i-des)
`equated`	equa-ted
`equidi-men-sional`	equi-di-men-sional
`method-ol-ogy(ies)`	meth-od-o-lo-gy(ies)
`method-olog-i-cal(ly)`	meth-od-o-logical(ly)
`nonar-chimedean`	non-ar-chi-me-dean
`philoso-pher`	phi-los-o-pher
`philoso-phies`	phi-los-o-phies
`plurisub-har-monic`	pluri-sub-har-monic
`polynya`	po-lyn-ya
`pol-y-semy`	poly-se-my
`potable`	po-ta-ble
`premise`	prem-ise
`prepo-si-tion`	prep-o-si-tion
`prepo-si-tional`	prep-o-si-tional
`semistable`	semi-sta-ble
`sto-plist`	stop-list
`transat-lantic`	trans-at-lan-tic
`transept`	tran-sept
`transpa-cific`	trans-pacific
`trapez-ium`	tra-pe-zium
`trape-zoid`	trap-e-zoid
`trape-zoidal`	trap-e-zoi-dal
`trigonom-e-try`	trig-o-nom-e-try
`trigono-met-ric`	trig-o-no-met-ric

Names and non-English words used in English text

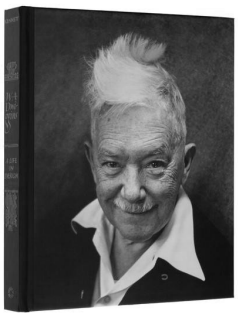
`Alexan-dria`	Alex-an-dria
`An-dalu-cia(n)`	An-da-lu-cia(n)
`An-dalu-sia(n)`	An-da-lu-sia(n)
`Bowditch`	Bow-ditch
`Brinkmann`	Brink-mann
`Canada`	Can-a-da
`Cana-dian`	Ca-na-di-an
`Chicago`	Chi-ca-go
`Franklin`	Frank-lin
`Ge-oge-bra`	Geo-gebra
`Hamil-ton(ian)`	Ham-il-ton(-nian)
`Haskell`	Has-kell
`Hedrick`	Hed-rick
`Hegelian`	Hegel-ian
`Hilbert`	Hil-bert
`Jeremiah`	Je-re-miah
`Laplace`	La-place
`Lester`	Les-ter
`Morawetz`	Mora-wetz
`Mordell`	Mor-dell
`Nicholas`	Nich-o-las
`Northamp-ton`	North-amp-ton
`Por-tuguese`	Por-tu-guese
`Richard`	Rich-ard
`Southamp-ton`	South-amp-ton
`Sylvester`	Syl-ves-ter
`Wilczyn-ski`	Wil-czyn-ski
`Wolfskehl`	Wolfs-kehl
`Woodrow`	Wood-row

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

Book review: *W. A. Dwiggins: A Life in Design*, by Bruce Kennett

Boris Veytsman

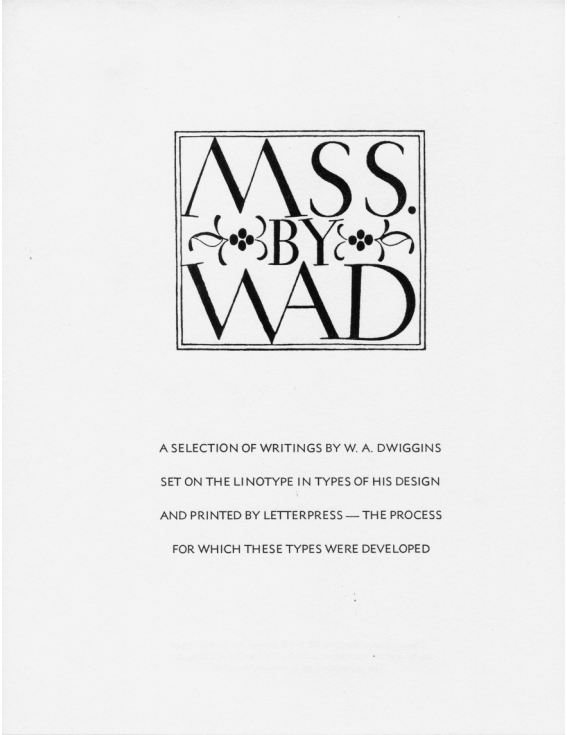
Bruce Kennett, *W. A. Dwiggins: A Life in Design*. Letterform Archive, San Francisco, 496pp., US$95, ISBN 978-0-9983180-0-4 (standard edition), US$350, ISBN 978-0-9983180-1-1 (deluxe edition). brucekennett.com/wa-dwiggins-a-life-in-design/

