

# TUGBOAT

Volume 43, Number 2 / 2022  
TUG 2022 Conference Proceedings

<b>TUG 2022</b>	86	Conference information and program
	89	Jim Hefferon / <i>TUG 2022 conference report</i>
	90	Robin Laakso / <i>TUG 2022 Annual General Meeting notes</i>
	93	Karl Berry / <i>David C. Walden, 1942–2022</i>
	96	Paulo Ney de Souza / <i>Interview with John Lees-Miller</i>
	100	Paulo Ney de Souza / <i>Interview with Boris Veytsman</i>
<b>Publishing</b>	104	Carlos Evia / <i>The future of technical documentation starts with its recent past</i>
<b>Resources</b>	108	Dag Spicer / <i>A stroll through computer history at the CHM</i>
<b>Fonts</b>	109	Steven Matteson / <i>Type design: Catching up to the past</i>
<b>Software &amp; Tools</b>	120	Peter K.G. Williams / <i>The Tectonic Project: Envisioning a 21st-century T<sub>E</sub>X experience</i>
	127	Island of T <sub>E</sub> X / <i>IoT theatre presents: The Tempest</i>
	130	Boris Veytsman / <i>Using knitr and L<sup>A</sup>T<sub>E</sub>X for literate laboratory notes</i>
	134	Marnanel Thurman / <i>yex: a T<sub>E</sub>X-alike typesetter in Python</i>
	136	Jean-Michel Hufflen / <i>Extracting information from (L<sup>A</sup>)T<sub>E</sub>X source files</i>
	142	Apu V, Rishi T, Aravind Rajendran / <i>L<sup>A</sup>T<sub>E</sub>X profiling of author submissions — completeness &amp; usability checking</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	148	L <sup>A</sup> T <sub>E</sub> X Project Team / <i>L<sup>A</sup>T<sub>E</sub>X news, issue 35, June 2022</i>
	155	Éric Guichard, Jean-Michel Hufflen / <i>Introductory L<sup>A</sup>T<sub>E</sub>X workshop, en français</i>
	156	Lloyd Prentice / <i>Self-publishing, L<sup>A</sup>T<sub>E</sub>X, and Markdown</i>
	159	Paulo Cereda, Phelype Oleinik / <i>The story of a silly package</i>
	162	Joseph Wright / <i>Key–value setting handling in the L<sup>A</sup>T<sub>E</sub>X kernel</i>
	164	Joseph Wright / <i>siunitx: Launching version 3</i>
	165	Joseph Wright / <i>Case changing: L<sup>A</sup>T<sub>E</sub>X reaches Unicode-land</i>
	167	Ulrike Fischer / <i>Using spot colors in L<sup>A</sup>T<sub>E</sub>X</i>
	172	Chetan Shirore, Ajit Kumar / <i>The luatruhtable L<sup>A</sup>T<sub>E</sub>X package</i>
<b>Multilingual</b>	176	Oleksandr Baranovskiy / <i>L<sup>A</sup>T<sub>E</sub>X classes for doctoral theses in Ukraine: Interesting tips and painful problems</i>
<b>Document Processing</b>		
<b>Humanities</b>	182	H. Andrew Black, Hugh J. Paterson III / <i>XLingPaper’s use of T<sub>E</sub>X technologies</i>
<b>Electronic Documents</b>	197	Dennis Müller, Michael Kohlhase / <i>A L<sup>A</sup>T<sub>E</sub>X-based ecosystem for semantic/active mathematical documents</i>
<b>ConT<sub>E</sub>Xt</b>	202	Hans Hagen, Mikael Sundqvist / <i>Pushing math forward with ConT<sub>E</sub>Xt lmtx</i>
<b>Hints &amp; Tricks</b>	207	Karl Berry / <i>The treasure chest</i>
<b>Abstracts</b>	209	TUG 2022 abstracts (Austin, Blakesley, Castañeda, Chernoff, Cheung, Claudio, Fine, Gundlach, Hickman, Jimenez, Khalighi, Luc, Mariano, Moore, Ohri, Park C., Park E., Preining, samcarter, Schmah, Vrabcová, Wu)
	213	MAPS: Contents of issue 52 (2022)
	214	<i>La Lettre GUTenberg</i> : Contents of issue 45 (2021)
	215	<i>Die T<sub>E</sub>Xnische Komödie</i> : Contents of issue 2/2022
<b>TUG Business</b>	215	TUG institutional members
	216	2023 T <sub>E</sub> X Users Group election
<b>News</b>	217	Calendar
<b>Advertisements</b>	218	TUG 2022 advertisements
	219	T <sub>E</sub> X consulting and production services

## TeX Users Group

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

### Individual memberships

2022 dues for individual members are as follows:

- Trial rate for new members: \$30.
- 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 may also choose to receive *TUGboat* and other benefits electronically, at a discount. All membership options are described at [tug.org/join](http://tug.org/join).

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 libraries and other organizations or individuals for whom memberships are either not appropriate or desired. Subscriptions are delivered on a calendar year basis. The subscription rate for 2022 is \$115.

### 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](http://tug.org/instmem) 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: August 2022]

Printed in U.S.A.

## Board of Directors

Donald Knuth, *Ur Wizard of TeX-arcana*<sup>†</sup>

Boris Veytsman, *President*\*

Arthur Rosendahl\*, *Vice President*

Karl Berry\*, *Treasurer*

Klaus H"oppner\*, *Secretary*

Barbara Beeton

Johannes Braams

Paulo Cereda

Kaja Christiansen

Ulrike Fischer

Jim Hefferon

Frank Mittelbach

Ross Moore

Norbert Preining

Raymond Goucher (1937–2019),

*Founding Executive Director*

Hermann Zapf (1918–2015), *Wizard of Fonts*

*\* member of executive committee*

*† honorary*

See [tug.org/board](http://tug.org/board) for a roster of all past and present board members, and other official positions.

### 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](http://tug.org)  
[tug.org/TUGboat](http://tug.org/TUGboat)

### Electronic mail

General correspondence,  
membership, subscriptions:  
[office@tug.org](mailto:office@tug.org)

Submissions to *TUGboat*,  
letters to the Editor:  
[TUGboat@tug.org](mailto:TUGboat@tug.org)

TeXnical support,  
public mailing list:  
[support@tug.org](mailto:support@tug.org)

Contact the  
Board of Directors:  
[board@tug.org](mailto:board@tug.org)

Copyright © 2022 TeX Users Group.

Copyright to individual articles within this publication remains with their authors, so the articles may not be reproduced, distributed or translated without the authors' permission.

For the editorial and other material not ascribed to a particular author, permission is granted to make and distribute verbatim copies without royalty, in any medium, provided the copyright notice and this permission notice are preserved.

Permission is also granted to make, copy and distribute translations of such editorial material into another language, except that the TeX Users Group must approve translations of this permission notice itself. Lacking such approval, the original English permission notice must be included. An information notice to the *TUGboat* editors regarding such redistribution is appreciated.

**2022 Conference Proceedings**

TeX Users Group  
Forty-third annual TUG conference  
Online  
July 22–24, 2022

# TUGBOAT

COMMUNICATIONS OF THE T<sub>E</sub>X USERS GROUP

TUGBOAT EDITOR      BARBARA BEETON

PROCEEDINGS EDITOR      KARL BERRY

VOLUME 43, NUMBER 2, 2022  
PORTLAND, OREGON, U.S.A.

# TUG 2022 — Online — July 22–24, 2022

The forty-third annual TUG conference  
<https://tug.org/2022> ■ [tug2022@tug.org](mailto:tug2022@tug.org)

## Conference committee

Jennifer Claudio  
 Rohit Goswami  
 Jérémy Just  
 Robin Laakso  
 Ross Moore  
 Norbert Preining  
 Will Robertson  
 Arthur Rosendahl, co-principal organizer  
 Paulo Ney de Souza, co-principal organizer  
 Boris Veytsman  
 Alan Wetmore

## Sponsors

TeX Users Group  
 Carleton Production Centre  
 DANTE e.V.  
 Google  
 Overleaf  
 speedata  
 STM Document Engineering Pvt Ltd  
 The University of Adelaide  
*with generous assistance from many individual contributors.*  
 Conference artwork: Jennifer Claudio

Thanks to all!



dante<sub>e.V.</sub>

Google

Overleaf

speedata

TeXFolio



### TeXnology Inc.

Amy Hendrickson  
 57 Longwood Ave. #8  
 Brookline, MA 02446  
 +1 617-738-8029  
 Email: [amyh \(at\) texnology.com](mailto:amyh@texnology.com)  
 Web: <https://texnology.com>

Full time L<sup>A</sup>T<sub>E</sub>X consultant for more than 30 years; have worked for major publishing companies, leading universities, and scientific journals. Our macro packages are distributed on-line and used by thousands of authors. See our site for many examples: [texnology.com](https://texnology.com).

- **L<sup>A</sup>T<sub>E</sub>X Macro Writing:** Packages for books, journals, slides, posters, e-publishing and more; Sophisticated documentation for users.
- Design as well as L<sup>A</sup>T<sub>E</sub>X implementation for e-publishing, print books and journals, or specialized projects.
- Data Visualization, database publishing.
- Innovative uses for L<sup>A</sup>T<sub>E</sub>X, creative solutions our speciality.
- L<sup>A</sup>T<sub>E</sub>X Training, customized to your needs, on-site or via Zoom. See <https://texnology.com/train.htm> for sample of course notes.

Call or send email: I'll be glad to discuss your project with you.

# TUG 2022 program

(All times and days listed are UTC.)

<b>Thursday July 21</b>	13:00	Éric Guichard, ENS de Lyon CNRS; <i>L<sup>A</sup>T<sub>E</sub>X</i> workshop (in French) Jean-Michel Hufflen, FEMTO-ST & Univ. of Bourgogne Franche-Comté	
	16:00	Cheryl Ponchin, IDA/CCR-P; Sue DeMeritt, IDA/CCR-La Jolla	<i>L<sup>A</sup>T<sub>E</sub>X</i> workshop (in English)
	19:00	Alexánder Borbón Alpízar, El Tecnológico de Costa Rico <i>(continued)</i>	<i>Introducción a L<sup>A</sup>T<sub>E</sub>X</i> (in Spanish)
<b>Friday July 22</b>	15:00	Boris Veytsman, T <sub>E</sub> X Users Group	<i>Conference opening</i>
	15:15	Peter K. G. Williams, Center for Astrophysics   Harvard & Smithsonian	<i>The Tectonic Project: Envisioning a 21st-century T<sub>E</sub>X experience</i>
	16:00	Carlos Evia, Virginia Tech	<i>The future of technical documentation starts with its recent past</i>
	16:45	Joseph Wright, L <sup>A</sup> T <sub>E</sub> X Project	<i>siunitx: Launching version 3</i>
	17:30	Paulo Cereda, Overleaf	<i>The story of a silly package</i>
	18:15	Jennifer Claudio, San Jose, CA	<i>Revamping a youth chess workbook using L<sup>A</sup>T<sub>E</sub>X packages</i>
	20:00	Paulo Cereda, Overleaf	<i>IoT theatre presents: The Tempest</i>
	20:45	Lloyd Prentice, Writersglen Pub.	<i>A self-publisher's take on T<sub>E</sub>X</i>
	21:30	Hubert Hickman, Matthew Mariano, Haibin Wu, Hong Dat Cheung, Essex Management LLC	<i>Using L<sup>A</sup>T<sub>E</sub>X deployed in AWS as a PDF report generation tool for a cancer clinical trial search engine</i>
	22:15	David Blakesley, Clemson University	<i>The residual concepts of production vs. the emergent cultures of distribution in publishing</i>
	<b>Saturday July 23</b>	00:00	Oliver Austin, UC Davis
00:45		Sarai Castañeda	<i>Fonts and formats of constitutions</i>
01:30		Boris Veytsman, T <sub>E</sub> X Users Group	<i>Using knitr and L<sup>A</sup>T<sub>E</sub>X for literate lab notes</i>
02:15		Norbert Preining, Mercari Japan, TUG, T <sub>E</sub> X Live	<i>T<sub>E</sub>X Live 2022 status update</i>
03:00		Vafa Khalighi, Sydney, Australia	<i>Right to left beamer documents in X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X</i>
11:15		Patrick Gundlach, speedata GmbH	<i>Boxes and glue: T<sub>E</sub>X algorithms reimplemented</i>
12:00		Chetan Shirore, K.T.H.M. College, Nashik; Ajit Kumar, Inst. of Chemical Technology, Mumbai	<i>The luatruhtable package in L<sup>A</sup>T<sub>E</sub>X</i>
12:30		Tereza Vrabcová, Masaryk Univ.	<i>A gentle introduction to Markdown for writers</i>
13:15		Jonathan Fine	<i>The UK T<sub>E</sub>X Users Group — a personal history</i>
14:00		samcarter	<i>Bricks and pieces</i>
15:30		Ulrike Fischer, L <sup>A</sup> T <sub>E</sub> X Project, Bonn, Germany	<i>New in stock — a walk though recent L<sup>A</sup>T<sub>E</sub>X improvements (that you may have missed)</i>
16:15		Joseph Wright, L <sup>A</sup> T <sub>E</sub> X Project Team	<i>Case changing: L<sup>A</sup>T<sub>E</sub>X reaches Unicode-land</i>
18:00			<i>TUG Annual General Meeting</i>
20:45		Nicolas Jimenez, MathPix, Inc.	<i>Bridging the gap between L<sup>A</sup>T<sub>E</sub>X/PDFs and the modern web</i>
21:30	Tia Luc	<i>Observations and analysis of Vietnamese text</i>	
22:15	Max Chernoff	<i>Comparing T<sub>E</sub>X engines and formats</i>	
23:00	Vafa Khalighi	<i>Typesetting mathematics in Persian</i>	

**Sunday  
July 24**


---

01:00	Paulo Ney de Souza, Books in Bytes	<i>Interview with Boris Veytsman</i>
01:45	Ross Moore, Macquarie University	<i>Accessible tables using Tagged PDF</i>
02:30	Vafa Khalighi	<i>Bidirectional multi-columns and paragraph footnotes in T<sub>E</sub>X</i>
07:00	Apu V, Rishi T, Aravind Rajendran, STM Document Engineering Private Ltd.	<i>L<sup>A</sup>T<sub>E</sub>X profiling of author submissions</i>
07:45	Oleksandr Baranovskiy, Doctor Barbarus Services & Institute of Mathematics of the National Academy of Sciences of Ukraine	<i>L<sup>A</sup>T<sub>E</sub>X classes for doctoral theses in Ukraine: Interesting tips and painful problems</i>
08:30	Dennis Müller, Michael Kohlhase, FAU Erlangen-Nürnberg, Germany	<i>sT<sub>E</sub>X3—A L<sup>A</sup>T<sub>E</sub>X-based ecosystem for semantic/active mathematical documents</i>
09:15	Jean-Michel Hufflen	<i>Extracting information from (L<sup>A</sup>)T<sub>E</sub>X source files</i>
11:00	Mikael P. Sundqvist, Lund University	<i>Pushing math forward with <code>luametatex</code> and <code>ConT<sub>E</sub>Xt</code></i>
11:45	Ulrike Fischer	<i>Using spot colors with L<sup>A</sup>T<sub>E</sub>X</i>
12:30	Joseph Wright, L <sup>A</sup> T <sub>E</sub> X Project Team	<i>Key-value setting handling in the kernel</i>
13:15	Marnanel Thurman	<i>Building a T<sub>E</sub>X-alike in Python</i>
15:00	Aditya Ohri, UC Berkeley; Tanya Schmah, University of Ottawa	<i>Machine translation of mathematical text</i>
15:45	Jonathan Fine	<i>Access and accessibility</i>
17:15	Paulo Ney de Souza	<i>Interview with John Lees-Miller</i>
19:00	Dag Spicer, Computer History Museum	<i>A walk through 2,000 years of computer history</i>
19:45	Christopher Park, San Jose, CA; Emily Park, UC Berkeley	<i>Musical composition typesetting</i>
20:30	Steven Matteson, Matteson Typographics	<i>Type design: Catching up to the past</i>
21:15	Boris Veytsman, T <sub>E</sub> X Users Group	<i>Conference closing</i>
≈ 21:15	<i>end</i>	

---

**TUG 2022 conference report**

Jim Hefferon

The TUG 2022 conference ran from July 21–24 (dates are for the Eastern US). As in the prior two years it was conducted online, over Zoom. The conference was completely free to participants, as well as to anyone watching the parallel YouTube live feed.

There were 232 registered attendees. While the largest number were from North America, there were also significant numbers from Europe, Asia, and Australia & Oceania, and in total there were representatives from six of the seven continents. The YouTube feed also had a steady list of watchers.

The first day was given over to L<sup>A</sup>T<sub>E</sub>X workshops, one each in French, English, and Spanish. The videos of these are already popular.

The remainder of the conference, from July 22 to July 24, consisted of forty three presentations along with a conference opening, a closing, and the TUG Annual General Meeting. The full schedule is at [tug.org/tug2022/program.html](https://tug.org/tug2022/program.html). As in the prior online versions of the conference, the talks ran on a 24-hour rotating schedule so that the times could be reasonably convenient for the presenters. To the extent that it was practical, talks were grouped into sessions by topic. The result is that, as with the past two years, we were treated to a fascinating mix of talks by speakers both from within and outside of the T<sub>E</sub>X community.

There were three keynotes: David Blakesley spoke on *The residual concepts of production vs. the emergent cultures of distribution in publishing*, Carlos Evia spoke on *The future of technical documentation starts with its recent past*, and Peter K. G. Williams spoke on *The Tectonic Project: Envisioning a 21st century T<sub>E</sub>X experience*. There were live interviews with Boris Veytsman, current TUG President, and with John Lees-Miller, the CTO of Overleaf. In addition, Computer History Museum senior curator Dag Spicer took us on a tour through computing history starting with the Antikythera Mechanism.

There were a number of themes to the contributions. Rather than trying to be comprehensive, I will mention only two topics that continue from prior years and have proved to be of special interest: talks that describe and expand on recent changes either in the L<sup>A</sup>T<sub>E</sub>X kernel or in the underlying T<sub>E</sub>X engines, and those addressing accessibility.

Once the talk videos have been processed, including editing and adding a transcription for closed caption, they will be posted on the TUG YouTube channel at [youtube.com/c/texusersgroup](https://youtube.com/c/texusersgroup).

The Annual General Meeting was on July 23. President Boris Veytsman made some remarks and Secretary Klaus H<sup>o</sup>ppner gave a presentation summarizing the events of the organization's prior year. After a little discussion from the floor, the meeting adjourned. Note that for confidentiality reasons this meeting is not recorded and so will not be available on YouTube. The minutes from the AGM are in this issue of *TUGboat*.

Organizing the conference is a huge job and volunteers did amazing work. The core group of the conference committee, Paulo Ney de Souza, J<sup>e</sup>r<sup>e</sup>my Just, Ross Moore, and Norbert Preining gave their time and energy for running the Zoom sessions and handling the very many details, and are now engaged in post-processing of the videos. In addition, Karl Berry and Jennifer Claudio provided support, along with the entire Board, the President, and Executive Director Robin Laakso. The T<sub>E</sub>X Users Group gratefully acknowledges the Zoom link provided by the University of Adelaide through Will Robertson's efforts. They were a conference sponsor and TUG is also grateful for sponsorship from Carleton Production Centre (Ottawa), DANTE e.V., Google, Overleaf, speedata, STM Document Engineering Pvt Ltd, and individual donors.

The fact that having so many people participating in the conference is great for T<sub>E</sub>X and friends, as well as great for those participants, has not escaped the conference organizers or the TUG Board. When we return to in-person conferences in the future they likely will include some kind of online component.