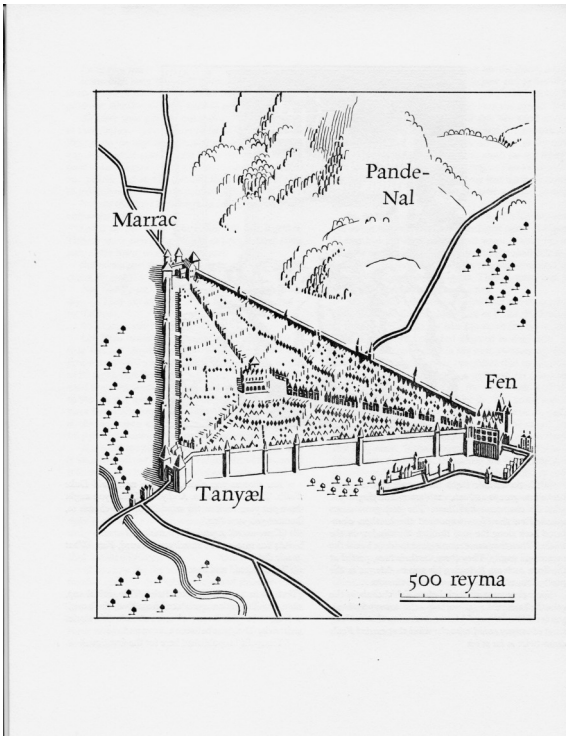


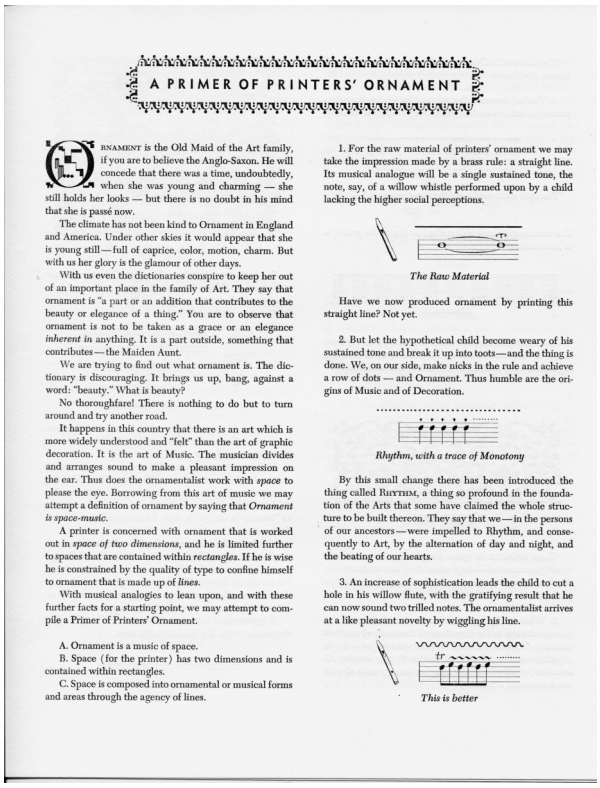
In 1923, when forty-three years old, William A. Dwiggins (or WAD to his many friends) was diagnosed with severe diabetes. His father died young due to diabetes, so Dwiggins understood he had just several years to live. He decided to wrap up his lucrative business as an advertising designer and printer, and to spend the rest of his life doing the work of love: book design and illustration.

Dwiggins was wrong: two years earlier insulin had been discovered, and his physician had access to this experimental drug. He lived for another thirty-three years, leaving a huge number of splendid books and illustrations, several fonts, as well as many essays, pamphlets, articles, plays and tales. He was incredibly prolific: at the peak of his creativity he designed 24 books per year. Furthermore, he made time for his many hobbies, including marionettes: WAD made important contributions to this ancient art, and his book about marionette making is still respected among practitioners. By the way, his expectation of "starvation" (as he wrote in a letter) was also wrong: Dwiggins quickly became widely recognized in the book world. Mergenthaler (the Linotype company) and Alfred Knopf had him on a retainer, and many publishers were happy to pay WAD for his work.

This book by Bruce Kennett is a fitting tribute to this master of American book design and typography. Kennett spent a decade and a half teaching about WAD, lecturing and writing. A talented designer himself, Kennett created a masterpiece of the book arts. His volume is typeset in LfA Aluminia, a digital revival of Dwiggins' Electra font by Jim Parkinson (Letterform Archive sells the set for $75). It has about 1,200 color illustrations with many reproductions of WAD's works, and a collection of his writings typeset in his fonts (the deluxe edition is accompanied by these samples printed on a letterpress; several are reproduced here).







A PRIMER OF PRINTERS' ORNAMENT

ORNAMENT is the Old Maid of the Art family, if you are to believe the Anglo-Saxon. He will concede that there was a time, undoubtedly, when she was young and charming — she still holds her looks — but there is no doubt in his mind that she is passé now.

The climate has not been kind to Ornament in England and America. Under other skies it would appear that she is young still — full of caprice, color, motion, charm. But with us her glory is the glamour of other days.

With us even the dictionaries conspire to keep her out of an important place in the family of Art. They say that ornament is "a part or an addition that contributes to the beauty or elegance of a thing." You are to observe that ornament is not to be taken as a grace or an elegance *inherent* in anything. It is a part outside, something that contributes — the Maiden Aunt.

We are trying to find out what ornament is. The dictionary is discouraging. It brings us up, bang, against a word: "beauty." What is beauty?

No thoroughfare! There is nothing to do but to turn around and try another road.

It happens in this country that there is an art which is more widely understood and "felt" than the art of graphic decoration. It is the art of Music. The musician divides and arranges sound to make a pleasant impression on the ear. Thus does the ornamentalist work with *space* to please the eye. Borrowing from this art of music we may attempt a definition of ornament by saying that *Ornament is space-music.*

A printer is concerned with ornament that is worked out in *space of two dimensions,* and he is limited further to spaces that are contained within *rectangles.* If he is wise he is constrained by the quality of type to confine himself to ornament that is made up of *lines.*

With musical analogies to lean upon, and with these further facts for a starting point, we may attempt to compile a Primer of Printers' Ornament.

A. Ornament is a music of space.

B. Space (for the printer) has two dimensions and is contained within rectangles.

C. Space is composed into ornamental or musical forms and areas through the agency of lines.

1. For the raw material of printers' ornament we may take the impression made by a brass rule: a straight line. Its musical analogue will be a single sustained tone, the note, say, of a willow whistle performed upon by a child lacking the higher social perceptions.

The Raw Material

Have we now produced ornament by printing this straight line? Not yet.

2. But let the hypothetical child become weary of his sustained tone and break it up into toots — and the thing is done. We, on our side, make nicks in the rule and achieve a row of dots — and Ornament. Thus humble are the origins of Music and of Decoration.

Rhythm, with a trace of Monotony

By this small change there has been introduced the thing called RHYTHM, a thing so profound in the foundation of the Arts that some have claimed the whole structure to be built thereon. They say that we — in the persons of our ancestors — were impelled to Rhythm, and consequently to Art, by the alternation of day and night, and the beating of our hearts.

3. An increase of sophistication leads the child to cut a hole in his willow flute, with the gratifying result that he can now sound two trilled notes. The ornamentalist arrives at a like pleasant novelty by wiggling his line.

This is better

The colophon reflects the love and the work that went into the book creation. I cannot help but reproduce it in full:

> *W. A. Dwiggins: A Life in Design* has been produced in standard and deluxe editions, a total of 2,200 copies. Bruce Kennett designed the book and composed the text with LfA Aluminia, a revival of WAD's Electra types that Jim Parkinson created expressly for this project. Photographs made at Letterform Archive were captured on a Phase One digital-camera system with raking light; the remaining photographs were made with Nikon digital SLR cameras, also with raking light. The paper for the body of the book is Sappi Opus Matte Text; the endpapers are Strathmore Writing Text, Ultimate White. The standard edition's cover is printed on Sterling Ultra Text. The boards of the deluxe edition are covered with Neenah Environment Raw Text, Concrete, printed with a reproduction of Dwiggins's design for Lakeside Press edition of Edgar Allan Poe's *Tales;* its spine is made from Saderra leather and Taylor Box, of Warren, Rhode Island, built its slipcases. Penmor Lithographers, of Lewiston, Maine, provided all the presswork, using high-definition stochastic screens, and Acme Bookbinding/HF Group, of Charlestown, Massachusetts, performed the smyth-sewing and binding. Production details for the letterpress portfolio that accompanies the deluxe edition are described on page 453.

Eleven chapters of the book correspond to eleven addresses where Dwiggins studio was located. This periodization of WAD's life is apt: each move corresponded to a change in the artist's life, interests or the line of work. There are three separate unnumbered chapters about Dwiggins' type design, marionettes and writings. It does not often happen that a lavishly illustrated book is also informative and well written. This book is an exception to this rule. The text is superb: Kennett does not try to overwhelm the reader with the arcana of typography, but he also does not oversimplify his writing. Instead Kennett's style is direct, honest and full of love — the same as the style of Dwiggins himself. Moreover, the book satisfies the strict requirements of an academic monograph: it has sources and notes (14 pages of small print), several pages of bibliography and suggested reading, and a useful index — a rarity nowadays.

The book is a great work of love and art. I consider it one of the best books of the year — or many years.