However, no operation of this scale is without hiccups. There were fewer registrants this year than last year, which may have been because of less advertising on social media. Zoom instituted a change making it harder for viewers to comment or ask questions, reducing the role of the conference as a place to interact. Although organizers quickly responded by adding links to the communication and media platforms Wonder.me and Zulip, in the future such links should be there at the start. And, although the instructions to authors mentioned the potential problems with including copyrighted material in the presentation, it was not prominent enough and consequently the first day's video immediately got a takedown notice (since rectified).

In summary, there were many talks, of excellent quality. There were many participants, many who might not have the chance to attend in person. Overall the conference was a great success.

◇ Jim Hefferon  
<https://hefferon.net>

---

## TUG 2022 Annual General Meeting notes

Notes recorded by Robin Laakso

Boris Veysman, TUG president, opened the meeting at 11:00 (PDT): “Welcome to the second online AGM.” He stated that the Zoom license has limits, and asked that participants raise a hand to say something. He said to please state your name and state if you are a TUG member. Let us know if you want to be heard. He requested that people be respectful, speak in good faith, and be courteous.

Boris showed two slides: Rules of the game and Topics (basically the meeting agenda).

Klaus Höppner, TUG secretary, gave a TUG status update and financial report. He showed a series of slides, most of which are included in this report (slides are omitted here if they merely duplicate information from web pages):

1. The current TUG board of directors. ([tug.org/board](http://tug.org/board))
2. “Formalities”: 2023 is an election year; 5 directors’ terms end + 3 open positions = 8 positions for election. ([tug.org/election](http://tug.org/election))
3. “Members end of June 2022”: TUG membership has declined slightly in 2022 so far; 2021 membership was up. Klaus commented that the trial membership program is quite successful, and leads to renewals in about 50% of cases.
4. “Profit & Loss 2021” included product sales: DVDs, *TUGboat*, Lucida and expenses: journals, software, Lucida, postage, payroll, overhead.
5. “Assets and Liabilities” and “Committed Funds” slides were next, both as of the end of 2021.
6. “Donations 2022”: Klaus commented that donations are up and quite generous this year. ([tug.org/donors](http://tug.org/donors))

Regarding the donation slide, Klaus further stated that the TUG board is very thankful for the generous UK-TUG donation, that no restrictions were made with the contribution, and that the TUG board was not involved in and has no knowledge of the discussion within UK-TUG leading to this donation. Jonathan Fine interrupted. Boris said he was aware of the raised hand and that JF would be heard when Klaus finished his report.

7. “International Conferences”: Klaus said that the ConT<sub>E</sub>Xt meeting will be in person again this year, and also that BachoT<sub>E</sub>X 2022 will be in person this year, but this time in September (not May). ([tug.org/meetings](http://tug.org/meetings))

8. “T<sub>E</sub>X Live/T<sub>E</sub>X Collection”: Klaus recognized the hard work of the DVD team. He said that the T<sub>E</sub>X Collection DVD is a joint effort in terms of manufacturing as well as software production. Members can order DVDs from their user group. Two user groups have professional offices, which promote the distribution of the software, among other things.

Klaus defined a subset of MiK<sub>T</sub>E<sub>X</sub>, replacing the separate proT<sub>E</sub>Xt, given the size constraints of the DVD; he thanked Christian Schenk for help with this project and the end result fitting on the T<sub>E</sub>X Collection DVD.

([tug.org/texcollection](http://tug.org/texcollection))

9. “Board Motions” ([tug.org/board/motions.html](http://tug.org/board/motions.html)):
  - 2021.6 Approval of the 2022 budget.
  - 2021.7 Adopt TUG bylaws update and AGM procedures. ([tug.org/bylaws](http://tug.org/bylaws))
  - 2022.1 Support for persons in war zones or other extraordinary circumstances.
  - 2022.2 Cover DOI fees for GUTenberg as needed.
  - 2022.3 Funding to support Ukrainian students attending BachoT<sub>E</sub>X.
10. A “Last words” slide ended the presentation: Farewell to Dave Walden (1942–2022).

### Discussion

Jonathan Fine asked Klaus to display slide 9 (“Donations 2022”) and share his screen. JF read the second sentence: TUG was not involved and has no knowledge about discussion within UK-TUG about how to distribute the money. JF said that TUG VP Arthur R[osendahl] was involved in the discussion. JF said Arthur R was on the committee. JF said Arthur R was involved in the discussion about how to distribute the money.

JF said that the TUG board doubled the quorum from 50 to 100. He said that no motion that is binding may be moved at the AGM without being approved by the board. JF stated that he submitted a motion on 13 July 22 with a supporting statement. He said he asked that it be circulated to members. He said that Klaus read it and gave a rebuttal. JF encouraged “ordinary” members to speak up. He asked other attendees to comment on the AGM procedures. No one did.

Keiran Harcombe asked if the board and other members of TUG would consider adopting a code of conduct for its members, board and others who work on projects?

Boris replied, saying that he agreed that the Board should discuss this.

Klaus H: Several board members are members of other user groups. There is always an agreement that within the TUG board I am a TUG board member and not a DANTE member. Everything on the TUG board is internal to that organization, not the other membership/organization. UK-TUG asked DANTE if they were interested in a donation upon dissolution.

Boris V: Please open the page about conferences. Correction: The ConT<sub>E</sub>Xt meeting should be September 2022 (not 2021). The ConT<sub>E</sub>Xt and BachoT<sub>E</sub>X meetings are quite close together. [Post-conference correction made on included slide.]

Keiran H and Jonathan F discussed whether the earlier comment by JF meant that there was a conflict of interest during the discussion leading to the dissolution of UK-TUG.

Boris V: Does anyone else have something they would like to speak about?

Keiran H: This may seem like an incredibly stupid question: has anyone in the TUG considered making a guide to all the various parts of the T<sub>E</sub>X world? How different systems work and the approach to them.

Boris V: I will comment. We have some documentation but always need more. Recently someone applied for T<sub>E</sub>X Development funds to write more documentation. If there are people who want to do this kind of work we welcome that.

David Carlisle: Joseph Wright and myself (David C) support a site (previously managed by Robin Fairbairns), [texfaq.org](http://texfaq.org), which provides a lot of documentation. The L<sup>A</sup>T<sub>E</sub>X team has a website as well.

[Post-conference addition: the web page [tug.org/levels](http://tug.org/levels) gives an extremely brief overview of the major components of the T<sub>E</sub>X world. Max Chernoff's presentation at this conference was on this topic.]

Question: Are we going to publish the slides?

Answer: Yes.

Klaus: This AGM report will be part of the minutes.

Paulo Ney: I would like to suggest that TUG support of GUT à la DOI could be offered to other user groups. TUG could do this for user groups worldwide. It would be very positive for many local groups. This would require [tug.org](http://tug.org) to hold a mirror copy of all those publications. There are only benefits. Include user groups under one umbrella. It would be very nice for other user groups to have this DOI support.

Boris V: This option was considered, but GUT did not go this way. [Post-conference clarification: TUG is happy to assist other groups with DOI support, technically or financially. We'll work together to determine what's best on a case-by-case basis.]

Any other comments or questions?

Jonathan F: I came across a 2019 email about the \$10K contribution for PDF accessibility. What is it being used for?

Boris V: We have provided a couple of small grants. It's not enough for a full time programmer, unfortunately, but we welcome interested persons.

[Post-conference addition: the recent addition of `amsmath` support to `latex2nemeth` project ([ctan.org/pkg/latex2nemeth](http://ctan.org/pkg/latex2nemeth)) was funded 50% by the PDF Accessibility Fund money and 50% by the T<sub>E</sub>X Development Fund.]

Keiran H: I wish to commend Jonathan Fine for trying to push this forward. The world needs this kind of accessibility work.

Boris V: We welcome proposals.

Jonathan F: This may not be the best place to say this. I am tremendously grateful for all the online work being done. Also I am personally grateful for all the work that has been done to make this conference happen.

Boris V: Thank you.

The meeting came to a close at 12:25 (PDT).

## Annual General Meeting 2022 of the TeX Users Group

Klaus H\"oppner (secretary) for the board

July 23, 2022

## Committed Funds (status end of 2021)

Fund	Amount
Bursary	5,927
CTAN	9,948
GUST e-foundry	406
LaTeX3	12,392
LuaTeX	1,494
LyX	11
MacTeX	8,499
PDF Accessibility	11,758
TeX Development	5,221
owed:	2,000
available:	3,221
Sum	55,655

## Members end of June 2022

End of June we had 1,102 paid members, with:

- 1,052 renewals, 50 new (33 of them trial, 11 joint)
- –48 compared to June 2021
- 85 institutional, 116 joint members
- 376 with electronic-only option
- 356 with auto-renewal option
- 34 of last year's 72 trial members renewed so far
- final numbers of last years:
  - December 2021: 1,210
  - December 2020: 1,189
  - December 2019: 1,238
  - December 2018: 1,214
  - December 2017: 1,178

## Donations 2022

- Google: 10,000
- Anonymous: 10,000
- UKTUG: 5,366

The remaining funds of dissolved UKTUG were donated to two user groups.

TUG was not involved and has no knowledge about discussion within UKTUG about how to distribute the money.

No restrictions were made with the donation.

## Profit &amp; Loss 2021

Income		Expenses	
Membership dues	79,320	Cost of goods sold	
Product sales	4,423	TUGboat	22,328
Contributions	21,311	Software	2,391
Annual Conference	2,636	Fonts	1,675
Other	749	Postage	1,827
		Other	372
		Office	
		Payroll	64,274
		Overhead	12,924
		Contributions	2,000
		Other	84
Sum	108,440	Sum	107,875
		Net gain	565

## International Conferences

Past

- TUG 2021 (online)
- DANTE autumn meeting (Germany, Sept. 2021 online)
- ConTeXt meeting (Belgium, Sept. 2021)
- GulT meeting (Italy, Oct. 2021 online)
- DANTE 2022 summer meeting (hybrid)

Upcoming

- ConTeXt meeting (Germany, Sept. 12–18, 2022)
- BachoTeX 2022 (Poland, Sept. 21–25, 2022)

## Assets and Liabilities (status end of 2021)

Assets		Liabilities	
Checkings/Savings	173,602	Committed funds	55,655
Accounts Receivable	395	Admin services	1,445
		Member income	10,075
		Payroll	1,280
Sum	173,997	Sum	68,455
		Equity	105,542

## TeX Live/TeX Collection

- TeX Live 2022 released as planned
- Team: Karl, Norbert, Siep Kronenberg, Akira Kakuto et al.
- TeX Collection DVDs produced by DANTE in Germany, in cooperation with TUG and various user groups, containing:
  - TeX Live
  - MiKTeX
  - MacTeX (Herbert Schultz)
  - CTAN snapshot (Manfred Lotz)
- Former proTeXt distribution for Windows is abandoned, replaced by a special MiKTeX subset defined by Klaus (due to space restrictions). Thanks to Christian Schenk for support!

**David C. Walden, 1942–2022**

Karl Berry



David Corydon Walden was born June 7, 1942, in Longview, Washington. He died April 27, 2022, at his home in East Sandwich, Massachusetts, of mantle cell lymphoma. He had been coping with the lymphoma for several years.

\*

Dave had a long and distinguished professional career as a programmer, technical manager, and general manager at Bolt, Beranek and Newman (BBN). His best-known programming project was also among his first: being part of the small team that developed the Interface Message Processor (IMP), the original packet-switching gateways, known today as network routers. This story is told in plenty of other places (there is an overview in [3]). Dave himself wrote about the rediscovery and resurrection of the IMP code to run on emulators ca. 2014 ([walden-family.com/impcode](http://walden-family.com/impcode)). With Bernie Cosell, he developed and then documented the famous will/won't/do/don't telnet negotiation protocol; Dave's own biography page [5] goes into some detail on this.

After his retirement from BBN, Dave became increasingly involved in writing and computer history. One of his first post-retirement projects, a new edition of one of his management books, brought him to L<sup>A</sup>T<sub>E</sub>X. He tells this story in his first *TUGboat* article (“Writing a big book in L<sup>A</sup>T<sub>E</sub>X”, *TUGboat* 24:2, [10]). After losing work and time due to Microsoft Word's lack of compatibility between versions, his attraction to (L<sup>A</sup>)T<sub>E</sub>X was driven by his twin desires for explicit ASCII markup and a stable system. As a programmer, he was much happier with T<sub>E</sub>X's explicitly-written source files than Word's invisible controls.

With that initial introduction, it quickly became clear how invaluable a contributor Dave would become to the T<sub>E</sub>X community. He immediately helped establish *The PracT<sub>E</sub>X Journal* ([tug.org/](http://tug.org/)

*pracjournal*), an online journal published by TUG with its first issue in 2005. Dave wrote a regular column for the journal (“Travels in T<sub>E</sub>X Land”, [8]). He also wrote the software to generate the journal's web site (from its inception in 2005 until its final issues), the first of several such projects he undertook.

His interests in T<sub>E</sub>X and computing history combined with his T<sub>E</sub>X interview project [9], which he pursued throughout nearly his entire tenure working with T<sub>E</sub>X: he conducted the first interview in 2004 (Dan Luecking, a T<sub>E</sub>X, METAFONT, and MetaPost contributor and package author for many years), and his last was in 2021 (Amelia Hugill-Fontanel, associate curator at the RIT Cary Graphic Arts Collection). As even this sample of two shows, Dave cast his net for interviewees widely and always found insightful and unusual questions to ask. (By the way, Dave always encouraged others to participate in this and his other projects, and a new volunteer to continue an interview series would be most welcome; just email me.)

After several years of interviews, enough material had been accumulated for a collection, and so another of Dave's interests was engaged: publishing books. He was instrumental in setting up TUG's book publications, in everything from acquiring the ISBN numbers, figuring out the best print-on-demand service (Lightning Source), collecting and editing the material, and generally directing the publication of TUG's first two books—the interviews (*T<sub>E</sub>X People*) and a commemoration of T<sub>E</sub>X's 32nd anniversary conference (*T<sub>E</sub>X's 2<sup>5</sup> Anniversary*). (Both are listed in [6], and linked from [tug.org/books/#tug](http://tug.org/books/#tug).)

In general, it seemed for any project Dave completed, he would then write up a report about it. He wrote so much! Throughout his many articles and talks relating to T<sub>E</sub>X, he emphasized the practical side of getting work done. He still used Word for simple documents, and the perennial question on T<sub>E</sub>X forums of “how to make T<sub>E</sub>X compete with Word” held no interest for him. His attitude was, let's try to improve the T<sub>E</sub>X system at what it already does well; for example, he wrote many notes about solving practical problems he faced in his writing.

In addition to these myriad T<sub>E</sub>Xnical projects, Dave was the treasurer for TUG from 2005–2011, and a board member for four years beyond that. As TUG treasurer, Dave worked extensively with long-time employee Robin Laakso, who handles financial matters. He recommended various improvements to TUG accounting, created new reports, and helped the board better understand financial statements. Dave had a lifelong appreciation of double-entry bookkeeping, saying in an interview for *MAPS* [13]:

David C. Walden, 1942–2022

Anyone who dismisses double-entry bookkeeping as boring or too complicated has thrown away the possibility of making use of [a] tremendously powerful organizational tool . . .

Although being treasurer and director of a tiny nonprofit (TUG) may seem rather modest after being a general manager for a major corporation (BBN), Dave repeatedly made the point that the two organizations have more in common than one might think, particularly in the *MAPS* interview and other interviews of him [12]—the same principles for creating viable businesses apply, whether for-profit or not, whether run by volunteers or paid staff. This made a substantial impact on the understanding of many of us on TUG’s place and its operations.

\*

As mentioned above, Dave was deeply interested in computing history, and devoted much of his post-retirement time to numerous history-related projects, including his final talk at a TUG conference (“Noticing history”, *TUGboat* 41:2, [10]). In this connection, he had a long association with the *IEEE Annals of the History of Computing* [7]. His close IEEE colleague David Hemmendinger kindly wrote the following description of Dave’s work with *Annals*:

Dave had a major role in *Annals* work. He joined the *Annals* board in 2006 and from 2008 until last year, he edited the Anecdotes department. From 2011 to 2013, he was the first editor of Interviews. When he resigned as Anecdotes editor last year, he became the Events & Sightings editor until his poor health prevented his continuing. He developed the `annals-extras.org` web site for supplementary material and bibliographic tools. According to it, Dave was co-author of five *Annals* articles, including two on  $\text{\TeX}$ , conducted ten interviews, wrote or co-wrote four anecdotes, wrote eleven E&S reports, and five book reviews.

Dave reluctantly became acting Editor in Chief for six months in 2014. He and colleagues then brought in 40 submissions, an unmatched record. I had joined the board shortly before he took over, and he got me actively involved with *Annals*. We worked on numerous projects, and I greatly enjoyed our collaboration.

Dave excelled as an editor, able to find common ground among reviewers and to guide authors in refining their articles and in improving the English when it was not an author’s native language. When the IEEE changed the magazine production process, Dave worked with its staff to facilitate submission of articles in  $\text{\LaTeX}$

as well as in Microsoft Word and to improve several aspects of the process.<sup>1</sup>

Dave had wanted to edit an issue on digital typography, but gave up that plan when he became sick. He was a guest co-editor of two special issues on the history of BBN and worked closely with guest editors of other issues. When Burt Grad and I edited the desktop publishing issues that grew out of a 2017 workshop, Dave was a co-editor in all but name.

The workshop that David mentions was held at the Computer History Museum, and brought together pioneers from the early days of desktop publishing. This was an intersection of Dave’s interests in publishing,  $\text{\TeX}$ , and computing history, and he was a key advisor in organizing the meeting, including the participation of Don Knuth and Chuck Bigelow, as well as attending himself. A transcript of the entire meeting is online [2], and of course Dave wrote a short report about it (“Collecting memories of the beginning of desktop publishing”, *TUGboat* 38:3, [10]). He also co-authored a two-part history of  $\text{\TeX}$  for the subsequent *Annals* special issues [1].

\*

In addition to his continual writing activities, Dave also returned to programming for some of his  $\text{\TeX}$  projects, mainly for generating web pages, including the Interview Corner, *The Prac $\text{\TeX}$  Journal*, and most extensively for *TUGboat*—he was instrumental in getting all past *TUGboat* issues online, and then generating the online lists of *TUGboat* articles accessible by author, title, and category (such as the link at [10]). He mostly wrote these programs in Perl, with an occasional dip into m4. He always made extensive use of macro capabilities; one of his longest and most technical articles for *TUGboat* was entitled “Macro memories” (*TUGboat* 35:1, [10]), where he discusses his experiences with several macro languages, including  $\text{\TeX}$ ’s. However, the implementation language was almost immaterial to him; he once wrote:

Despite various programming disciplines that have been popular during my era of computing (the move from assembly language to high-level languages, avoidance of goto’s, structured programming, object-oriented programming, etc.), in any high level language I still program as if I was writing FORTRAN in 1964, e.g., `if X eq Y, goto Label A.`

<sup>1</sup> As usual, Dave wrote an article after finishing this effort: “An experience of trying to submit a paper in  $\text{\LaTeX}$  in an XML-first world”, *TUGboat* 40:3, [10].

\*

On a personal note: over the 18 years of our collaboration, Dave and I exchanged thousands upon thousands of emails, and were able to meet in person at many TUG conferences and elsewhere. I was privileged to work with him on nearly all his T<sub>E</sub>X and book projects, and enlisted him for help on plenty of my own. Aside from our T<sub>E</sub>Xnical efforts, we shared wide-ranging interests in books and movies [11] and regularly exchanged recommendations. Our connection through the years was one of the great and unexpected pleasures of my life, and he continues to be an inspiration. He was a remarkable mentor, colleague, and most of all friend.