◇ Boris Veytsman
Systems Biology School and Center for Simulation and Modeling, George Mason University, Fairfax, VA 22030
borisv (at) lk dot net
http://borisv.lk.net

The Treasure Chest

This is a selection of the new packages posted to CTAN (`ctan.org`) from April–August 2018, 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 `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. See also `ctan.org/topic`. Comments are welcome, as always.

◇ Karl Berry
`tugboat (at) tug dot org`

biblio

`bath-bst` in `biblio/bibtex/contrib`
Univ. of Bath reference style for BibTEX.

fonts

`casiofont` in `fonts`
Support for the Casio ClassWiz font.
`dsserif` in `fonts`
Double-struck serifed font for math.
`notocjksc` in `fonts`
Noto CJK (Source Han) fonts for Simplified Chinese.
* `stickstoo` in `fonts`
STIX2 fonts with enhanced LATEX support.
`stoneipa` in `fonts`
Support for Stone Sans Phonetic.

graphics

`ketcindy` in `graphics`
Create graphics for TEX with Cinderella.
`milsymb` in `graphics/pgf/contrib`
Ti*k*Z-based drawing of military symbols.
`penrose` in `graphics/pgf/contrib`
Ti*k*Z library for producing Penrose tilings.
`postage` in `graphics`
Stamp letters with Deutsche Post online service.
`pst-contourplot` in `graphics/pstricks/contrib`
Draw implicit functions using "marching squares" algorithm.
`tikz-nef` in `graphics/pgf/contrib`
Draw networks using Neural Engineering Framework methods.
`tikz-network` in `graphics/pgf/contrib`
Draw generalized networks with Ti*k*Z.
`tikzmarmots` in `graphics/pgf/contrib`
Draw marmots with Ti*k*Z.

language/japanese

`endnotesj` in `language/japanese`
Japanese-style endnotes.

macros/generic

`modulus` in `macros/generic`
Non-destructive modulus and integer quotient.
* `texdate` in `macros/generic`
Date printing, formatting, manipulation.

macros/latex/contrib

`axessibility` in `macros/latex/contrib`
Assistive technology support for math in PDF.
`cellprops` in `macros/latex/contrib`
Accept CSS-like selectors for `tabular`, `array`, etc.
`clrdblpg` in `macros/latex/contrib`
Control `pagestyle` on `\cleardoublepage` pages.
`clrstrip` in `macros/latex/contrib`
Place contents in a full-width color strip.
`competences` in `macros/latex/contrib`
Track skills assessed by classroom tests.
`digicap-pro` in `macros/latex/contrib`
Captions for digital photos, using Adobe Distiller.
`ecothesis` in `macros/latex/contrib`
Theses at the Univ. Federal de Viçosa, Brazil.
`erw-l3` in `macros/latex/contrib`
Utilities built on `expl3`.
`etsvthor` in `macros/latex/contrib`
Abbreviations for study group at Eindhoven Univ.
`gatherenum` in `macros/latex/contrib`
Displayed enumerations via `align*` and `enumerate`.
`guitartabs` in `macros/latex/contrib`
Draw guitar tablatures.
`hyperbar` in `macros/latex/contrib`
Interactive barcode fields in PDF forms.
`includernw` in `macros/latex/contrib`
Include `.Rnw` inside `.tex`, invoking `knitr`.
`inline-images` in `macros/latex/contrib`
Include inline images in base64.
`jnuexam` in `macros/latex/contrib`
Exam class for Jinan University.
`libertinus-otf` in `macros/latex/contrib`
Support for Libertinus OpenType fonts.
`manyind` in `macros/latex/contrib`
Support multiple indexes.
`mathfont` in `macros/latex/contrib`
Use TrueType and OpenType fonts in math mode; compatible with XETEX and LuaTEX.
`musikui` in `macros/latex/contrib`
Creating "arithmetical restoration" puzzles.
`nexus-otf` in `macros/latex/contrib`
Support for the Nexus OpenType fonts.

onedow in macros/latex/contrib
Diagrams for the bridge card game.
opacity-pro in macros/latex/contrib
Transparency and blend modes with Adobe Distiller.
padcount in macros/latex/contrib
Pad numbers (token lists) with arbitrary characters.
pdfpc-movie in macros/latex/contrib
Embed movies for the PDF Presenter Console.
pdfoverlay in macros/latex/contrib
Overlay text on an existing PDF file.
powerdot-tuliplab in macros/latex/contrib
Powerdot theme for the TULIP Lab.
qrcstamps in macros/latex/contrib
QR codes as dynamic stamp annotations.
statistics in macros/latex/contrib
Compute and typeset statistics tables and graphics.
tagpdf in macros/latex/contrib
Experimental tagging using pdfLaTeX and LuaLaTeX.
tlc-article in macros/latex/contrib
Document class for formal documents.
topletter in macros/latex/contrib
Letter class for the Politecnico di Torino.
ucsmonograph in macros/latex/contrib
Documents for Univ. of Caxias do Sul, Brazil.
worksheet in macros/latex/contrib
Worksheet creation.
xbmks in macros/latex/contrib
Create cross-document PDF bookmark tree.
xfakebold in macros/latex/contrib
Fake bold for outline fonts.