Dave has already been remembered in many other places, by family, friends and colleagues, including Internet pioneers with whom he worked. The family's obituary is at [olsonparent.com/obituary/David-Walden](https://olsonparent.com/obituary/David-Walden), and includes several personal remembrances. The *New York Times* obituary [3], by noted author Katie Hafner, provides an excellent capsule summary of his professional career, especially his early work on the proto-Internet. A more comprehensive biography of Dave's remarkable careers is published in *Annals* [4]. On an Internet history list, there are personal accounts of some of his earlier technical work ([elists.isoc.org/pipermail/internet-history/2022-May/](https://elists.isoc.org/pipermail/internet-history/2022-May/)).

We are honored and appreciative that Dave designated TUG as one of the two charities for gifts to be made in his memory ([tug.org/donate](https://tug.org/donate)). The other is the hospital that cared for him through his illness (Beth Israel Deaconess Medical Center, Dept. of Medical Oncology, 330 Brookline Ave., Boston, MA 02215).

Dave is survived by his wife Sara, son Luke, daughter-in-law Mindy Sobota, grandchildren Ada and Kai Sobota-Walden, brother Daniel Walden and his wife June, sister Velma Hampson and her husband Paul, sister-in-law Susan Cowles, and numerous nieces and nephews.

We all miss you, Dave.

## References

- [1] Barbara Beeton, Karl Berry, David Walden. T<sub>E</sub>X: A Branch of Desktop Publishing, Parts 1 and 2. *IEEE Annals of the History of Computing*, vol. 40, no. 3 (July–Sept. 2018), pp. 78–93; vol. 41, no. 2 (April–June 2019), pp. 29–41. [doi.org/10.1109/MAHC.2018.033841114](https://doi.org/10.1109/MAHC.2018.033841114) [doi.org/10.1109/MAHC.2019.2893731](https://doi.org/10.1109/MAHC.2019.2893731)
- [2] Computer History Museum. Desktop Publishing Pioneer Meeting, day 1, session 1, catalog number 102740205. The other sessions are linked. [computerhistory.org/collections/catalog/102740205](https://computerhistory.org/collections/catalog/102740205)
- [3] Katie Hafner. David Walden, computer scientist at dawn of Internet, dies at 79. *New York Times*, May 3, 2022. [nytimes.com/2022/05/03/technology/david-walden-dead.html](https://nytimes.com/2022/05/03/technology/david-walden-dead.html) (Generally accessible via a re-tweet from TUG: [twitter.com/TeXUsersGroup/status/1522061721757642753](https://twitter.com/TeXUsersGroup/status/1522061721757642753).)
- [4] Alex McKenzie. David Corydon Walden's five careers. *IEEE Annals of the History of Computing*, vol. 44, no. 3 (July–Sept. 2022), forthcoming.
- [5] David Walden. Brief biography of me, BBN, and the Internet. [walden-family.com/dave](https://walden-family.com/dave)
- [6] David Walden. Published books. Includes all his publicly-available books: on quality management, BBN history, and T<sub>E</sub>X. [walden-family.com/public/mybooks](https://walden-family.com/public/mybooks) [amazon.com/David-Walden/e/B0030R3E4A](https://amazon.com/David-Walden/e/B0030R3E4A)
- [7] David Walden. His own description of his involvement with IEEE and the IEEE Computer Society, including an extensive bibliography. [ethw.org/David\\_Walden\\_walden-family.com/ieee/](https://ethw.org/David_Walden_walden-family.com/ieee/)
- [8] David Walden. Travels in T<sub>E</sub>X Land. A compendium of all his works relating to T<sub>E</sub>X, including his column of the same name written for *The PracT<sub>E</sub>X Journal*. [walden-family.com/texland](https://walden-family.com/texland)
- [9] David Walden. TUG Interview Corner. [tug.org/interviews](https://tug.org/interviews)
- [10] David Walden. Publications in *TUGboat*. [tug.org/TUGboat/Contents/listauthor.html#Walden,David](https://tug.org/TUGboat/Contents/listauthor.html#Walden,David)
- [11] David Walden. Personal web site. [walden-family.com](https://walden-family.com)
- [12] David Walden and Karl Berry. David Walden interview for TUG. [tug.org/interviews/walden.html](https://tug.org/interviews/walden.html) (included in the *T<sub>E</sub>X People* book, [tug.org/store/texpeople](https://tug.org/store/texpeople))
- [13] David Walden and Frans Goddign. David Walden interview: A conversation about writing and learning and some books to read. *MAPS* 34, Najaar 2006, pp. 81–84. [ntg.nl/maps/34/16.pdf](https://ntg.nl/maps/34/16.pdf)

◇ Karl Berry  
karl (at) freefriends dot org

---

## Interview with John Lees-Miller

Paulo Ney de Souza

This interview took place on 24 July 2022, during the TUG 2022 online conference. John Lees-Miller is co-founder and CTO of Overleaf.



**Jérémy Just (JJ):** Interviews are always highly awaited in the TUG conferences. So now I'm pleased to leave the chair to Paulo Ney for an interview with John Lees-Miller, the co-founder of Overleaf.

John, Paulo, it's up to you.

**Paulo Ney de Souza (PN):** Thank you.

Welcome, John. The idea is we have a conversation about what you do and what your interests are and so forth.

So the first question I want to ask you is, what was your first contact with computing? Can you tell us how it happened, even if you can remember it?

**John Lees-Miller (JLM):** It was a pretty long time ago, but the first one I can remember was in primary school, so I must have been like seven or eight years old. We had some old Apple IIs in the library.

I remember they ran a small number of programs. One I can remember was this touch-typing game. We had to race by touch typing, and I was really bad at that. I could not get my head around how people could remember so many buttons. So I didn't get off to a great start with computing, but fortunately my school had a few other computers. And within the next couple of years, I remember there was a Tandy, which was an old make of MSDOS-compatible sort of PC, and it had Quick Basic on it.

Some of my friends and I started learning how to write simple programs on that. I think most of them were like text-based adventure games. So a lot of ifs and elses, that was about it. It was a good system, though. Pretty easy to learn.

**PN:** And when were you introduced to  $\text{T}_{\text{E}}\text{X}$  and  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ ?

**JLM:** Pretty late, to be honest. I think I started around 2006 because I was doing an internship, and that's something we'll probably come back to. I was interning at a company that's working on self-driving vehicles and we had some mathematical modeling to do.

And so most people were writing things up in Word and I wanted to try something else. So I think I started with LyX, the L-Y-X, the "what you see is what you get" editor. And I wrote up a few papers in that, and then eventually I realized that I could just write the  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  source, and then I switched to it after that. So, yeah, pretty late.

I will say it's amazing to me that we have so many students using Overleaf today at the undergraduate and sometimes even secondary level. I don't think I'd done anything other than write things out by longhand for most of my university career until that internship.

**PN:** You do have a PhD in mathematics, isn't it?

**JLM:** Yeah. I studied computer science as an undergraduate, and then I went on to do a PhD in engineering mathematics over at Bristol (UK).

**PN:** That's where your statistics part comes from. I read some of your papers and the ones on the game 2048 are very interesting. And some of the statistics there went beyond what I could understand easily.

To anybody listening to this interview, I do strongly recommend it; the game is addicting. It's called 2048. You can play it on the web, you can play it on an iPhone, or whatever.

But the papers by John are a little bit hard to get into. There is a higher level of math that's required to understand every step of it, but extremely interesting and also beautiful.

Let me ask you this: What moves you more, technology or coding?

**JLM:** Probably technology in general. I guess when I started out at Overleaf, I was pretty much doing the coding. But now Overleaf has grown to over 60 people, so my role has changed quite a lot. So I tend to work at a slightly higher level now. I occasionally get to touch the code, but mostly we have people who are professionals who do most of that now. So I try to operate a slightly higher level, looking at things like architecture and how the various bits fit together and long term vision for where we're going. I guess that's more on the technology side.

I feel like I should always say that my understanding of  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  is also very much a practitioner's

understanding. I did do Overleaf support, first line and second line and third line support for many years, and there I learned just how little  $\LaTeX$  I really know. We later hired some very capable  $\LaTeX$  experts like Lian Tze Lim and several more after her, who showed me that I know almost nothing, but I was still able to support some people who had basic  $\LaTeX$  questions over the years.

**PN:** You were an experienced coder before you met  $\TeX$  and  $\LaTeX$ ?

**JLM:** Yes.

**PN:** And that is what you used to start and build Overleaf?

**JLM:** Yes. I was already working as a software engineer, I guess going way back. I started it up pretty young. So when I was in middle school, after doing some Quick Basic, that was around the 2000 dot.com bubble, so anyone who could code anything could get a job. I actually got a part-time job even though I was like 13, writing [code]. I guess at that point it was Visual Basic rather than Quick Basic, but same idea.

So I started out there and then I did the computer science degree, in which mostly I studied the mathematical bits more than the practical ones, but I still worked quite a bit on the side. So I gained experience there. And I think depending on how far back in my history of blogs you go, there's a blog post from 2009 which I think is the earliest public record of Overleaf, or at least the idea behind Overleaf.

Back in the day there was this service called Etherpad, which is like a very early basic precursor to Google Docs. And so my collaborators and I, when I was a PhD student, were basically using that to write our papers. But then Google bought Etherpad and shut it all down. So before they'd done that, I'd actually written a bunch of scripts that sort of did crazy things like download the Etherpad and compile it, upload the PDF somewhere. So it kind of had all the components of Overleaf. But then as soon as Google came in and shut the whole thing down, I decided, well, it can't be too hard to start writing my own Etherpad thing. And so that's how Overleaf got started.

**PN:** You did meet John Hammersley before Overleaf, correct?

**JLM:** Right. So we were both working at a company called Advanced Transport Systems and we built the world's first computer-guided taxi system. It went on to be called the Heathrow Pod at Heathrow Airport, and was basically 20 computer-guided taxis that ran

on their own roads. So this was very basic compared to the stuff that Google and Tesla, well, not Google any more, Waymo and Tesla are doing today, but we actually managed to get something into production in 2011, after that opened.

John and I met at the company. I think we were in the systems research division. It was a very small company, I think even smaller than Overleaf is now, but there was still a systems research division. We worked on simulation software mostly. One of the good things about the company was that it was very open. So we worked with a lot of people in academia and that was another reason that we needed a good collaboration tool, so that we could work on our papers together, with all the people at university and in the company. That's still something that people do on Overleaf today.

**PN:** What do you consider more important in terms of skills, to have competing, complementary or similar skills as your partner, in developing an idea like Overleaf?

**JLM:** You definitely benefit from having complementary skills. Pretty rare to find someone who can found a business as a solo founder, but they certainly do exist. There are just a lot of different aspects of a business that you need to cover. So the split between John H and I was always that John looked at more of the business side of things, more of the commercial side, and I looked after the technology side and that worked pretty well.

John is actually pretty technical. He doesn't always let on, but he was also fairly technical. But he could do, much more effectively than I, many of the commercial parts of the business, like talking to our early customers in the publishing industry, raising investment; he definitely led all of that. That also freed up some of my time to focus on initially building the products and the prototypes and all of that, and then eventually going to manage the engineering team because there's a lot of hiring required to go from two people in 2012 to 60-odd today, ten years later.

**PN:** What would you name as your biggest challenge right now at Overleaf?

**JLM:** I think we have quite a few growing pains. So every time you double in size, something stops working, is my experience. Communication just gets harder and harder as you get larger. I'm always amazed that companies with thousands of employees can work at all. It's pretty challenging, even with 60 employees, to keep everyone on the same page, aligned. Something that we are still working on, I

would say. I think all companies still have to work at that, no matter how big they are.

Probably these days it's almost a cliché, but it's about communication and trying to get all these people with different skills to work together, and work together efficiently. We certainly benefit from having a very passionate group of people. Something I feel extremely lucky and proud of is that everyone at Overleaf is very passionate about either the technology or the impact that we have on making science communication a little bit faster and easier. And there's lots of people that just really like L<sup>A</sup>T<sub>E</sub>X and T<sub>E</sub>X. I feel it's a very special company to have so many people [who] are really passionate about the mission that we have.

**PN:** Is T<sub>E</sub>X Live a big stone on your path? T<sub>E</sub>X Live changes?

**JLM:** Well, we're extremely grateful to T<sub>E</sub>X Live. Without T<sub>E</sub>X Live we would really struggle to maintain any kind of compatibility between Overleaf and the offline world, and that's something that we definitely try to do. Ideally, if it works offline, it should work on Overleaf, and vice versa. We have very few things that are Overleaf-specific.

That said, yes, T<sub>E</sub>X Live is a challenging target. The fact that it's always a moving target is probably the biggest challenge we have because we have to cut a release and ship that to millions of people.

Right now, T<sub>E</sub>X Live 2022 is in the testing phase [at Overleaf]. So basically what we do is we take a snapshot of T<sub>E</sub>X Live sometime pretty soon after the official release and then we run it against all of our gallery projects. Overleaf has a template gallery where there are around 10,000 projects that people have submitted that are licensed appropriately for us to be able to use them for testing. We basically just run all of those and see what breaks. And if something very serious breaks, then we have to figure out how to patch our image to try to fix that without breaking too many other things. So, yeah, we definitely do find the T<sub>E</sub>X Live release process a bit of a struggle and we are in contact with the T<sub>E</sub>X Live team.

Usually by the time we find one of the problems with our gallery, though, we discovered it's already been reported by someone in the open source community. So it's pretty rare that we actually find a new bug, but we certainly hit most of the bugs that everybody else hits. So hopefully, if all goes well, I think it's in the final round of testing and the key guy, Eric, is on holiday for a week, but when he's back, I think we're going to hit the publish button and that will become available.

**PN:** Would you allow me to ask a question that I asked JH last time around?

**JLM:** Sure.

**PN:** Will we be able to use Overleaf on our iPhones and Android as a native app?

**JLM:** I can neither confirm nor deny that there are plans to do that, but a native app is definitely on our radar.

One of the things we're working on at the moment is that we've moved the entire editor, the actual sort of thing that you type into, from a piece of software called Ace to a newer piece of software called Code Mirror Six, which is a big project. So currently I think 20% of users can see the new editor because we roll out most things incrementally now, because if we roll out to everyone at once, we very quickly get overwhelmed if there's anything wrong with it. So that should have much better support on mobile. It uses a very different approach to actually making the editor work, which plays a lot better with mobile browsers.

So at least for this year, and probably next year, our focus is just on getting it working better on web browsers, on mobile devices. But I am sure that the day will come when we do make the jump into native apps. I just don't know when.

**PN:** Thank you very, very much. This was a wonderful conversation and hope to have you here back again sometime soon.

**JLM:** Great, thanks for having me.

**PN:** Thank you.

**JLM:** If anybody has any questions, I'm happy to stick around.

**PN:** Thank you. If you want to join the floor, just raise your hand and we still have time. So if you have any questions for John, raise your hand and we will bring you in.

But first, John, I don't know if you can talk about all of this, but how much of the company is remote and how much is it physically local?

**JLM:** Yes, I can definitely talk about that. It's in our job ads. We are now basically all remote.

We have staff in, actually I've lost count of the number of countries now, but most of our staff are in the UK, the US, some in Canada; we have a growing number in Germany, and places like France and Portugal. So we are all remote.

We're not a fully distributed team though, so we do try to constrain our hiring to a range of time zones. That means that people have overlap. So

basically we have what are called core hours, which are 2:00 p.m. to 5:00 p.m., UK time. So if you're in the UK, you tend to have a slightly quieter morning and if you're in the US, you have a slightly quieter afternoon to get on with things, and then all the meetings in the core hours. That's how we manage that.

**Frank Mittelbach (FMi):** Can you hear me, John?

Your last statement about your gallery testing and everything made me wonder if we should talk with you about the possibility to align that with the L<sup>A</sup>T<sub>E</sub>X releases. Right now you probably have seen our sins in various respects, because every half year we have the L<sup>A</sup>T<sub>E</sub>X releases quite heavily sort of improving stuff. But also in corner cases breaking stuff, which is a natural thing if you have millions of users out there.

You may or may not know that we run a development version of L<sup>A</sup>T<sub>E</sub>X which is available to everybody. The intent is that developers and users can make use of that before we actually hit the street. I don't know exactly how you do your gallery testing, but if you have this as a sort of process, it might be quite helpful for everybody if that process could encompass running the development release at the late stage before we switch over. Then you would probably find bugs that otherwise will be found by you when you [take] the full release.

I think by the end of the day that would save you effort as well as saving everybody else pain. And if we have a sort of feedback loop this way, I think that could be very beneficial to everybody just to bring this up as a potential sort of alignment between our team and Overleaf there in the future.

**JLM:** Yes, we would be very happy to talk about that. As you say, we will hit the bugs anyway. I think we have talked about it at a previous TUG, and certainly something we're still interested in doing, setting up some kind of L<sup>A</sup>T<sub>E</sub>X CI, continuous integration, where we can just run it against our large sample of documents and report back on error rates, basically what we do internally now.

The other thing I'd say would be interesting is that we track performance across the T<sub>E</sub>X Live releases as well. One of the things that we see is that every year things tend to get slightly slower. So we would also be very interested in trying to set up some kind of benchmark test set because we're not sure that our gallery test is particularly representative as a benchmarking tool. But probably we could set up some sort of thing we would be more comfortable with as a benchmarking tool that could also alert on things like performance regressions, which at the moment I think there's not a lot of visibility into.

**PN:** You have a question from Jérémy on the chat.

**JJ:** Can you tell us about the computing power behind Overleaf? How many computing nodes, and what kind of nodes?

**JLM:** I can say a few things, and it ties into the slight performance decreases every year as more and more servers are required. Overleaf is hosted on the Google Cloud platform now. When we did the integration with ShareL<sup>A</sup>T<sub>E</sub>X in 2017, we were hosted on essentially every cloud service, which meant that if any cloud service was down, Overleaf had some kind of problem. So one of the things that I've been working on for the last five years is consolidating all of our hosting on Google Cloud.

The number of nodes, I don't think we give an exact number, but I can say it is many, many hundreds of cores, continuously compiling people's L<sup>A</sup>T<sub>E</sub>X; and there's also some fraction of that for running the service. But our L<sup>A</sup>T<sub>E</sub>X compilers are by far the most compute-hungry thing that we do.

**JJ:** Thank you.

**PN:** Thank you very much, John, for joining us, taking time out of your Sunday for this.

Welcome to TUG, and please come back and join us at other years.

**JLM:** Great. Thanks again for having me. Thanks, everyone. Have a good rest of the day.

---

## Interview with Boris Veytsman

Paulo Ney de Souza

This interview took place on 23 July 2022, during the TUG 2022 online conference. Boris Veytsman has been a member of the TUG board since 2010, and President since 2017.



**Paulo Ney de Souza (PN):** Let me start with something very small. How do you pronounce your name? Veytsman [pronounced like “day”] or Veytsman [pronounced like “night”]?

**Boris Veytsman (BV):** “Vaytsman”. The German translation is “Weitzman”, which would be pronounced like “night”, but since I’m not German . . .

**PN:** That’s interesting, it took me too long to find out.

Can you tell me how you had your first contact with computing? Can you remember it?

**BV:** I was born in what was then the Soviet Union, which was several decades behind the United States in computers at that time. So I first read about computers when I was, I think in 7th grade, or something like this. I really loved the idea. But for the first year or so it was mostly pen-based computing. I wrote down programs, I think in Pascal. Then I played computer. I computed it with a pen.

Then when I was at 8th grade, I won in the Programming Olympiad and I was invited to the so-called Summer School of Young Programmers in Novosibirsk, in the Siberian Department of Academy of Science. They had really great computers and they had the School for Young Programmers, which was the brainchild of Academician Yershov, who was a great authority in teaching programming. So there it was 1980 exactly. I remember the date because it was Olympiad days and when a lot of people were watching sports, I was watching BESM-6 (БЭСМ-6), the great Soviet computer.