macros/latex/contrib/beamer-contrib/themes

beamertheme-focus in m/l/c/b-c/themes
Clean and minimalist beamer theme.
beamertheme-npbt in m/l/c/b-c/themes
Collection of themes relating to FOM Hochschule.

macros/latex/contrib/biblatex-contrib

biblatex-math in m/l/c/biblatex-contrib
Univ. of Bath reference style for BibLaTeX.
biblatex-socialscienceshuberlin in m/l/c/b-c
Social sciences at HU Berlin.

macros/luatex/generic

kanaparser in macros/luatex/generic
Kana parser for LuaTeX supporting ASCII input.
luavlna in macros/luatex/generic
Prevent line breaks after nonsyllabic prepositions and single-letter conjunctions.

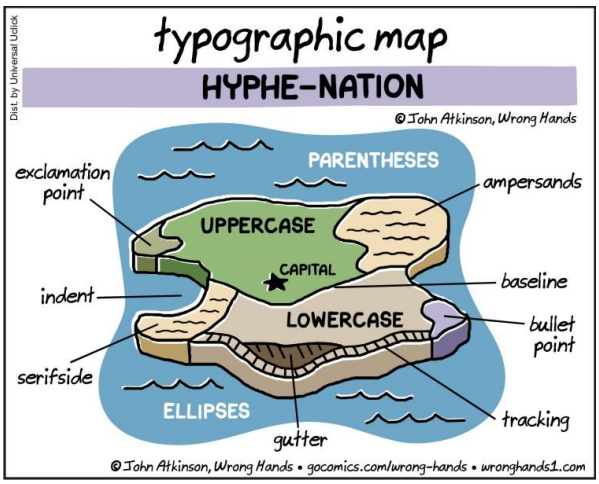
macros/luatex/latex

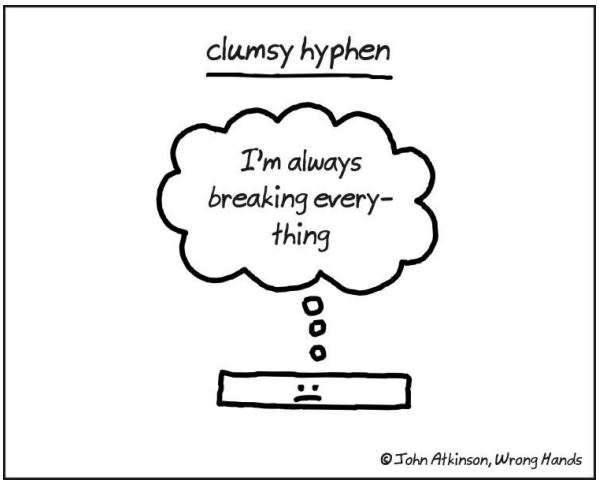
bezierplot in macros/luatex/latex
Approximate smooth function graphs with splines.
gurps in macros/luatex/latex
Generic Universal Role Playing System materials.
lualatex-truncate in macros/luatex/latex
Hyphenation fixes for LuaLaTeX.

macros/luatex/latex/plantuml
Rendering diagrams specified in PlantUML.
typewriter in macros/luatex/latex
Typeset with randomly variable monospace font.
wallcalendar in macros/luatex/latex
Wall calendar layouts and internationalization.

macros/xetex/latex

businesscard-qrcode in macros/xetex/latex
Business cards with QR codes.
cqubeamer in macros/xetex/latex
Beamer template for Chongqing Univ.
kurdishlipsum in macros/xetex/latex
Generic text examples in Kurdish.
na-border in macros/xetex/latex
CornPop border font support for Arabic and French.





Comics by John Atkinson (http://wronghands1.com).

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 there.

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
+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. 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.

Dominici, Massimiliano
Email: `info (at) typotexnica.it`
Web: `http://www.typotexnica.it`
Our skills: layout of books, journals, articles; creation of LaTeX classes and packages; graphic design; conversion between different formats of documents.

We offer our services (related to publishing in Mathematics, Physics and Humanities) for documents in Italian, English, or French. Let us know the work plan and details; we will find a customized solution. Please check our website and/or send us email for further details.