This was a time when computers would take several rooms and they had a lot of terminals and teletypes. I don’t know, have you seen teletypes? This was an overgrown typing machine where you type something, then the computer would type to you, and you basically talk on a big roll of paper. And the thing about this was that they didn’t have enough terminals, they didn’t have enough teletypes. And so they had a very strict rule. There was a number of levels. School students (I was in 8th grade) were on the lower level. Then there were undergrads, then graduate students, and so on and so on, up to the head of the institute. And the idea was that if you are working on a terminal or teletype and somebody from the higher rank would come and there were no free spaces, then somebody from the lower rank should stop what they were doing, save their work, and release the terminal for somebody at the higher end. And you can understand that since I was a school student, I was the lowest rank at all. So I was bumped many times and I was okay. I think, when I grow up I will have a teletype at my home and nobody will bump me from this teletype. And now basically what happens is now I have this phone which I can connect to any computer in the world. It by itself has probably more computing power than back in my childhood.

So basically when I was a kid, when I get this teletype in my room, I would be absolutely happy and I would not want anything else. Now I have so many terminals around me, sometimes I think I’m happy as much as I would be when I was eighth grade.

**PN:** I can give you a word of consolation, because I can relate very well to your story, because I was introduced to computing inside the military. So there they would just show you their rank, and you had to leave. It was only in Berkeley that I was able to have access to terminals on a standard basis. So I can relate to that story very well.

We got across this issue, this next issue, several times in this conversation, and crossing from Odesa<sup>1</sup> to Novosibirsk was no easy task. You would get invited normally if you just won the Olympics. It’s a long way. It’s a very long way.

**BV:** Yes, it’s a very long way. And 1980, this was the year of Moscow Olympiad. And so what happened was that you could not buy any plane tickets because all aircraft were taken to serve the Olympiad, and you could not buy any railway tickets except the

---

<sup>1</sup> Editor’s note: Since the Russian invasion of Ukraine, many people use the Ukrainian spelling “Odesa” instead of the Russian spelling “Odessa”; we follow suit here.

most uncomfortable. I think it was three days travel in probably the worst possible railway car you can imagine. But I was much younger, and that was fine.

**PN:** Did you end up meeting other kids from the Olympics there?

**BV:** The kids in the Programmer Olympics, in which most of my school participated, were interested in general because it was exactly the days of the Olympic Games, and lots of people were watching. But somehow, for those of us programming, especially for those who saw computers for the first time in their life, we would rather be behind the terminals than behind the TVs.

**PN:** This was most like a summer school?

**BV:** It was a summer school. It was a very interesting idea by Novosibirsk people. They started to do an experiment. What is the optimal age to teach kids computing? And by computing, I don't mean *using* computing, no, but active computing, when you teach them programming. They have very interesting ideas about special computer languages for the young kids, and so on. And they had a lot of school students.

If I'm not mistaken, the big conclusion of the research was that the optimal age to teach kids coding is fourth grade. Because if you tried before, they don't have yet enough skills and not yet enough attention to start coding computers. And after this, it's like language. It goes downhill after fourth grade. So from their point of view, I started to code actively at 7th or 8th grade. From their point of view, I was probably too old to become a good programmer.

**PN:** Well, I'll keep your number fourth grade in mind, because I just became a grandparent.

How and when were you first introduced to  $\text{\TeX}$  and friends?

**BV:** I'm a physicist by training. So one of the things that physicists do, they write a lot of math, text and so on. So when I was first working, it was the Soviet Union, and then it became Ukraine in 1990s.

In the 1990s we had computers, and we had a very interesting scientific word processor, which was called ChiWriter. Probably some old hands like you and me remember this thing. So I wrote my first paper in ChiWriter. I remember that in many cases it didn't have enough fonts so it had a font editor—you would make your own mathematical font and your own mathematical symbols.

Then when I got a postdoc position in the United States, I wrote, I think, a paper or two in Microsoft Word which was not very pleasant, mostly because

it had a lot of equations. And then a friend of mine who was a young professor at the same department said, you know, there is this strange thing called  $\text{\TeX}$  and it's much easier to write math in it. It was something like 1994, and there was this program Oz $\text{\TeX}$  for Macintosh which had really ugly rendering of text and math on the screen. It used DVI files and the DVI viewer was really bad so the only way to see how beautiful is your text was to send it to the printer. The preview was bad but I started to play with this and I thought, well, it's programming. I know how to program and now I see my text as a program and I can just write down comments how my text should look, and the concept was so attractive for me. I started to work on this.

**PN:** So you were already an experienced programmer by the time you met  $\text{\TeX}$ ?

**BV:** I wasn't a programmer. For a couple of years of my life I was what was called a scientific programmer, and after these two years I understood that I really love programming as a hobby, but working eight hours a day programming is probably not for me, so I'm not a programmer. I really love programming if I'm not made to do it from the morning to the evening; let's say that I'm an amateur programmer.

**PN:** I have to ask because you are responsible for a great number of CTAN packages and related work, and a lot of them in support of publishing.

**BV:** Yes.

**PN:** How did this relationship grow up? There is a huge lot of CTAN packages that have no relationship to publishing, as you know, like  $\text{\TeX}$  by itself, but with your packages you see a vein of publisher support.<sup>2</sup> How did this happen? Is it because of your own interest or ...

**BV:** I don't know. I think sometime in the beginning of 2000 I started to know  $\text{\TeX}$  was a hobby for me. I wrote a couple of packages and then I saw this, then somewhere I saw a possibility to publish an ad for  $\text{\TeX}$  consulting in *TUGboat*. Well, it's a nice hobby; maybe they can pay me for this hobby. And I published an ad, and somebody called me and said, we need this sort of style. Why not. And somehow it went from there, and I guess most of it was ads in *TUGboat*. There was word of mouth, and, you know, as I said, programming and  $\text{\TeX}$  programming as a hobby is something that you do, I don't know, on the weekends and evenings. It's probably one of the most fascinating things you can do.

<sup>2</sup> <https://ctan.org/author/veytsman>

**PN:** You mentioned your original training as a physicist . . . It looks like that you always wanted to have a foot inside industry and a foot inside academia.

**BV:** The thing is, as one of my mentors said in a similar situation, I have a rather short attention span. So there are a lot of people, and I deeply respect the people who can do the same thing for years, for decades, and they just do and do and do, and for me it was always, okay, I understand how to do it, I probably do it well, what else can I do with my life? So I moved and moved. I have to say that in my life, I did all the things from designing vacuum cleaners to proving theorems and doing a lot and a lot of things in between.

So I wouldn't say that I want to be both in academia and industry. I would rather say that I want to be in as many places as possible. Yeah, of course it has its own drawbacks, because if you do something from year to year you probably can get a lot of interesting things and a lot of success. But unfortunately after several years I usually become bored with what I'm doing and try to change it.

**PN:** I mean for somebody that wanted to have their feet in several different places you must be quite happy right now, having worked for George Mason for so long now, inside CZI [Chan Zuckerberg Initiative], and also president of TUG, with probably one of the longest tenures I have seen.

**BV:** I'm really surprised that it was long. I need to think about.

**PN:** What can you share with us about the work inside CZI? Is it gratifying?

**BV:** I joined CZI at the point when I decided that my main job was becoming boring. At the time I was working for, it was ITT, it's now L3Harris, I think, and it was a very interesting job because we created and maintained what's called ADS-B, basically one of the primary systems for air traffic safety. So I loved to say for many years that if you fly over the United States and don't collide with anybody, then I probably have some . . . I certainly would not say I am completely responsible for your flights not having collisions, but perhaps you may want to thank me in part. But at this point, instead of building the system, we started to maintain the system and it's much less interesting, it's much more routine. So I said let's change everything I did and I moved to Chan Zuckerberg Initiative which had a very interesting idea to completely cure, manage, or prevent all diseases before the end of the century.

As one of my hobbies, I did some biomedical research for many years. I said okay, let's try to do

this and I started to work. I had been working there for five years (another long stretch). My primary work was in what is called science of science. The idea was to try to understand how people in biomedical fields read literature, and how can we improve the reading literature understanding, and so on.

I worked in this area, then Covid struck and people did not know what to do. And I somehow got recruited in the Covid effort in CZI and our sister organization, Chan Zuckerberg Biohub, and we started to think about mathematical modeling for Covid, and I recalled my physicist training and started to write models. We published several papers, including papers in journals like *Phys. Rev.*, *Phys. Biology*, *Scientific Reports*, . . . Basically, it became very fascinating. So right now I've somehow drifted back into physics, biophysics and so on.

By the way, it's one of the secrets if you have a long tenure at the same place. (I have been with Harris for 20 years and have been with CZI five years.) If you have a short attention span and are interested in new things, one of the things you can do, you can reinvent what you are doing even on the same job. And I did this several times at Harris. I'm trying to do this at CZI and basically even if you stay, it's better to think that you are changed.

**PN:** Let's change then to one other job, which is president of TUG. I remember that you have stated that your goals were to keep TUG intact, relevant in a changing world of that setting when you started. How do you gauge what you have achieved on that goal and what still needs to be done?

**BV:** Okay, there are two things: you want to keep the lights on, you want to keep going; and you want to go into new directions, and T<sub>E</sub>X is a huge world. There's a lot of moving things, that it is difficult to move them all. TUG has lots of people with quite different ideas of what they want. So I think that at the first part of this, keeping the lights on, what we are doing, what we all are doing is good. We have distributions coming out, we have members coming in. I have spent some time trying to stop the bleeding, the constant lowering down the numbers of members. It's very difficult to do.

Going into new directions, it's very difficult to make volunteers to work on the problems unless they think they are most important. So I tried to explain and try to influence. I think we did something in two directions I think are most important. This is accessibility and electronic books. We have now much more progress. I don't think that a lot of this is due to my work. I think that there are much better people and people with much better visions,

and they did this, and many of them have talks in this conference. I hope that some of what I was doing, talking and trying to convince people that this is what should be done, was helpful. I hope that it helps to move a little bit in this direction. But I don't have to say that my contribution was substantial or even significant. I want to hope that I did something to convince people to go there.

We also have a lot of success in electronic formats. There are several things that people are doing in several directions. I really love the ebook package. I really love this new HINT format which has a lot of potential. I hope that it could be adopted. I hope its business window is not closed and people are not already entrenched in older and worse formats. And there are lots of developments in this area.

So again, let me answer your question whether I was able to do what I wanted. I would say that there is a lot of movement in the right direction. I don't think that my role in this movement was large enough, but I would like to think that I did some pushing there. But one of the things you get when you work at TUG, when you have so many very smart, very dedicated, very knowledgeable people use it, you learn humility. You just understand that almost anybody, our developers, is much better than you. You just must be humble and understand them.

**PN:** If you allow me, I'd like to change the subject a little bit as well again. You are from Odesa.

**BV:** Yes.

**PN:** Lots of people around the world do not know it very well. I have been to it. I went to Kyiv for a conference in symbolic computation in the late eighties and had the chance to do a very small trip because my wife is a nurse and Crimea helped us — the first place of nursing — and the fastest place to get to at that time was Odesa. And I have a very nice remembrance of that time. Right now, a city which is besieged, and being a short lead from Mariupol, Crimea, and all these places that we hear every day. Do you have friends and families in there that worry you?

**BV:** Yes, I have friends there. My immediate family is here [in the US], but I have a lot of relatives and I talk daily with people from Odesa. It's incredibly sad what's going on and I never even thought that it would be like this. Especially, I don't know what is your impression about it. Did you like it when you were there?

**PN:** The meeting, it was my first time inside the Soviet Union. People here in the US would not even travel to the Soviet Union at that time. And

I had to go to other places to *then* pick up a plane to go there. The computing was very rudimentary, especially symbolic computation. So everybody was very eager to talk to me. So I felt like I knew more than I really knew, and I felt that I could help a little bit more than I really could.

I had just started at the time playing with Maxima here and Richard Fateman had brought the Maxima code and had installed it on the VAXes at Berkeley. They were all very eager to experiment and we did run many computations with them over (postal mail) letters, letters that were written and say oh, we would like to try the integration on this particular function here in Maxima and see what happens. And then I will try and send them the result by mail, not email. And so this visit to there was very nice. I remember buying CDs for a dollar which were very, very good, Philharmonicas that were exquisite music. And while here in the US we would pay \$12 for a CD, \$14 for a CD. And it was a nice trip.

**BV:** It's interesting that you have been in at the beginning of Maxima; great program and I still use it almost daily. To tell the truth, it's my favorite computer algebra system. But speaking of Odesa, besides it's being beautiful and the city of my childhood, what I really like about the city, it's one of the things that Tony Judt — a philosopher and very interesting writer — called Corner Cities. I don't know whether you know this notion. It's basically a place where several different cultures come together, coincide, intermingle in the very old days, much before me, anybody from a difficult origin — Ukrainian, Russian, Greek, Jewish, Yiddish — it was a multicultural place in the best meaning of the city. It was much less than this in Soviet times when Greeks were deported, and so on, but still it started to become more and more multicultural and more and started to return to its old idea of intermingling of different cultures and different things.

What's going on now, I mean this aggression from Russia, is some sort of reaction to the idea of multicultural free life, because for me the idea of freedom, of openness, and so on, is something which really related to the city of my childhood. It's something which was important for me.

Nice that you have been there and nice that you played with Maxima. It's old days, it's a great program and I really love it. I work with all, I think, or most commercial computer algebra systems, and now I return to Maxima.

**PN:** When I found out that it could do integrations and that later on I found out there is an algorithm

to do integrations. If a function has an integral, it *will* compute it. It doesn't tell you how long it will take, but it *will* compute it. And that was one of the things that meant the most in computing for many years.

**BV:** It looked like magic. And since I was trained to do this integration in the old Soviet manner, when it would probably be considered cruel and unusual here, when they just made you to work and work and work on this. And when they saw how Maxima was doing. . . .

**PN:** Do you think that this aggression from Russia could have been averted? Like more integration, more transit in between the two societies that were built after the breakup, more that there was no way that it was written on the wall that the autocratic government in Russia would only be to this enemy.

**BV:** I'm not a specialist in this, and I have . . . It's very difficult to say. My gut feeling is that Russia had enough problems and enough of very difficult things inside that it was very tempting for them to try to solve these problems on the path of war and aggression. So my gut feeling is that, at some point, it was probably inevitable at some level. But again, I'm very much afraid of saying something in the field which I'm not competent, so it's just my thought.

**PN:** We've run out of time. I wish you the best for especially for your family that stayed behind. That probably would be the worst thing that you can have with us and we share concerns with you. I guess every day I have at the back of my mind people that I have known there, and I have very long time, and we should do more about this in the US.

I wish you a very successful future at CZI and also with that, because your goals for accessibility and ebook are just top priorities right now. And thanks for taking the time to talk with me.

**BV:** Thank you very much for very interesting questions and very thoughtful talk. I really enjoyed it very much. Thank you very much. And let me use this occasion to tell you how I'm grateful for everything you have been doing for this conference. It would be absolutely impossible without your huge, huge, huge work.

**PN:** It was nothing. Thank you. All right.

---

## The future of technical documentation starts with its *recent* past

Carlos Evia

### Abstract

This keynote presentation addresses how recent trends to align technical documentation practices with “developer-friendly” workflows may be detrimental to documentation authors and their users. A proposed solution is in the recent past of technical documentation as a discipline, where tools and ideas rooted in structured authoring and markup, reuse, and personalization can still provide solutions to present—and future—needs related to technical content.

### 1 Introduction

As I was preparing this keynote presentation, I realized that a subtitle—or even alternative title for it—should be “We all owe something to a former IBMer”, or the more complimentary “Damn, those former IBMers were right”.

I do love talking about the future of technical documentation, but I also enjoy talking about its past, and particularly its *recent* past. Technical documentation is still a very much needed and important product, or genre, of technical communication. As a bigger umbrella term to contain the processes of developing and conveying technical information from experts to a non-technical audience (or the dreaded “laypeople” noun), technical communication is pretty much concerned with documentation, but mainly as part of a universe of genres or products that require that mediation between user and expert that characterizes the job of technical communicators.

And here is where I will go back to the recent past of technical communication. Ten years ago, the Adobe Technical Communication Suite (TCS) team distributed on several social media channels a video titled “Future of Tech Comm”. The video used stop-motion animation and fast-draw techniques to summarize the features of “Adobe’s Tools and Services” for technical communication. As a pair of rapidly animated hands assembles Lego pieces, the video’s narrator describes that “for some, [the future of technical communication] is all about more and more structured content and the ability to work faster and smarter with XML and DITA constructs”. Approaching the end of its 2:30-minutes runtime, the video claims that “it is most certainly an exciting future to be in” [1].

## 2 The Darwin Information Typing Architecture

At this point we encounter a first batch of former IBMers, as the main engine behind the future of tech comm heralded by Adobe in that video was DITA (Darwin Information Typing Architecture). This should be interesting for you as L<sup>A</sup>T<sub>E</sub>X users, because it involves a markup language. DITA is an open standard for structuring and publishing technical information. Sounds familiar? DITA was—big surprise—developed by IBM in the late 1990s, and is very much alive as an open standard maintained by the non-profit consortium OASIS (Organization for the Advancement of Structured Information Standards).

In an interview for the website “DITAWriter”, Don Day, a former IBMer who was one of the original developers of DITA, chronicled the origins of his XML experiments while working for IBM in the last decade of the 20th century as follows:

With the advent of XML as a new markup standard in 1998, the Customer and Service Information (C&SI) group began adopting a Tools and Technology mantra under Dave Schell who was the strategy lead. By 1999, Dave was aware of my participation as IBM’s primary representative with the XSLT and CSS standards activities at the World Wide Web Consortium, and I delivered a presentation at a formative meeting in California that forecast the possibility of XML to solve IBM’s still-lingering problems with variant tools and markup usage [2].

DITA consists of a set of design principles for creating “information-typed” modules at a topic level and for using that content in delivery modes such as online help, technical documentation, and product support portals on the Web. Day explained that, when naming the standard, DITA “represented a great deal of messaging in a compact and memorable acronym”:

- **Darwin:** for specialization and how things could “evolve” from a base;
- **Information Typing:** for representation of knowledge as typed units;
- **Architecture:** a statement that this was not just a monolithic design but an extensible tool that could support many uses [2].

IBM eventually donated DITA as an open standard, which is currently maintained by OASIS. DITA, however, “has evolved substantially since that initial donation to encompass a very wide scope of requirements indeed” [7, p. 6]. At the OASIS DITA Technical

Committee, the standard continually evolves with the purpose “to define and maintain the Darwin Information Typing Architecture (DITA) and to promote the use of the architecture for creating standard information types and domain-specific markup vocabularies” [8]. At OASIS, a small army of former IBMers (including Kris Eberlein, Eliot Kimber, and Michael Priestley) has kept the DITA standard evolving and with very healthy adoption and usage figures.

Just as in L<sup>A</sup>T<sub>E</sub>X we declare a document class to start a file, in DITA we can use different topic types that have specific semantic elements to structure, and later publish, technical information. The literature focuses on three topic types that “represent the vast majority of content produced to support users of technical information” [4, p. 7]: concept, task, and reference, which Pringle & O’Keefe define succinctly as follows:

- **Concept:** contains background information and examples;
- **Task:** includes procedures (“how to” information);
- **Reference:** describes commands, parameters, and other features [9, p. 235].