Hendrickson, Amy
57 Longwood Ave. #8
Brookline, MA 02446
+1 617-738-8029
Email: `amyh (at) texnology.com`
Web: `http://texnology.com`
LaTeX Macro Writing: Packages for print and e-publishing; Sophisticated documentation for users. Book and journal packages distributed on-line to thousands of authors.

More than 30 years' experience, for major publishing companies, scientific organizations, leading universities, and international clients.

Graphic design; Software documentation; LaTeX used for Data Visualization, and automated report generation; e-publishing, design and implementation; Innovation to match your needs and ideas.

LaTeX training, customized to your needs, on-site — have taught classes widely in the US, and in the Netherlands and Sweden.

See the TeXnology website for examples. Call or send email: I'll be glad to discuss your project with you.

Latchman, David
2005 Eye St. Suite #6
Bakersfield, CA 93301
+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.

Sofka, Michael
8 Providence St.
Albany, NY 12203
+1 518 331-3457
Email: `michael.sofka (at) gmail.com`
Personalized, professional TeX and LaTeX consulting and programming services.

I offer 30 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in TeX and LaTeX: Automated document

Sofka, Michael (cont'd)
conversion; Programming in Perl, C, C++ and other languages; Writing and customizing macro packages in TEX or LATEX, `knitr`.

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.

TEXtnik
Spain
Email: `textnik.typesetting (at) gmail.com`
Do you need personalised LATEX class or package creation? Maybe help to finalise your current typesetting project? Any problems compiling your current files or converting from other formats to LATEX? We offer +15 years of experience as advanced LATEX user and programmer. Our experience with other programming languages (scripting, Python and others) allows building systems for automatic typesetting, integration with databases, ... We can manage scientific projects (Physics, Mathematics, ...) in languages such as Spanish, English, German and Basque.

Veytsman, Boris
132 Warbler Ln.
Brisbane, CA 94005
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`
TEX and LATEX consulting, training, typesetting and seminars. Integration with databases, automated

Veytsman, Boris (cont'd)
document preparation, custom LATEX packages, conversions (Word, OpenOffice etc.) and much more.

I have about two decades of experience in TEX and three decades of experience in teaching & training. I have authored more than forty packages on CTAN as well as Perl packages on CPAN and R packages on CRAN, published papers in TEX-related journals, and conducted several workshops on TEX and related subjects. Among my customers have been Google, US Treasury, FAO UN, Israel Journal of Mathematics, Annals of Mathematics, Res Philosophica, Philosophers' Imprint, No Starch Press, US Army Corps of Engineers, ACM, and many others.

We recently expanded our staff and operations to provide copy-editing, cleaning and troubleshooting of TEX manuscripts as well as typesetting of books, papers & journals, including multilingual copy with non-Latin scripts, and more.

Webley, Jonathan
Flat 11, 10 Mavisbank Gardens
Glasgow, G1 1HG, UK
07914344479
Email: `jonathan.webley (at) gmail.com`
I'm a proofreader, copy-editor, and LATEX typesetter. 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, and converting a Word document to LATEX. I've done hundreds of papers for journals over the years. Samples of work can be supplied on request.

TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:
`tug.org/instmem.html`
Thanks to all for their support!

American Mathematical Society, *Providence*, *Rhode Island*

Association for Computing Machinery, *New York*, *New York*

Aware Software, *Newark*, *Delaware*

Center for Computing Sciences, *Bowie*, *Maryland*

CSTUG, *Praha*, *Czech Republic*

Harris Space and Intelligence Systems, *Melbourne*, *Floida*

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*

Nagwa Limited, *Windsor*, *UK*

New York University, Academic Computing Facility, *New York*, *New York*

Overleaf, *London*, *UK*

Springer-Verlag Heidelberg, *Heidelberg*, *Germany*

StackExchange, *New York City*, *New York*

Stockholm University, Department of Mathematics, *Stockholm*, *Sweden*

TEXFolio, *Trivandrum*, *India*

TNQ, *Chennai*, *India*

University College Cork, Computer Centre, *Cork*, *Ireland*

Université Laval, *Ste-Foy*, *Québec*, *Canada*

University of Ontario, Institute of Technology, *Oshawa*, *Ontario*, *Canada*

University of Oslo, Institute of Informatics, *Blindern*, *Oslo*, *Norway*

VTEX UAB, *Vilnius*, *Lithuania*

2019 TEX Users Group election

Karl Berry
for the Elections Committee

The positions of TUG President and five members of the Board of Directors will be open as of the 2019 Annual Meeting, which we expect to be held in July or August 2019 in Palo Alto, California, USA.