For authors of technical documentation, these foundational topic types provide constraints and structures beyond a presentation-oriented template. In DITA, authors can create consistent topics to assemble collections of information with elements that can be reused even at the phrase level. For example, a concept could be an introduction to a particular software package, while tasks can provide instructions on how to install and use the software package, and a reference topic can list common extensions and tools associated with the package.

In practical terms, DITA’s topic types include XML tags for content “moves” or strategies (such as a short description, steps, and examples) frequently used in technical publications. Pure XML does not provide a defined set of tags, but DITA does offer a catalog of elements and attributes relevant for technical communicators.

The *Darwin* component of DITA is one of its main “selling” points. DITA is customizable for specific situations that will still keep it as part of the standard. Maybe the element types included in the default *task* topic type are too generic for CompanyX. The company can then *specialize* the task type to create, rename its own structural elements that will still validate in DITA-aware tools.

And that is the recent past of technical documentation: workflows and tools based on DITA became mainstream and enabled practices such as

single sourcing, modularity, and multi- or omnichannel publishing. Let me spend a few minutes describing those, and you will see that you can do similar things with  $\LaTeX$ -based workflows for academic and scientific publication.

## 2.1 Single sourcing

A team can have a common repository of topics, or even element types (a common legal disclaimer in a paragraph) that many files can use. By referencing the single source, all files that mention it would be automatically updated if there's a change in the source. Much better than copy-paste. We can, of course, do some of that with  $\LaTeX$ .

## 2.2 Modularity

Nothing new for  $\LaTeX$  users here, but in DITA authoring a whole “document” can be composed with pieces from different sources. In the particular case of DITA, most of these aggregation processes will use a file type known as a map, which includes hyper references to topics and other resources (internal or external).

## 2.3 Multi- and omnichannel publishing

Like DVI on steroids, multichannel publishing is one of DITA's key features. By default, the DITA Open Toolkit can publish to PDF, HTML, Markdown, Eclipse Help, HTML Help, and other formats. Similar to  $\LaTeX$ , the user community has also contributed with plugins that enable publishing to many other formats. Surprising no one, many of those publishing pipelines involve a stop in  $\LaTeX$ -land.

Omnichannel publishing is even more interesting, as it enables content-as-a-service approaches that “serve” DITA topics or components via APIs. DITA is particularly good at this because of the semantic value that its XML tags and attributes can provide as metadata for filtering and customization.

## 3 The present

And this is when we get to the present of technical documentation. Adoption numbers and success stories should not hide that the evolution of technical documentation takes place on a slightly rocky path; even in practitioner circles, there has been pushback and criticism against XML and its relationship with technical communication. In blogs and social media exchanges, some practitioners have questioned the status of XML, and DITA, as the main markup language for information products. While acknowledging DITA's effectiveness as a replacement for large user manuals in complex industries, a few authors lament that “this form of structured content can feel

cold and clinical, especially to those from the editorial or marketing side of content” [10, p. 20]. Others argue that in the world of computing code verbose languages are becoming obsolete, but intelligent content still relies on XML and its nested tag structures.

Those are valid concerns. DITA, as it evolves as an open standard, needs to address them and learn from its users. And here we have another former IBMer to the rescue. Michael Priestley, one of the key architects of DITA back in the late 1990s, has been working on a simplified version of the standard known as Lightweight DITA (LwDITA). As a disclaimer, I was involved in the development of LwDITA and worked closely with Priestley for many years. Although I am not an active member of OASIS any more, I am still a DITA and LwDITA peddler (see this keynote presentation as an example!).

LwDITA is a topic-based architecture for tagging and structuring intelligent content using flexible markup options [3]. Lightweight DITA aims to streamline the DITA authoring experience by presenting three formats for content creation:

- **XDITA**, an XML format with a subset of DITA elements that can be used for validated authoring and complex publishing chains;
- **HDITA**, an HTML5 format that can be used for either authoring or displaying content;
- **MDITA**, a Markdown format with a subset of XDITA elements that can be used for maximizing input readability while maintaining structure in content.

An author does not need to use all three “flavors” at the same time to adopt LwDITA. They can work in HDITA all the time and they would still be using LwDITA. Authors can live in an MDITA environment without XML or HTML tags and would still be using LwDITA. All three LwDITA formats are compatible with each other and with DITA XML. For a team of authors with diverse technical backgrounds and communication skills, the different formats of LwDITA allow collaboration and content exchange in a centralized solution. For example, CompanyX can hire a technical writer to create instructions in XDITA (based on XML) while a marketing professional writes a description of the app's features in HDITA (based on HTML5), and an engineer uses MDITA (based on Markdown) to create a basic API reference. All their topics are treated as DITA and can take advantage of the standard's reuse, filtering, and single-sourcing capabilities.

Now, yet another former IBMer has come out to warn users about going too lightweight. Michael

Iantosca warns about the impending doom of Markdown and reStructuredText (another lightweight authoring format that has become popular with developers and some technical authors). Iantosca [6] states that the degree of granularity and flexibility that what he labels as cognitive content requires cannot happen with lightweight languages. Iantosca defines cognitive content as follows:

[...] a strategy, an architecture, and an operational model. It enables dynamic, machine-based discovery, mining, analysis, retrieval, assembly, and delivery of non-linear content objects using advanced semantic technologies that rely on predictive relationships between content objects and inbound signals [5].

### 3.1 Conclusion

And that's where DITA excels. But this requires going back to the recent past of technical documentation and to the future of "tech comm" that Adobe heralded in its promotion video. Let lightweight languages work in a LwDITA-like environment and save the intense structure for DITA.

### 3.2 What does this mean for $\text{\LaTeX}$ users?

I am a big proponent of ventilating silos, and here I see Markdown as a good bridge to connect the academic and scientific typesetting of  $\text{\LaTeX}$  with the technical documentation of DITA and LwDITA. The `markdown` package is your friend if you need (or want) to use modules from a single source repository. If you need stronger features to achieve cognitive content or omnichannel publishing, then you can move to XDITA and full DITA XML. That's how I see the future of technical documentation.

### References

- [1] Adobe TCS. Future of Tech Comm, Jul 2012. <https://www.youtube.com/watch?v=dSdhnyDF0YY>
- [2] DITA Writer. Don Day and Michael Priestly on the beginnings of DITA: Part 2, Oct 2018. <https://www.ditawriter.com/don-day-and-michael-priestly-on-the-beginnings-of-dita-part-2/>
- [3] C. Evia. *Creating intelligent content with Lightweight DITA*. Routledge, New York, NY, 2019.
- [4] J.T. Hackos. *Introduction to DITA: A user guide to the Darwin Information Typing Architecture including DITA 1.2*. Comtech Services, Denver, CO, 2011.
- [5] M. Iantosca. A future powered by knowledge graphs. BrightTALK. <https://www.brighttalk.com/webcast/9273/525482>
- [6] M. Iantosca. Mark-Duh? The impending doom of Markdown and reStructured Text, Mar 2022. <https://thinkingdocumentation.com/blog/f/mark-duh-the-impending-doom-of-markdown-md-and-restrucured-tex>
- [7] E. Kimber. *DITA for practitioners*. XML Press, Laguna Hills, CA, 2012.
- [8] OASIS Open. OASIS Darwin Information Typing Architecture (DITA) TC. [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=dita](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita)
- [9] A.S. Pringle, S. O'Keefe. *Technical writing 101: A real-world guide to planning and writing technical content*. Scriptorium Press, Research Triangle Park, NC, 2009.
- [10] S. Wachter-Boettcher. *Content everywhere: Strategy and structure for future-ready content*. Rosenfeld Media, Brooklyn, NY, 2012.

◇ Carlos Evia  
Virginia Tech  
Blacksburg, VA 24061  
USA  
cevia (at) vt dot edu  
<https://carlosevia.com>

## A stroll through computer history at the CHM

Dag Spicer



Dzhou, Wikipedia, CC BY-SA

The Computer History Museum is the world’s leading institution for the preservation, display and interpretation of computing history, from antiquity to the present day. It began life as the DEC (Digital Equipment Corporation) Computer Museum in the mid-1970s when DEC co-founder and president Ken Olsen and DEC VP of Engineering Gordon Bell rescued the legendary 1951 MIT Whirlwind computer from the landfill. Its modern Silicon Valley instantiation began in 1995 when Gordon Bell and entrepreneur Len Shustek hired Dag Spicer, then a PhD student in history and electrical engineering at Stanford University, to manage the collection.

Starting with some 1,200 seminal objects, many from the first generation of computing, CHM’s collection now includes over 120,000 items, comprising hardware, software, ephemera, media, and over a linear mile of computer-related documentation. The collection can be searched.<sup>1</sup> About 1% of this collection is on permanent display in CHM’s award-winning exhibition, “Revolution: The First 2000 Years of Computing”, and we’ll be taking a look at some of the canonical objects from “Revolution” during this talk. These will include the Abacus,<sup>2</sup> the Antikythera Mechanism,<sup>3</sup> ENIGMA,<sup>4</sup> SAGE,<sup>5</sup> IBM System/360,<sup>6</sup> Cray-1,<sup>7</sup> Xerox Alto,<sup>8</sup> Apple-1,<sup>9</sup> an

original Google server rack,<sup>10</sup> and more. The presentation is application centered so we’ll focus on what problems people were trying to solve at the time and how that can compare with what came before and what came after.

A second major exhibit, “Make Software: Change the World”,<sup>11</sup> celebrates the role of code in our lives by focusing on seven application stories: Photoshop, Wikipedia, Texting, MRI, Car Crash Simulation, and World of Warcraft. The idea is that each of these types of software often affects us, either directly or indirectly, as we go through life. It’s a near statistical certainty that most advertising images you see, for example, have been processed, to some degree, in Photoshop.

Beyond its public exhibit functions, CHM is also a world-class center for research into the history of computing. At its research center, anyone wishing to access the Museum’s massive archive of computing literature and reference material has only to make an appointment. There are no restrictions or costs (unless you request scans).

In-person visitors to the Museum can also enjoy three unique “retrocomputing” demonstrations of vintage technology of the 1960 DEC PDP-1,<sup>12</sup> the 1959 IBM 1401 Electronic Data Processing System,<sup>13</sup> and the 1956 IBM RAMAC<sup>14</sup> — the head-disk assembly of the world’s first disk drive. Plenty of whirling tape drives, flashing lights, and interesting mechanical sounds await you!

◇ Dag Spicer  
dspicer (at) computerhistory dot org

[Editor’s note: From its start in Marlborough, Massachusetts, the museum moved to Boston, where, on 21 May 1986, the publication of Knuth’s *Computers & Typesetting* series was celebrated with a “coming out” party. Read about it at [tug.org/TUGboat/tb07-2/tb15complete.pdf](http://tug.org/TUGboat/tb07-2/tb15complete.pdf) starting on page 93.]

<sup>1</sup> [www.computerhistory.org/collections/search/](http://www.computerhistory.org/collections/search/)

<sup>2</sup> [www.computerhistory.org/revolution/calculators/1/1](http://www.computerhistory.org/revolution/calculators/1/1)

<sup>3</sup> [www.computerhistory.org/revolution/calculators/1/42](http://www.computerhistory.org/revolution/calculators/1/42)

<sup>4</sup> [www.computerhistory.org/revolution/birth-of-the-computer/4/82/334](http://www.computerhistory.org/revolution/birth-of-the-computer/4/82/334)

<sup>5</sup> [www.computerhistory.org/revolution/real-time-computing/6/120](http://www.computerhistory.org/revolution/real-time-computing/6/120)

<sup>6</sup> [www.computerhistory.org/revolution/mainframe-computers/7/161](http://www.computerhistory.org/revolution/mainframe-computers/7/161)

<sup>7</sup> [www.computerhistory.org/revolution/supercomputers/10/7](http://www.computerhistory.org/revolution/supercomputers/10/7)

<sup>8</sup> [www.computerhistory.org/revolution/input-output/14/347](http://www.computerhistory.org/revolution/input-output/14/347)

<sup>9</sup> [www.computerhistory.org/revolution/personal-computers/17/312](http://www.computerhistory.org/revolution/personal-computers/17/312)

<sup>10</sup> [www.computerhistory.org/revolution/the-web/20/391](http://www.computerhistory.org/revolution/the-web/20/391)

<sup>11</sup> [www.computerhistory.org/makesoftware/exhibit/](http://www.computerhistory.org/makesoftware/exhibit/)

<sup>12</sup> [www.computerhistory.org/exhibits/pdp-1/](http://www.computerhistory.org/exhibits/pdp-1/)

<sup>13</sup> [www.computerhistory.org/exhibits/ibm1401/](http://www.computerhistory.org/exhibits/ibm1401/)

<sup>14</sup> <http://www.ed-thelen.org/RAMAC/>

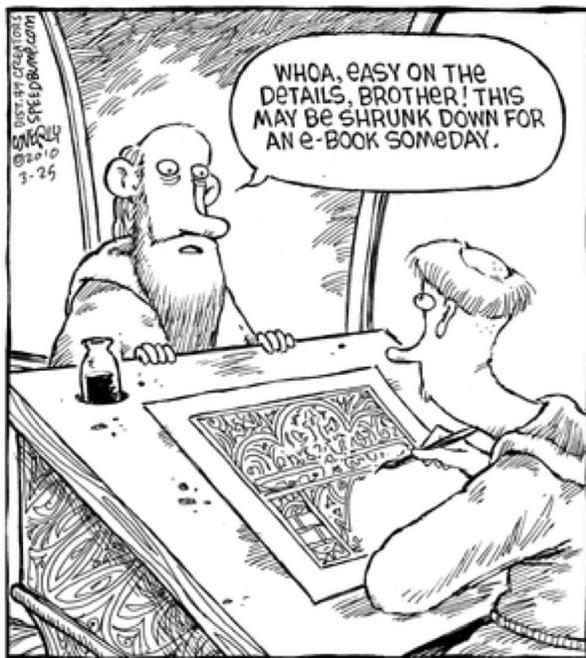
**Type design: Catching up to the past**

Steven Matteson

Editor’s note: This is a lightly edited transcript of the talk given at the TUG 2022 conference. Some of the illustrations are omitted here; for the full set, and the video of the talk, see [tug.org/tug2022](http://tug.org/tug2022).

**Abstract**

The typographer’s goal is to provide the best possible reading experience for the reader. Thirty years of disruptive technologies have made this a greater challenge despite the overwhelming number of type designs available to us. Steve Matteson will give several historical and contemporary examples where fonts have been adapted or designed to meet constantly changing technological demands.



Thanks for the invitation back to speak this year. “Today’s type designer ought to design typefaces for specific needs.” This quote is from Chuck Bigelow, who I met in 1987 when he received the Goudy Award at RIT. It was an epiphany for me as a student. I’d already fallen hopelessly in love with letters — typefaces, typography, lettering. I’d been steeped in letterforms but where did I fit in?

TWO THOUSAND YEARS OF LETTER DESIGN  
 Two Thousand Years of Letter Design

After all, as a student, you study 2,000 years of letterforms and think ‘it’s all been done’. There’s no way I can do better or improve on the achievements of the masters.

Typeface designs have evolved on a parallel course with changes in the way in which technology reproduces them. With each change in technology we, as type designers, have to think about what were the best examples of design using previous technologies. With this understanding we can improve designs for the future.

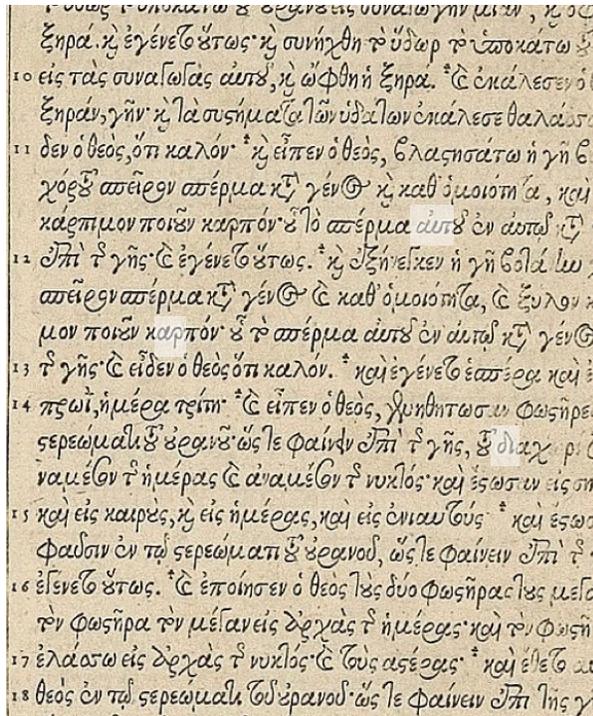
While some of these designs were certainly influenced by fashion, political movements and basic function rather than form, many also saw some underlying change in technology which had an impact on the success of the new approach to design.



When I started working with the latest desktop publishing tools at school and at work, I could see that not all fonts could hold up to the limitations of resolution and imaging technologies. New typefaces could be designed like bitmap fonts — which were used for individual size — or scalable fonts like Lucida, a family of different styles which worked well together even for bad laser printer output.



Gutenberg’s font of type had about 290 sorts, so if a line came out too long or short he had options to adjust the width of the line. Many remark that his columns line up as well as the scribes could do by hand.



Jump forward 100 years and we see another example of ligatures mimicking handwriting — in this case the Greek script. Christophe Plantin’s polyglot bible (ca. 1560) shows here, when viewed in detail, ligatures of common combinations of two and three letterforms, and alternative letter designs.

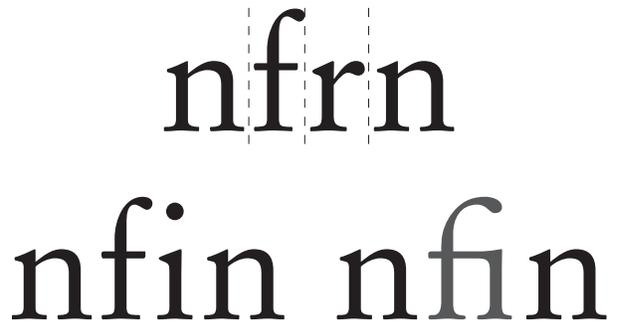


For the Latin alphabet — post Gutenberg — we have a more basic set of ligatures to help with the characters which are problematic — like the Qu combination in the Jenson example but more notably

in the f and long s (which I’ll leave out of this talk because it’s an odd duck we no longer use).

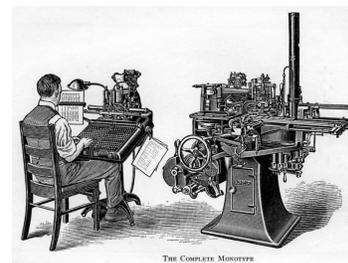
In this slide you can see how a type founder could cast the type so that it would overhang onto the shoulder of the following letter. This allowed for elegant f shapes, but would cause awkward situations like the dot of the ‘eye’ or the ascenders of h, l etc.

The solution was to create a new sort with a specially designed pair of letters.



Linotype machine, 1884

When the Linotype machine was invented the graceful overhang of f had to be tempered for the sake of mechanization. Ligatures could be added in manually, as shown in this slide, but were often just left out.



Monotype machine, 1897

The release of the Monotype keyboard and caster 13 years later reintroduced ligatures to the mechanized work flow. An operator could keyboard ligatures directly in the flow of text. For this and other reasons the Monotype machine was preferred for fine book publishing. Again looking to the past to restore lost aesthetics.

## Gabriola

fi ffij ffk ffk | gg tt | ογ θη σφ α π̄ας

## Zapfino

Th ff ft tt th gg pp ph

OpenType, 1996

Two contemporary examples bring us back to Platin's and Gutenberg's attempts at mimicking handwriting through the use of ligatures (among many other design tricks). John Hudson's tour de force, Gabriola has over 4500 characters to include Latin, Greek and Cyrillic. But it is unique in its extensive collection of ligatures. I think there are 100 f-ligatures alone. The beautifully comprehensive Greek is similar to Platin's with as many as three characters joined as though they were written by hand.