The terms of these individuals will expire in 2019: Barbara Beeton, Susan DeMeritt, Michael Doob, Cheryl Ponchin, Norbert Preining.

Continuing directors, with terms ending in 2021: Karl Berry, Johannes Braams, Kaja Christiansen, Taco Hoekwater, Klaus Höppner, Frank Mittelbach, Ross Moore, Arthur Reutenauer, Will Robertson, Herbert Voß.

The election to choose the new President and Board members will be held in early Spring of 2019. Nominations for these openings are now invited. A nomination form is on this page; forms may also be obtained from the TUG office or via `tug.org/election`.

The Bylaws provide that "Any member may be nominated for election to the office of TUG President/ to the Board by submitting a nomination petition in accordance with the TUG Election Procedures. Election ... shall be by ... ballot of the entire membership, carried out in accordance with those same Procedures."

The name of any member may be placed in nomination for election to one of the open offices by submission of a petition, signed by two other members in good standing, to the TUG office; the petition and all signatures must be received by the deadline stated below. A candidate's membership dues for 2019 must be paid before the nomination deadline. The term of President is two years, and the term of TUG Board member is four years.

Along with a nomination form, each candidate must supply a passport-size photograph, a short biography, and a statement of intent to be included with the ballot; the biography and statement of intent together may not exceed 400 words. The deadline for receipt of complete nomination forms and ballot information is

07:00 a.m. PST, 1 March 2019

at the TUG office in Portland, Oregon, USA. No exceptions will be made. Forms may be submitted by fax, or scanned and submitted by email to `office@tug.org`; receipt will be confirmed by email.

Information for obtaining ballot forms from the TUG website will be distributed by email to all members within 21 days after the close of nominations. It will be possible to vote electronically. Members preferring to receive a paper ballot may make arrangements by notifying the TUG office; see address on the form. Marked ballots must be received by the date noted on the ballots.

Ballots will be counted by a disinterested party not affiliated with the TUG organization. The results of the election should be available by mid-April, and will be announced in a future issue of *TUGboat* and through various TEX-related electronic media.

2019 TUG Election — Nomination Form

Only TUG members whose dues have been paid for 2019 will be eligible to participate in the election. The signatures of two (2) members in good standing at the time they sign the nomination form are required in addition to that of the nominee. **Type or print** names clearly, using the name by which you are known to TUG. Names that cannot be identified from the TUG membership records will not be accepted as valid.

The undersigned TUG members propose the nomination of:

Name of Nominee: _____

Signature: _____

Date: _____

for the position of (check one):

☐ **TUG President**

☐ **Member of the TUG Board of Directors**

for a term beginning with the 2019 Annual Meeting.

1. _____ (please print)

   _____ (signature) _____ (date)

2. _____ (please print)

   _____ (signature) _____ (date)

Return this nomination form to the TUG office via postal mail, fax, or scanned and sent by email. Nomination forms and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received at the TUG office in Portland, Oregon, USA, no later than

07:00 a.m. PST, 1 March 2019.

It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will late or incomplete applications be accepted.

Supplementary material may be sent separately from the form, and supporting signatures need not all appear on the same physical form.

- ☐ nomination form
- ☐ photograph
- ☐ biography/personal statement

TEX Users Group
Nominations for 2019 Election
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

(email: `office@tug.org`; fax: +1 815 301-3568)

Calendar

2018

Aug 28 – 31 — 18th ACM Symposium on Document Engineering, Halifax, Nova Scotia, Canada. www.doceng2018.org

Sep 2 – 8 — 12th International ConTEXt Meeting, "Unusual usage of ConTEXt", Prague–Sibřina, Czech Republic. meeting.contextgarden.net/2018

Sep 13 — Lecture: W. A. Dwiggins by Bruce Kennett, Museum of Printing, Haverhill, Massachusetts. museumofprinting.org

Sep 9 – 14 — XML Summer School, St Edmund Hall, Oxford University, Oxford, UK. xmlsummerschool.com

Sep 11 – 15 — Association Typographique Internationale (ATypI) annual conference, "Type legacies, honouring the heritage, designing type today", Antwerp, Belgium. www.atypi.org

Sep 15 — DANTE 2018 Herbsttagung and 59th meeting, Chemnitz, Germany. www.dante.de/events.htm

Sep 30 — *TUGboat* **39**:3, submission deadline.