Hermann Zapf's Zapfino has less than half the characters of Gabriola, and only supports Latin characters. Naturally a connected typeface such as this requires a huge number of special combinations to help fool the reader into thinking that a document was written specifically for him or her. Who better to concoct such a scheme than a master calligrapher such as the late professor.



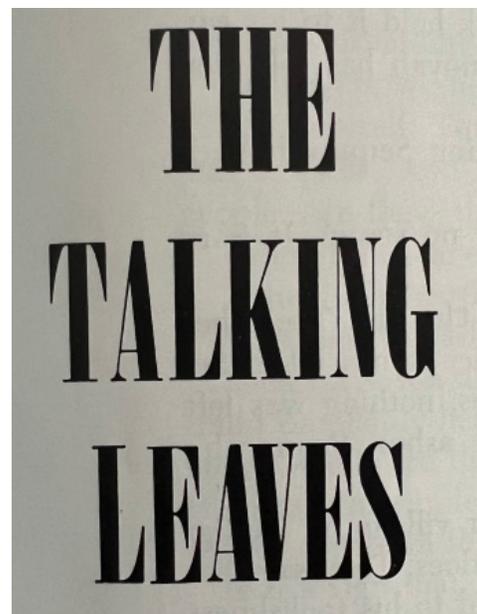
Left: 180pt kerned; right: 16pt unknerned

**Kerning.** As I alluded to with the Jenson example, Kerning was once a physical and time consuming process. To cut the wood or lead shoulder away from two letters so they fit together neatly required a steady hand and a good saw and file. I experienced this firsthand in our once-great letterpress lab at RIT when, just as our professor walked in, there was a loud 'ping' sound as a piece of type ricocheted out of a power saw. A lab assistant was then admonished for attempting to miter a small piece of metal type and had lost his grip but luckily not any fingers.



Courtesy of StarshapedPress.com

This lovely example from Starshaped press, used in an article 'Let's hear it for the kern', shows the amount of material that would have to be removed to fit the AV in this font. It also shows that, while the second line is kerned, it's not much better, given the resulting uneven spacing. The I and L would also need to be spaced apart with spacing material to reflect the openness of the AVA combination.



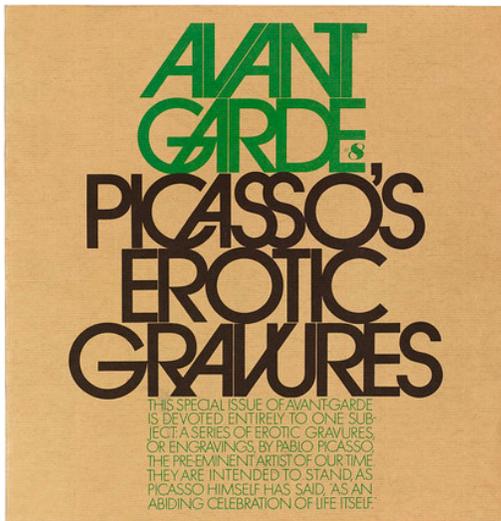
The Talking Le av es

Kerning, as often as not, could make things worse in display typography where not every piece of type is notched. Here the unkered word Talking looks better than the kerned result below: Leaves becomes three fake words: le av es.

In the late '70s and early '80s  
 “Tight but not touching”  
 was the art director’s mantra  
 for headline typography.

*(Some touching may have taken place anyway.)*

Things quickly changed with phototypesetting, where it was comparatively very easy to kern. Just reverse or advance the filmstrip, expose a character and move on. No risk of losing a finger. “Tight not touching” was one descriptor of this style of typography. Others called it sexy spacing. Whatever you called it, this style dominated the scene for easily two decades. I call it kerning and not tracking because the people that were really good at this were not just entering a negative spacing value for whole lines of text - they carefully adjusted each letter combination in each word for the greatest effect.



design by Herb Lubalin

The master of this style was Herb Lubalin. As a lettering artist he knew exactly what he was doing when it came to interlocking and fitting letters. In the case of his Avant Garde typeface he nudged letters so close together he needed a huge set of ligatures of overlapping shapes to get the effect he wanted. Sexy spacing indeed.



By the time desktop publishing was taking off the sexy spacing schtick was overdone, even farcical. Got a long headline? Just keep hitting the minus button. Never mind using a condensed typeface or just a smaller size.

GOOGLE: 4,430,000 HITS

What is kerning?  
 How to kern like a designer  
 Kerning 2020 update  
 Kerning text simply  
 Art of kerning  
 New rules for kerning  
 Mathematical route to kerning  
 10 simple steps to kerning  
 13 tips to avoid bad kerning  
 30 kerning fails  
 Kerning: It's not always a good thing

Out of curiosity I googled kerning and found 4.4 million hits. Mostly it seems by people who never actually designed a typeface in their lives. All these rules and steps written by people with, perhaps, no practical experience at all. Thank you internet. My favorite—and probably the best piece of advice—kerning is not always a good thing. A title for a communication arts article that I haven't actually read but the title gives me some hope.

AVA no kern

VAV over-kern

AVAILABLE just right

MATTESON → MATTESON

As far as my own thoughts and approaches for kerning: I go in with the understanding that it is not a cure-all. You can kern to improve the basic rhythm

Type design: Catching up to the past

in a typeface but it is impossible to solve the balance of every imaginable scenario — even my own name. In fact I vividly remember my first kerning assignment in school where I was given a 72 pt drawer of type and told to typeset my name in all caps. The instructor said to me ‘ooh you’re going to have fun with this’, referring to the big white gap caused by the TT combination.

AVA ÅVÅ ÁVÁ ÄVÄ ÂVÂ  
Avin Åvin Ávin Ävin Âvin  
“A” “Å” “Á” “Ä” “Â”  
Ta Tå Tá Tâ Tâ Tã Tã Tã Tã . .

Today’s big advantage with kerning is the fact that we can consistently space every re-occurrence of a letter shape. Accented letters and punctuation can all be adjusted to fit together the same way a scribe would combine them by hand.

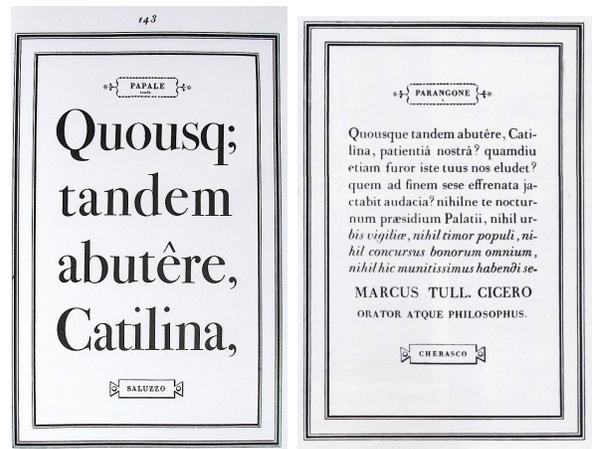


Aldus Manutius’ Virgil and Horace, printed in 1501

Finally I’ll discuss imaging of type and how it’s affected type design. We’ve already had a good look at Jenson and Gutenberg so let’s jump ahead a bit to 1501. When one speaks of beautiful books and beautiful typography, it’s pretty easy to get swept away with Aldus Manutius’ Virgil and Horace. These are the first use of italic types in print. They were cut by Francesco Griffo and based on pre-Carolingian calligraphic hands like the work of Giovanni Tagliente.

Italics were very popular during the Italian Renaissance both for their fashionable look and the fact that their narrow letter width allowed more lines to fit per page.

Printed on smooth vellum, the meticulous details of this gorgeous type hold up beautifully. Thin strokes are not too thin, the rhythm instilled by the italic slant is very regular and even in color.



In 1788 Giambattista Bodoni printed his specimen of typefaces — about 140 individually cut sizes of types. The *Manuale Tipografico*, as it’s called, clearly shows the contemporary fashion of upright stress and extremely fine hairlines. But it also illustrates an important reality that people in today’s digital type world forget. Each size is a slightly different design, cut by hand for maximum style, legibility and proportion.

## Display

Text Size With infinite complacency men went to and fro over this globe about their little affairs, serene in their assurance of their empire over matter. It is possible that the infusoria under the microscope do the same. No one gave a thought to the older worlds of space as sources of human danger, or thought of them only to dismiss the idea of life upon them as impossible or improbable. It is curious to recall some of the mental habits of those departed days. At most terrestrial men fancied there might be other men upon mars, perhaps inferior to themselves and ready to welcome a missionary enterprise. Yet across the gulf of space, minds that are to our minds as ours are

## Display

Text Size With infinite complacency men went to and fro over this globe about their little affairs, serene in their assurance of their empire over matter. It is possible that the infusoria under the microscope do the same. No one gave a thought to the older worlds of space as sources of human danger, or thought of them only to dismiss the idea of life upon them as impossible or improbable. It is curious to recall some of the mental habits of those departed days. At most terrestrial men fancied there might be other men upon mars, perhaps inferior to themselves and ready to welcome a missionary enterprise.

### Scalable type expectations

To illustrate: what we expect today is for the same typeface to work at every size we select. 4 or 400 point — same letter, just a different size. If this were really meant to be, Bodoni’s largest and most contrasting size for display would be difficult to read in text. On the left you can see the condensed forms and somewhat wobbly rhythm compared to a true text cut on the right.

6pt **Bodoni**

12pt **Bodoni**

72pt **Bodoni**

1994 ITC Bodoni

Conversely, if we set the text size as big as the display, we see how heavy the hairlines are in order for them to hold up at small sizes. ITC Bodoni claims to be entirely accurate to the type punches that Bodoni cut for 6, 12 and 72pt. I remain a bit dubious about this though I haven't had the pleasure of going to Parma Italy to see for myself.

**Display**

Text Size With infinite complacency men went to and fro over this globe about their little affairs, serene in their assurance of their empire over matter. It is possible that the infusoria under the microscope do the same. No one gave a thought to the older worlds of space as sources of human danger, or thought of them only to dismiss the idea of life upon them as impossible or improbable. It is curious to recall some of the mental habits of those departed days. At most terrestrial men fancied there might be other men upon mars, perhaps inferior to themselves and ready to welcome a missionary enterprise. Yet across the gulf of space, minds that are to our minds as ours are to those of the beasts that perish, intellects vast and cool and

**Display**

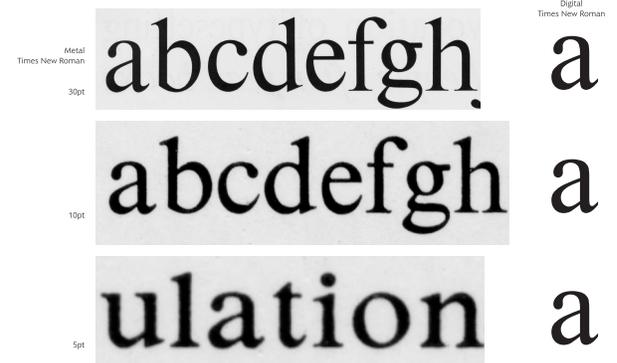
Text Size With infinite complacency men went to and fro over this globe about their little affairs, serene in their assurance of their empire over matter. It is possible that the infusoria under the microscope do the same. No one gave a thought to the older worlds of space as sources of human danger, or thought of them only to dismiss the idea of life upon them as impossible or improbable. It is curious to recall some of the mental habits of those departed days. At most terrestrial men fancied there might be other men upon mars, perhaps inferior to themselves and ready to welcome a missionary enterprise. Yet across the gulf

2019 Walbaum

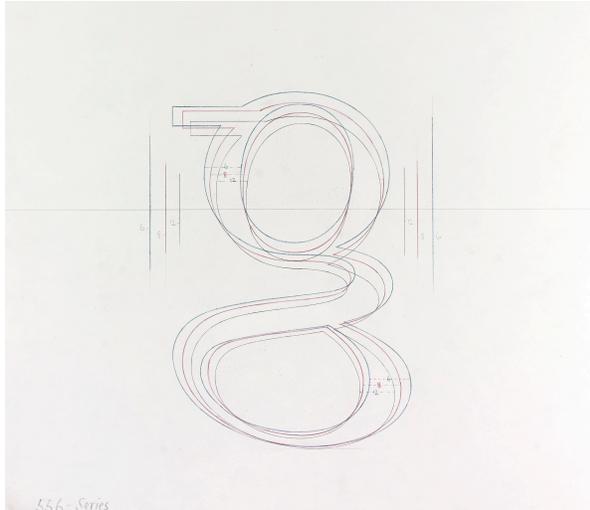
Another quick example of using a display face for text. Walbaum's extreme hairlines are beautiful and elegant in the intended size, but blow out in text. This may remind some of things like the Vogue magazine logo, whose hairlines are so thin they'd never print well for anything smaller than a logo.



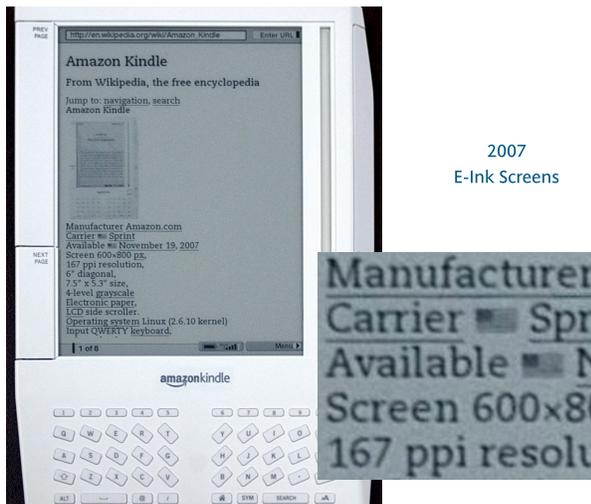
Times New Roman, made by Monotype for casting metal newspaper type, had over 30 different master sizes created. Each unique in slightly different ways. When you look at the catalog you might be struck by how legible Times is at 5pt and how elegant it is at 72. This cannot be matched in digital unless different designs are created for this wide range of uses. The digital version of Times was based on the 14pt metal size drawings.



The idea being that 14 was text-like enough for extended reading but delicate enough for large sizes. You could call it Times New 'Average'. It works reliably in text but struggles for the extreme small use. It looks ok in display but loses a great deal of its finesse.



The Monotype archive has every drawing of every size of typeface. I found this lovely unique illustration of size overlays. This g from a typeface called Spectrum by Jan van Krimpen shows the nuanced changes from 6, 8, 12pt sizes. You can see slight adjustments to weight and the amount of prescribed spacing on each side.



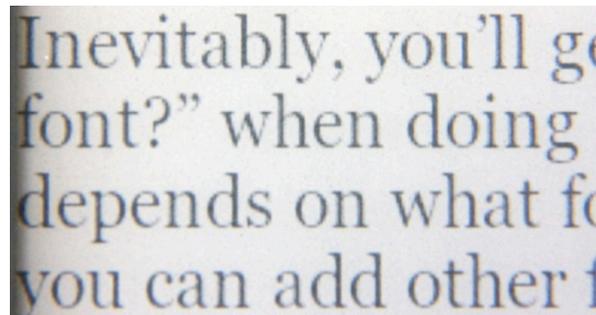
This notion of making designs for specific sizes, or size ranges, is what drew me into the fray of the e-reader boom. Seeing how really well designed fonts that were made specifically for print were falling apart in digital products encouraged me to look at size masters for solutions.

The problem of e-Readers was not just of low resolution; the poor contrast of e-ink screens added to the deformation of letterforms. The magnetic disks that were supposed to flip to their black or white sides in the e-Reader hardware didn't always do so, adding to the grey-cast of the screens. I think

one engineer told me 'white' was actually about 70% grey in the best scenarios.

Amazon had chosen a pretty solid design for the Kindle: Thesis Serif by Lucas de Groot. It had humanist book proportions, was mono-linear with sturdy serifs. Most designs of this ilk are not book proportioned — they're typically very geometric such as Rockwell or Stymie and not great for reading extended text.

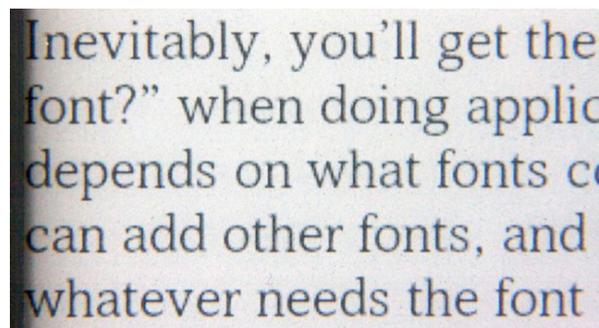
## Inevitably, you'll g



Georgia, by Matthew Carter, renders poorly on an early e-ink screen

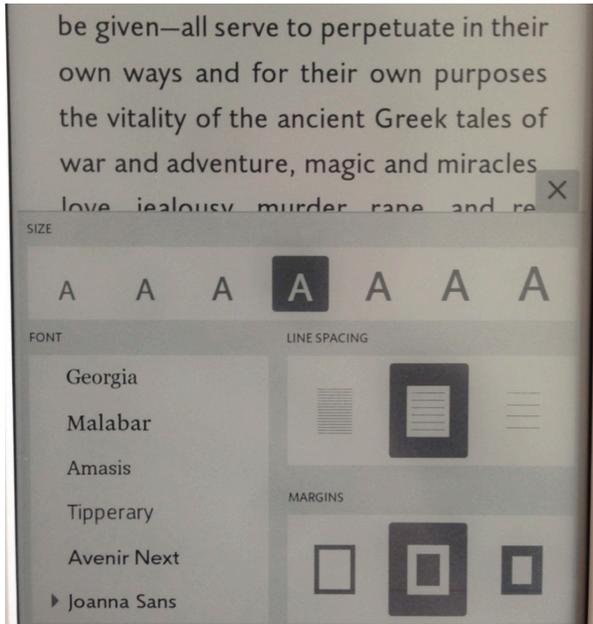
When Barnes and Noble started developing their competing product, the Nook, I immediately suggested what I thought were outstanding low resolution text fonts — Lucida and Georgia. The results of both were quite bad; one could never have predicted how bad without actually seeing the product firsthand. Here you can see Georgia falling apart with the 4 bits of greyscale and low contrast causing many letter details to consistently drop completely out of sight. Unfortunately I don't have a picture of it but Lucida almost looked like a stencil design as hairlines simply looked detached from the stems.

## Inevitably, you'll get the

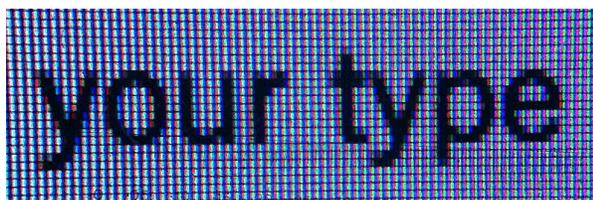


Monotype's Amasis typeface works better but still needs improvements

I decided to try Monotype’s Amasis family as a starting point — shown here with improved contrast but still not quite right. I made several edits to the design creating an “e-paper specific” version which bumped up the visual contrast by strengthening hairlines a bit further and increasing the x-height slightly — just as one would do in making a small-size font master.



Now, with e-paper’s latest generation the resolution is 300dpi and the contrast closer to 90% white. The font menu is far more extensive as a result. The fonts still need more robust qualities than many existing digital book faces but the options are far greater in number than when they started.