Oct 4 – 7 — Ladies of Letterpress + Print Week, St. Louis, Missouri. www.letterpressconference.co

Oct 5 – 7 — Oak Knoll Fest XX, New Castle, Delaware. www.oakknoll.com/fest

Oct 18 – 19 — Centre for Printing History & Culture, "Baskerville in France", Birmingham City University, Birmingham, UK. www.cphc.org.uk/events

Oct 24 — Award Ceremony: The Updike Prize for Student Type Design, Speaker: Victoria Rushton, Providence Public Library, Providence, Rhode Island. www.provlib.org/updikeprize

Oct 25 – 27 — American Printing History Association's 43rd annual conference, held jointly with The Friends of Dard Hunter, "Matrices: The Social Life of Paper, Print, and Art", Iowa City, Iowa. printinghistory.org

2019

Jan 4 – 5 — College Book Art Association Biennial Meeting, "The Photographic Artists' Book", The University of Arizona, Tucson, Arizona. www.collegebookart.org

Feb 3 – 6 — CODEX VII 2019 Book Fair and Symposium, Richmond, California. www.codexfoundation.org

Mar 1 — **TUG election:** nominations due. tug.org/election

Mar 2 – 4 — Typography Day 2019, "Experimental Typography". IDC School of Design, Indian Institute of Technology Bombay, Mumbai, India. www.typoday.in

Apr 26 — **TUG election:** ballots due. tug.org/election

Jul 3 – 5 — Seventeenth International Conference on New Directions in the Humanities (formerly Books, Publishing, and Libraries), "The World 4.0: Convergences of Knowledges and Machines", University of Granada, Granada, Spain. thehumanities.com/2019-conference

Jul 9 – 12 — Digital Humanities 2019, Alliance of Digital Humanities Organizations, Utrecht, The Netherlands. adho.org/conference

Jul 19 – Aug 1 — SIGGRAPH 2019, "Generations", Los Angeles, California. s2019.siggraph.org

Status as of 25 August 2018

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568, email: office@tug.org). For events sponsored by other organizations, please use the contact address provided.

User group meeting announcements are posted at tug.org/meetings.html. Interested users can subscribe and/or post to the related mailing list, and are encouraged to do so.

Other calendars of typographic interest are linked from tug.org/calendar.html.

TUGBOAT Volume 39 (2018), No. 2

Introductory

105 *Susanne Raab* / The `tikzducks` package
• introduction, tutorials, and fun with these Ti*k*Z graphics

Intermediate

155 *Karl Berry* / The treasure chest
• new CTAN packages, April–August 2018

122 *Paulo Cereda* / Arara — TEX automation made easy
• TEX build tool supporting MVEL expressions and more

113 *Will Robertson* / Font loading in LATEX using the `fontspec` package: Recent updates
• usage summary, filename loading recommendation, futures

117 *Joseph Wright* / Supporting color and graphics in `expl3`
• LATEX2ε vs. LATEX3 drivers, color, comparison with Ti*k*Z, examples

119 *Joseph Wright* / `siunitx`: Past, present and future
• origination, concepts, API, and enhancements

Intermediate Plus

107 *Frank Mittelbach* / A rollback concept for packages and classes
• supporting rollback by date or version for packages, as well as the kernel

131 *Ross Moore* / Implementing PDF standards for mathematical publishing
• five-year plan for archivable and accessible mathematical PDFs

126 *Will Robertson* / The Canvas learning management system and LATEXML
• connecting coursework documents in LATEX with web-based management

Advanced

136 *Jaeyoung Choi, Ammar Ul Hassan, Geunho Jeong* / FreeType_MF_Module: A module for using METAFONT directly inside the FreeType rasterizer
• rendering METAFONT fonts on demand from within FreeType

143 *S. K. Venkatesan* / WeTEX and Hegelian contradictions in classical mathematics
• attempted computability of math markup; hierarchy of infinities

Reports and notices

98 TUG 2018 conference information

100 *Joseph Wright* / TUG goes to Rio
• conference report

104 *Joseph Wright* / TEX Users Group 2018 Annual Meeting notes

147 TUG 2018 abstracts (Behrendt, Coriasco et al., Heinze, Hejda, Loretan, Mittelbach, Moore, Ochs, Veytsman, Wright)

149 From other TEX journals: *MAPS* 48 (2018); *Die TEXnische Komödie* 2–3/2018; *Eutypon* 38–39 (October 2017)

151 *Bart Childs* and *Rick Furuta* / Don Knuth awarded Trotter Prize
• a prize awarded by Texas A&M University to illuminate connections between science and religion

152 *Barbara Beeton* / Hyphenation exception log
• update for missed and incorrect U.S. English hyphenations

153 *Boris Veytsman* / *W. A. Dwiggins: A Life in Design*, by Bruce Kennett
• review of this superb and lavishly illustrated biography of Dwiggins

156 *John Atkinson* / Comics: Hyphe-nation; Clumsy

157 TEX consulting and production services

158 Institutional members

159 *TUG Election committee* / TUG 2019 election

160 Calendar