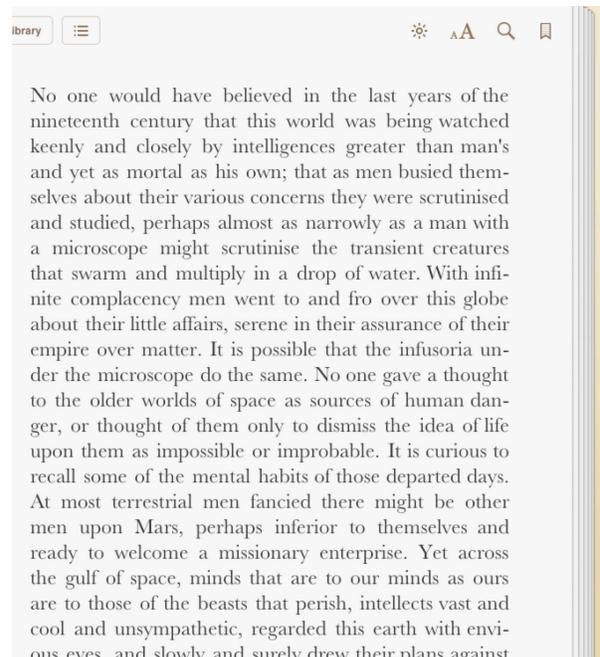


At the same time all of this was happening with e-paper, LCDs were gaining in pixel density. LCD monitors, however, varied a great deal from one to another and to complicate things further every

platform, browser and some apps all rendered type differently.

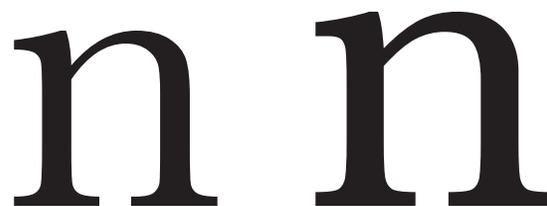
Interesting side note: E-reader or tablet companies would order screens, reject them based on some hardware flaws and they’d get sold to automotive companies for their navigation systems further degrading that sector for reading.

This image of different rasterizations of the same typeface, Verdana, illustrates well the moving targets we faced in designing fonts for screen use. The Segoe typeface I designed for Microsoft’s OS in 2004 was carefully analyzed with the ClearType rasterizer turned on. Now the technology has changed and resolution has improved I’m not sure what’s being used but it still looks ok.



2010 iPad, 132dpi LCD screen

When the iPad came out Apple heralded their book app. They provided a menu of fonts chosen at random rather than by quality for reading on screen. Only Palatino was pleasant to read. They offered Monotype Baskerville but it looked delicate and wan on the iPad screen.



Baskerville MT

Baskerville E-Text

Type design: Catching up to the past

ere but poor entertainment, and the f one of the worst of men, among th our Host made one, who, had he udent, would have outdone his his I think is the most perplexed l. From hence, Saturday, Dec. 23, a ly day, after an Intolerable night's forward only observing in our way rated on a Navigable river w<sup>th</sup> in-nd people more refined than in / towns wee had passed, tho' vicious and Tavern being next neighbours. difficult River wee come to Fairfield id were much refreshed as well with <sup>th</sup> Latter I employed in enquiring n and manners of the people, &c. c town, and filled as they

could get nothing there but poor en Impertinant Bable of one of the w many others of which our Host ma bin one degree Imudenter, woul Grandfather. And this I think is night I have yet had. From hence, very cold and windy day, after a Lodging, wee hasted forward only the Town to be situated on a Na diferent Buildings and people m some of the Country towns wee had enough, the Church and Tavern be Having Ridd thro a difficult River v where wee Baited and were much r the Good things w<sup>th</sup> Latter I em; concerning the Town and manne: This is a considerable town, and fil

So I once again turned to size masters for the solution. The eText version of Baskerville is based on the 6pt drawings for metal. The serifs and hairlines are quite sturdy and the stems are darker for greater contrast. The height was larger to look 'bigger on the body', becoming more legible in comparative text.

Alas, as things go with Apple, the advice to switch fonts fell on deaf ears. Fonts aren't as sexy as apps. Luckily both Amazon and B&N picked up the design. Jeff Bezos, in an unprecedented nod to fonts, remarked with fascination that the font he was showing an audience was from the 18th century!

**T**he day Thorvald's mother gave changed. Creidhe was weaving, h flying, a fine web of blue and cr perfect pattern, testimony to the s her. So industrious was she, and s been forgotten. The bestowal of s was surely best suited to a mom Margaret spoke to her son quietly hearth. Creidhe could see them i weaving chamber. They did not ar in this most orderly of household door slam open, and she saw Thor

switchboard. Within fifteen seconds every light was on.  
 "Our cameramen have taken film that has either been confiscated or deliberately exposed. Our reporters' stories have disappeared. Yet we do have film, ladies and gentlemen, and we have correspondents right here in the studio—not professional reporters, but eyewitnesses to what may be the greatest disaster this country has ever faced ... and I do not use those words lightly. We are going to run some of this film for you now. All of it was taken clandestinely, and some of it is of poor quality. Yet we here, who have just liberated our own television station, think you may see enough. More, indeed, than you might have wished."  
 He looked up, took a handkerchief from his sport-coat pocket, and blew his nose. Those with good color TVs could see that he looked flushed and feverish.

For all the work put into getting the fonts to look good we still have to put up with stultifying, dull, typography of e-readers. Horrible hyphenation/justification, feeble attempts at chapter navigation and the occasional flying drop capital.

If You Answer the Door for Strangers...

You see it all the time on television. There's a knock at the front door. And, on the other side, someone is waiting to tell you the news that changes everything. On television, it's usually a police chaplain or a firefighter, maybe a uniformed officer from the armed forces. But when I open the door—when I learn that everything is about to change for me—the messenger isn't a cop or a federal investigator in starched pants. It's a twelve-year-old girl, in a soccer uniform. Shin guards and all.

"Mrs. Michaels?" she says.  
 I hesitate before answering—the way I often do when someone asks me if that is who I am. I am and I'm not. I haven't changed my name. I was Hannah Hall for the thirty-eight years before I met Owen, and I didn't see a reason to become someone else after. But Owen and I have been married for a little over a year. And, in that time, I've learned not to correct people either way. Because what they really want to know is whether I'm Owen's wife.

It's certainly what the twelve-year-old wants to know, which leads me to explain how I can be so certain that she is twelve, having spent most of my life seeing people in two broad categories: child and adult. This change is a result of the last year and a half, a result of my husband's daughter, Bailey, being the stunningly disinviting age of sixteen. It's a result of my mistake, upon first meeting

Regular Answer  
 Fake Bold Answer  
 Bold Answer

In the Libby library e-book application they have very few good type choices and pull out the oldest trick in the book saying they have a bold weight of a font but actually let the application alter the regular in a pseudo bold. Literally the rasterizer smears the bitmap in the x direction to appear a little darker. In this case the font gets tracked, too, or spaced out in the process. This really gets messy in the word firefighter (fourth line) where the fi ligatures look tight while the rest of the word expands.



Microsoft: BEYOND CALIBRI

Bierstadt  
 Grandview  
 Seaford  
 Skeena  
 Tenorite



For all the problems that crop up it's nice to know that companies like Microsoft still appreciate the value of good fonts. Last year they launched five new designs to augment their Office applications. And they're not just new fonts for the sake of new fonts. Their purpose is to expand their font menu into genres they didn't support well before.

Particularly appreciated is their attention to purpose. Size-specific designs are included in all of these families, including my Bierstadt. Not only is there a display version of each (most useful for PowerPoint headlines) but a focus on things like width: how much information one can fit in a spreadsheet cell vs. how legible (or illegible) the design becomes.

**48pt PowerPoint Bold Display**

- 24pt Bullets using semibold
- Large sizes are more compact and impactful
- Smaller sizes are spaced for greater legibility

**Effective PowerPoint Slides**

- Legible sub-headings
- Simple but effective hierarchy

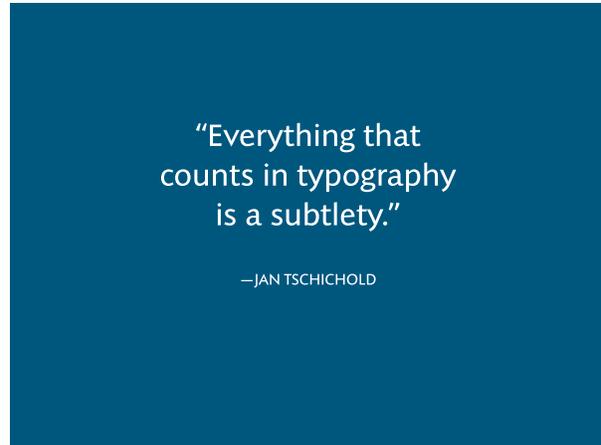
**48pt PowerPoint Bold Display**

- 24pt Bullets using semibold
- Large sizes are more compact and impactful
- Smaller sizes are spaced for greater legibility

**Effective PowerPoint Slides**

- Legible sub-headings
- Simple but effective hierarchy

Examples of PowerPoint with just one size font fits all (above), vs. separate display and text versions (below) to optimize the aesthetics of a PP presentation.



I'll leave you with this quote from Jan Tschichold. It is profound in many ways since type design is an aggregate of many many detailed decisions which all affect the performance of the type—be it legibility or fashion.

Though we may be armed with the most advanced fonts and authoring tools ever known, it still feels like a constant battle to raise the bar of quality, beauty and function.

With the technology landscape continually changing there will always be problems for type designers and typographers to overcome—I've barely scratched the surface here and I've not even mentioned writing systems other than Latin.

The next generation of type designers can look at Mr. Bigelow's words in 1987 and rest assured that their work is far from being finished.

◇ Steven Matteson  
 mattesontypo (at) gmail dot com  
 mattesontypographics.com

## The Tectonic Project: Envisioning a 21st-century $\TeX$ experience

Peter K. G. Williams

### Abstract

Tectonic is a software project built around an alternative  $\TeX$  engine forked from  $X_{\text{Y}}\TeX$ . It has been created to explore the answers to two questions. The first question relates to documents: in an era of 21st-century technologies—where interactive displays, computation, and internet connectivity are generally cheap and ubiquitous—what new forms of technical document have become possible? The second question relates to tools: how can we use those same technologies to better empower people to create excellent technical documents?

The premise of the Tectonic project is that while  $\TeX$  may be venerable, it is still an ideal system for creating “21st-century” technical documents—but that a project with an independent identity and infrastructure can make progress in ways that can’t happen in mainline  $\TeX$ . Tectonic is compiled using standard Rust tools, installs as a single executable file, and downloads support files from a prebuilt  $\TeX$  Live distribution on demand.

In the past year, long-threatened work on native HTML output has finally started landing, including a possibly novel Unicode math rendering scheme inspired by font subsetting. Current efforts are fleshing out this HTML support using  $X_{\text{Y}}\TeX$ : *The Program* as a test case, with an eye towards substantially improving the documentation of Tectonic itself. While Tectonic positions itself as “outside of” traditional  $\TeX$  in a certain sense, the project could not exist without the efforts of the entire  $\TeX$  community, to whom the author and the project are gratefully indebted.

### 1 Introduction

This article will motivate the Tectonic project (§2), discuss some of its distinctive characteristics (§3), delve into how it is implementing HTML output (§4), and briefly discuss the outlook for its future (§5).

### 2 Motivation

I (PKGW) will motivate the Tectonic project with a somewhat stylized history of my journey through the  $\TeX$  ecosystem. My background is in scientific research (astronomy), and to the best of my recollection, my first use of  $\TeX$  was for typesetting problem sets in college. I still remember the satisfaction of creating a beautifully typeset equation, and understanding that there was no other tool in the world

[Next](#) | [Up](#) | [Previous](#) | [Contents](#) | [Index](#)

[Next: Quality of Printed Images](#) [Up: Figures and Image Conversion](#) [Previous: An Embedded Image Example](#) | [Contents](#) | [Index](#)

### Image Sharing and Recycling

 [change](#) [be](#)<sup>96.1</sup>

It is not hard too see how reasonably sized papers, especially scientific articles, can require the use of many hundreds of external images. For this reason, image sharing and recycling is of critical importance. In this context, “sharing” refers to the use of one image in more

**Figure 1:** A screenshot of typical  $\LaTeX$ 2HTML output. From [www.sci.utah.edu/~macleod/latex/latex2html/Enode8.html](http://www.sci.utah.edu/~macleod/latex/latex2html/Enode8.html), chosen arbitrarily.

that could typeset math nearly as well—at least, none that could be freely used by a college student.

During my college career (2002–2006, if you must ask), it was clear that the Internet and World Wide Web were on their way to transforming society. But for the most part, design on the web was notoriously poor.  $\LaTeX$  could be converted to HTML and rendered, but the results resembled Figure 1: legible, mostly, but absolutely inferior to what could be accomplished in PDF. And while HTML documents had hypertext capabilities, they were generally *static* documents: words and figures arranged on a page in a facsimile of the “real thing”: ink on paper.

For me, there were two major “watershed moments” demonstrating that web documents weren’t always going to be inferior. First, the release of Google Maps (February, 2005; [maps.google.com](http://maps.google.com)) showed that websites could be *applications*, not just static documents. In my mind, this opened up exciting possibilities for new forms of scientific and technical communication: not just hypertextual facsimiles of paper, but *interactive* documents.<sup>1</sup> More broadly, I’ll define *21st-century documents* as those that leverage the technologies that have been unrolling since then: documents designed for a world where interactive digital displays, computation, and internet connectivity are often cheap and ubiquitous. (I don’t love this terminology—smacks of naïve futurism—but don’t have anything better.) In principle, 21st-century documents can target any of a variety of technology platforms, but in my opinion, the web is the only one that matters. Web content can be experienced nearly anywhere, from smartphones to billboards, and private industry is spending *billions* of dollars every year to enhance its power and reach. Nothing else comes close.

<sup>1</sup> See the slides to my talk, which are in HTML, for an example of an embedded interactive plot ([tug.org/tug2022/assets/html/Peter\\_K\\_G\\_Williams-TUG2022-slides/](http://tug.org/tug2022/assets/html/Peter_K_G_Williams-TUG2022-slides/)). Because *TUGboat* is delivered in PDF format, I can’t reproduce the plot here.

But in 2005, the state of web typography was still pretty poor, and so PDF was still easily the best choice for scientific papers and other kinds of *technical documents*. While I won't attempt to define this category precisely, common characteristics of technical documents include substantial length; use of mathematics, figures, or tables; complex structure; dense internal or external referencing; and more recently, integration with source code and computation. While every kind of document deserves excellent typography, I will assert that technical documents probably suffer *more* from bad typography than non-technical ones. The second watershed moment for me was therefore the release of the `pdf.js` library for displaying PDFs in the web browser (July, 2011; [mozilla.github.io/pdf.js/](https://mozilla.github.io/pdf.js/)). What better way to demonstrate that you can do high-precision typography on the web than by demonstrating that you can render arbitrary PDF files?

After seeing `pdf.js`, I was convinced that all of the pieces were in place to start trying to bring the typographic quality of  $\text{\TeX}$  to the world of web-native, 21st-century documents. Even attempting this would surely require new software to be created — but alongside my scientific training, I've been involved in open-source software development since even before I started college, and I'm more than happy to tackle such problems myself.

My first push, undertaken around 2014, was an attempt to do a clean-room implementation of the core  $\text{\TeX}$  engine in JavaScript, using *The  $\text{\TeX}$ book* as a reference. It went about as well as you might expect. I made decent progress, but quickly came to understand that the  $\text{\TeX}$  engine itself is just the tip of the proverbial iceberg of code needed to compile actual modern  $\text{\LaTeX}$  documents, and that *The  $\text{\TeX}$ book* is only a partial — in fact, sometimes misleading — guide to how modern  $\text{\TeX}$  engines operate, with no discussion of  $\varepsilon\text{-}\text{\TeX}$ , Unicode, OpenType fonts, and more. I concluded that in order to compile “real”  $\text{\LaTeX}$  documents for the web, one would need to build on “real”  $\text{\LaTeX}$ .

So I started to look into hacking  $\text{\TeX}$ . More specifically, I looked into modifying the  $\text{X}\_{\text{Y}}\text{\TeX}$  engine, since it includes support for Unicode and OpenType fonts, which struck me as essential for creating truly web-native documents. As a person with a long history in open-source projects, the experience was frankly frustrating and discouraging. Even the traditional first step for getting to understand a codebase — checking out the source code from version control — felt like an ordeal, primarily due to a lack of clarity about which repository to use, the huge size and deeply nested structure of the  $\text{\TeX}$  Live repository,

and the use of Subversion. Modern software development conveniences, above all the availability of some kind of GitHub-like “pull request” mechanism and continuous integration (CI), were missing.

While there's no shortage of fads in the world of software development, there have been some real advances, and as a developer I find them extremely important. For me, undertaking a software project without them is like trying to write a research paper in Microsoft Word rather than  $\text{\LaTeX}$ . I can do it if I have to, but I won't enjoy it, and the inferior tools close off entire ways of working that I don't want to give up. I didn't feel that I could work with the  $\text{\TeX}$  code in the way that I wanted to, if I was forced to use the existing infrastructure.

But if you take a software project and rebuild its development infrastructure, it is unlikely to be feasible to merge your changes back into the original source. Such changes result in a long-lived *fork*, not a temporary *branch*. Forking a project is a weighty decision, not to be taken lightly. But as I thought about the experiments I wanted to try, I came to believe that forking was an appropriate path. Besides allowing me to explore newer development tools, it would allow me to explore a new “persona” for the project — a distinct brand identity. This may sound like business jargon, because it is, but it captures the right concept. I wanted the ability to try *all sorts* of things that you couldn't do with traditional  $\text{\TeX}$ : tidy up the output, change default behaviors, drop compatibility with various ancient packages. It wouldn't be right to describe such a system as a regular  $\text{\TeX}$  system. New branding gives an opportunity to reset user expectations all at once, without having to explain the details of individual technical changes.

### 3 The Tectonic Project

The narrative of the previous section has suggested an interrelated set of gaps in the  $\text{\TeX}$  ecosystem:

- support for creating modern HTML output with a full-featured  $\text{\TeX}$  engine;
- a modernized developer experience;
- a modernized user experience; and
- a project with a distinct brand identity to serve as a platform for experimentation.

Launched in 2016, the Tectonic project aims to fill these gaps. Key elements of its design are as follows.

#### 3.1 Form factor

Tectonic is delivered as a single executable, named `tectonic`, that bundles the capabilities of  $\text{X}\_{\text{Y}}\text{\TeX}$ , `bibtex`, `xdvipdmtx`, and supporting machinery for driving

these *engines*. The executable is designed to be as self-contained as possible, with minimal dependencies on system libraries, environment variables, user configuration files, or external tools. In particular, dependencies on Ghostscript have been removed for security, eliminating PostScript capabilities.

### 3.2 Engine implementation

The engines, most notably  $X_{\text{T}}\text{E}_{\text{X}}$ , are implemented with C/C++ code obtained from the standard WEB2C pipeline implemented by  $\text{T}_{\text{E}}\text{X}$  Live. The C/C++ files extracted from the pipeline have been extensively reformatted and refactored to make them more human-readable and, for instance, reintroduce symbolic constants that do not survive the WEB2C workflow. A few refactorings have been conducted automatically with *coccinelle* [2]. Due to these customizations, however, engine updates from  $\text{T}_{\text{E}}\text{X}$  Live cannot be automatically incorporated into the Tectonic codebase. This is the price of forking. To aid the process of synchronizing Tectonic with  $\text{T}_{\text{E}}\text{X}$  Live, a framework called *tectonic-staging* (code repository at [github.com/tectonic-typesetting/tectonic-staging/](https://github.com/tectonic-typesetting/tectonic-staging/)) contains a pipeline that can automatically generate a readable set of C/C++ “reference sources” from the  $\text{T}_{\text{E}}\text{X}$  Live repository. When updates from a new  $\text{T}_{\text{E}}\text{X}$  Live release are to be incorporated into Tectonic, the pipeline is run and changes to the reference sources are manually imported into Tectonic’s codebase. This system heavily leverages the change-tracking features of the *git* version control system.

### 3.3 Use of Rust

Excepting the engines, Tectonic is implemented in the Rust language. Rust is a systems-level language focusing on performance, reliability, and productivity. Rust’s packaging and compilation model is an excellent fit for a project like Tectonic: while Rust offers a sophisticated package ecosystem that makes it easy to import support for anything from the HTTPS protocol to image loading, it compiles by default into self-contained executables that lack external dependencies. Rust also has excellent support for cross-platform work and bridging with C and C++ code. Rust’s packaging tool, *cargo*, allows codebases to be organized into “crates” with well-defined interfaces, and has a “feature” system for the management of build options. The main Tectonic codebase currently consists of 22 crates.

More broadly, the Rust language has a similar spirit to  $\text{T}_{\text{E}}\text{X}$ . Both are regarded as best-in-class tools that can be demanding, but rewarding as well. Both have a reputation for being complicated and hard to learn. Despite this reputation, Rust has

achieved a tremendous level of success in a relatively short period of time, being named the “most loved language” by [stackexchange.com](https://stackexchange.com) for seven consecutive years as of this writing. Rust supporters generally attribute this success to several factors. First, Rust is technically excellent: it actually delivers on its promises in a rigorous way, and third-party Rust packages are often well-designed, performant, and reliable. Second, Rust has excellent tooling, with high-quality built-in support for package management (*cargo*), documentation, testing, and more. Third, the Rust user community explicitly values being welcoming and inclusive. Many aspects of the Rust design aim to support new users, most famously the Rust compiler’s error messages, which generally offer impressively clear diagnoses of problems and useful advice for fixing them. Spanning these factors is a theme of *experience-centered design*: elements of the Rust ecosystem are designed primarily around a vision of what it will be like for people to use them, with technical goals flowing from that vision. Tectonic explicitly aims to emulate these characteristics of the Rust ecosystem and community.

### 3.4 Bundles

Tectonic can be delivered as a single executable because it can download files from a backing  $\text{T}_{\text{E}}\text{X}$  distribution on the fly, during document compilation. This functionality is implemented by virtualizing the I/O subsystem underlying the engines so that it can search not just the local filesystem, but also remote “bundles”, for files. Files from bundles are cached locally, and the implementation is designed such that the network is needed only if a new file must be fetched. Bundles are created using a reproducible, automated pipeline based on the  $\text{T}_{\text{E}}\text{X}$  Live installation process ([github.com/tectonic-typesetting/tectonic-texlive-bundles/](https://github.com/tectonic-typesetting/tectonic-texlive-bundles/)). The bundle file as served over the network is essentially a large Unix *tar* file with an associated index, which the *tectonic* program downloads in pieces using HTTPS byte-range requests.

The bundle scheme is also the backbone of Tectonic’s approach to reproducible document builds. Bundle files are intended to be immutable, and it is possible to associate a given Tectonic document (see below) with a specific bundle, identified by its url or a SHA256 cryptographic digest based on its contents. It is thus possible to specify the exact  $\text{T}_{\text{E}}\text{X}$  distribution that a document should be built against. There is an associated loss of flexibility: to update or extend a package contained in the bundle, you must generate your own bundle or install the package files locally, currently on a per-document basis.

### 3.5 Document model

Tectonic offers a “document model” for defining compilations. Its design is heavily indebted to that of Rust’s `cargo` tool. Tectonic documents are directory structures indicated by the existence of a file named `Tectonic.toml` at the root. This file, in the TOML structured-data format (`toml.io`), declares basic metadata about a document and how it should be built. By default, the top-level document source code is contained in a subdirectory named `src` in files named `_preamble.tex`, `index.tex`, and `_postamble.tex`, that are processed in that order.

A document can be built by running the command `tectonic -X build` anywhere in its source tree. This will create one or more versions of the document in a `build` directory below the `Tectonic.toml` file. The `-X` flag marks the use of Tectonic’s “version 2” command-line interface, which uses a “sub-command” or “multi-tool” paradigm like `git` or `svn`. For compatibility, the default mode of operation is still “version 1”: `tectonic myfile.tex` will compile the specified input file without invoking the document model. This form of one-shot compilation is accessible in the version 2 interface with `tectonic -X compile myfile.tex`. The version 1 interface will eventually be deprecated and the `-X` flag will become optional.

The main purpose of the Tectonic document model is to make document builds automatable, reproducible, and analyzable by rendering document-specific choices as configuration, rather than (e.g.) a string of command-line options. For instance, the `Tectonic.toml` file can record what  $\text{\TeX}$  format file a build requires, or whether it needs shell-escape functionality. (A runtime flag can override this setting when the document to be built is not from a trusted source.) As alluded to above, the specification can define multiple outputs for a single document, such as PDFs in both US Letter and A4 sizes.

While the document model has not yet been developed thoroughly in Tectonic, it is expected to provide a platform for additional utilities in the future. For instance, a future `tectonic -X format` command might automatically reformat a document’s sources into a consistent style, or `tectonic -X doc` might generate meta-documentation about available control sequences customized to a particular document’s selection of packages.

## 4 HTML output

Although high-quality HTML output has been a goal of the Tectonic project from its inception, little work has happened on this front — until this year. Interesting progress has begun to occur.

The overall approach taken while implementing HTML output for Tectonic has been to focus on achieving high-quality results with documents that specifically target that output format. While the eventual goal is to be able to produce good HTML output from arbitrary input documents, that is a larger problem that is being avoided for the time being. Current efforts also prioritize visual appearance over proper semantic tagging, and focus on the English language.

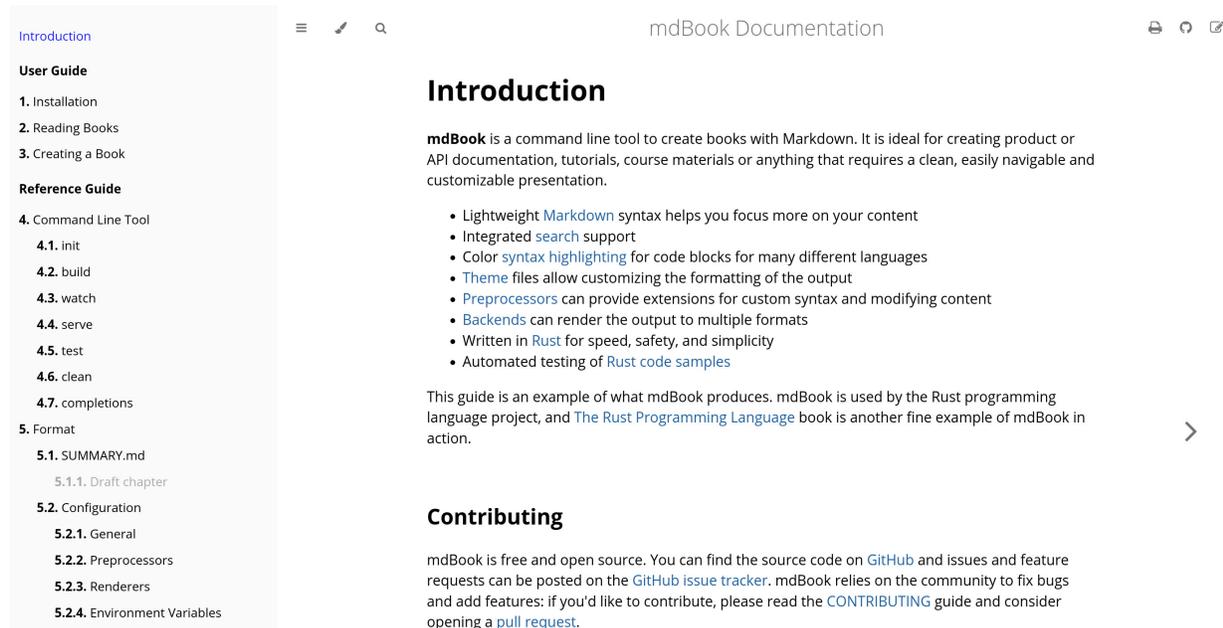
As a broad approach, when HTML output is called for, a special flag in the  $\text{\XeTeX}$  engine is activated that alters various aspects of its behavior. Linebreaking of paragraphs is disabled to avoid dealing with hyphenation, and `\specials` are inserted to indicate engine-suggested locations for insertions of HTML tags such as `<p>`. The resulting output file is essentially in the  $\text{\XeTeX}$  XDV format, but it is relabeled as a new “SPX” format to avoid confusion. (SPX stands for “semantically-paginated XDV”, but it is a misnomer because semantic pagination turned out to be technically infeasible.)

A new processing step written in Rust, `spx2html`, uses the Tera templating framework (`tera.netlify.app`) to convert the single SPX file into one or more HTML outputs, and creates or copies associated files such as CSS stylesheets, JavaScript user interface code, and font files. The `spx2html` stage is designed under the assumption that the input document uses OpenType fonts everywhere, including mathematics, via the `unicode-math` package. This dramatically reduces the problem space by allowing the code to only work with fonts that can be rendered directly by the browser.

### 4.1 Precise typography in canvases

Initial HTML work focused on demonstrating the precise character sizing and positioning needed to render constructs such as “ $\text{\TeX}$ ”. In Tectonic, the bulk of document text is emitted directly into the HTML, but areas needing careful typographic layout are handled specially as *canvases*. Layout in the canvas mode can either be activated automatically by the engine (for instance, in math mode) or manually by the author (with `\specials`).

Because CSS commands can be used to move individual HTML elements arbitrarily, the actual positioning is not difficult, although some care needs to be taken to achieve proper alignment relative to the text baseline for inline expressions. More challenging is the fact that the SPX file specifies how *glyphs* in a font should be placed, while the HTML output must be Unicode text — and these are distinct concepts. In many cases, there is a direct mapping



**Figure 2:** The standard mdBook layout, discussed in subsection 4.2. From [rust-lang.github.io/mdBook/](https://rust-lang.github.io/mdBook/).

between the two, encoded in a font’s Unicode CMAP table; but not always. For instance, a display math environment might call for a large version of an integral sign that cannot be “reached” by emitting a U+222B INTEGRAL character, which maps to a smaller version of the glyph.

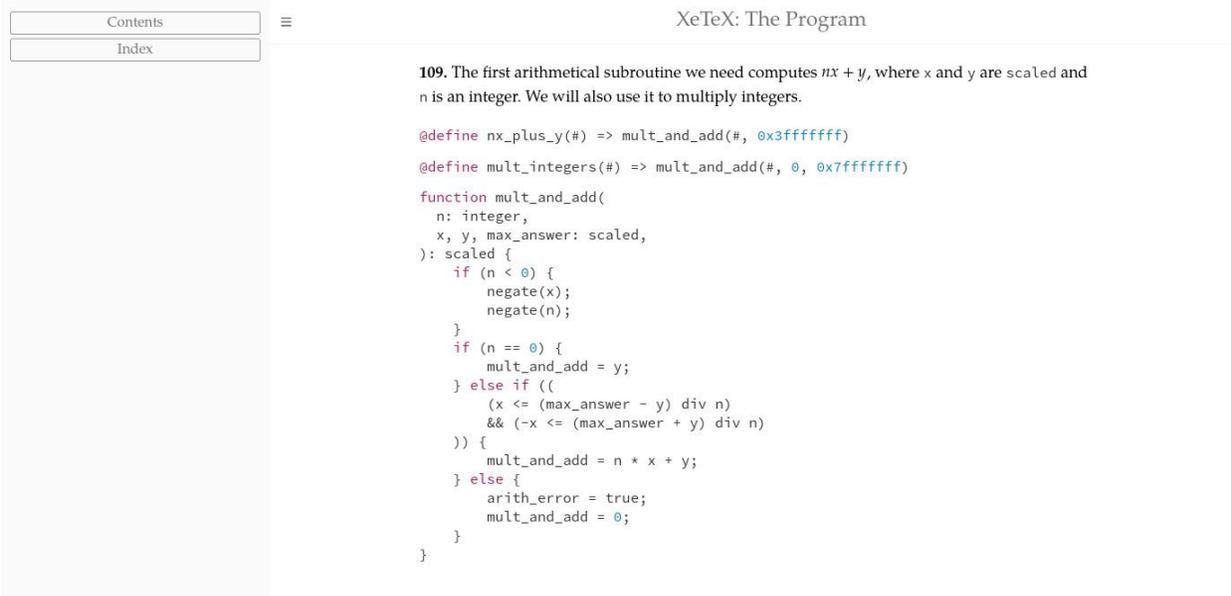
When Tectonic encounters this problem, it addresses it by creating an additional version of the relevant font file with a customized CMAP table. This *variant glyph* approach is a slightly generalized form of font subsetting, although Tectonic’s implementation is much more naïve than a “real” font subsetter. In the example above, the new font’s CMAP table might replace the mapping of U+222B INTEGRAL from the default integral glyph to the large version needed by the display math in question. The HTML for that canvas will then include a `<span>` tag styled to load that font, sized and positioned appropriately, containing a single U+222B INTEGRAL character.

In order to determine whether a new variant-glyph font must be created, Tectonic must parse and invert the CMAP tables of the fonts used by the document that it is processing. This process is potentially fragile, since in full generality it essentially requires inverting character “shaping” algorithms. Note, however, that it only needs to occur for characters that occur within canvases. For the main text of a document, in almost all cases Tectonic can emit Unicode output directly from the ActualText information emitted by the  $X_{\text{T}}\text{T}_{\text{E}}\text{X}$  engine.

## 4.2 The chrome: HTML, CSS, JavaScript

Tectonic’s approach aims to minimize the amount of web design occurring at the  $\text{T}_{\text{E}}\text{X}$  level. Instead, HTML content derived from the  $\text{T}_{\text{E}}\text{X}$  input is inserted into predesigned templates. It is important to emphasize that in modern web design, such templates inevitably consist of interdependent pieces of HTML, CSS, and JavaScript code. These pieces combine to form the *chrome* of the resulting web document. Chrome encompasses everything from the high-level page layout to interactive functionality such as search, hideable sidebars, and non-linear navigation. High-quality default chrome is an essential component of the web document production pipeline.

Current efforts focus on a clean design emulating that of the tool mdBook ([rust-lang.github.io/mdBook/](https://rust-lang.github.io/mdBook/)), a Markdown-based, web-native documentation system. A screenshot of the default mdBook layout is shown in Figure 2. On a large screen, the default view is divided into a main content area and a sidebar. The main body text is centered within the main content area, with a maximum width to prevent line lengths from becoming excessive. An unobtrusive title bar sticks to the top of the page, but auto-hides while the reader scrolls through the main content. On mobile displays, the sidebar remains hidden by default, and can be opened using the “hamburger menu” of the title bar.



**Figure 3:** A snapshot of the in-development `tt-weave` presentation of `XeTeX: The Program`, with obvious debts to `mdBook` (Figure 2).

After a great deal of exploration, I believe that this layout may well be *the* optimal design for presenting general-purpose technical documents on the web. On wide screens, the positioning of the main body text becomes awkward if it is not nearly centered in the browser window. If the maximum width of the text is not limited, lines become too long, as seen on Wikipedia. On mobile, there is little enough room that there should be virtually nothing else on screen besides the main text while reading—a constraint that is accommodated well by the combination of the sticky, auto-hiding title bar and toggle-able sidebar.

Finally, many chrome designs attempt to cram information into multiple sidebars, headers, and footers. These clutter the page and are difficult to use on mobile. A better alternative is to provide these sorts of extras as “modals”, overlays that can be quickly brought up and dismissed using icons in the title bar (on all platforms) or keystrokes (on desktops). This is an example of the way in which chrome consists of more than just page layout: 21st-century documents can have full-blown user interfaces.

### 4.3 `tt-weave`

Current work on Tectonic’s HTML output is focusing on a very specific test case: `XeTeX: The Program`, the variant of `TeX: The Program` [1] produced from `XeTeX`’s patched `WEB` code. It is close to ideal because it is long, highly structured, densely cross-referenced, used frequently by the author, and available as `TeX` source.

The `TeX` source generated by the traditional `weave` program is highly tuned for print output. Although I could potentially have worked to create HTML output from the `weave`-generated `TeX` code, I had an additional goal. I regret to say that I have always found the code listings generated by `weave` extremely difficult to read, even though I know that a great deal of care has gone into their design. I wanted to see if I could create a `weave`-like tool that could reformat the `XeTeX WEB` code into the sort of monospaced, syntax-colored format that I’m more familiar reading.

The result of that work is a Rust tool called `tt-weave` ([github.com/pkgw/tt-weave](https://github.com/pkgw/tt-weave)). It serves essentially the same function as `weave`, but parses the Pascal portions of `WEB` code with a high level of semantic awareness and emits them as blocks of specialized `TeX` code in an indented, monospaced format with embedded commands controlling syntax colorization and interlinking. The syntax of the emitted code is rewritten to superficially resemble C and Rust. For instance, logical “and” is represented using `&&` rather than `^`. No semantic transforms are attempted, however. Indexing information is emitted into JSON data files that can be used by the web chrome. The `tt-weave` program is not intended to be a general-purpose `WEB` processor, and contains numerous hacks specific to the patched `xetex.web` input file. A snapshot of its output is shown in Figure 3. The design shown here is updated relative to the

version included with the HTML slides associated with this presentation.

As of this writing, the corresponding chrome is under development. The entire book text can be rendered into a single HTML file (~10 MB) that is actually comfortable to use in the browser, and the online slides associated with this article include a snapshot of this form of output. Such a large page is an impractical delivery mechanism for general use, however, and work is underway to subdivide the output for dynamic loading. This will also help with integrating Tectonic’s output into industry-standard web development frameworks (e.g., `npm`, `webpack`), which would significantly boost the productivity of development by making it convenient to adopt technologies such as SASS (`sass-lang.com`) or TypeScript (`typescriptlang.org`).

## 5 Outlook

The Tectonic project has been successful thus far, gathering ~2,800 “stars” on GitHub and registering 47 distinct project contributors as of the time of this writing. While much work remains to be done to make the HTML output framework generally usable, the variant-glyph technique successfully addresses the most technically demanding problem in the current system.

The documentation of the Tectonic project — somewhat ironically, given its subject matter and aspirations — is lacking. Once the `tt-weave` effort has demonstrated good success with the existing  $X_{\mathcal{T}}\text{TEX}$ : *The Program* book, the intention is to start creating new documentation to remedy this situation.

Thus far, the prime person driving work on Tectonic has been the author of this article. Since the project’s inception, however, the hope has been to make it a welcoming place for new contributors, and as the project matures, that is more important than ever. There are *numerous* areas — non-Latin scripts, accessibility, non- $\text{\LaTeX}$  workflows,  $\text{\TeX}$  internals — where more expertise from around the  $\text{\TeX}$  world would be hugely beneficial. People interested in engaging with the Tectonic community should visit the Tectonic discussion forum attached to its GitHub repository at `github.com/tectonic-typesetting/tectonic/discussions`.

Of course, Tectonic only exists because it is building on the work of the hundreds, if not thousands, of people who have collaborated to build the  $\text{\TeX}$  ecosystem over the past few decades. While Tectonic positions itself as “outside of” traditional  $\text{\TeX}$  in a certain sense, the sincere intent is to credit and celebrate the work of all those people as fully as possible. With immense gratitude, I thank you for sharing your wonderful creation with the world.

## References

- [1] D.E. Knuth. *TEX: The Program*, vol. B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
- [2] J. Lawall, G. Muller. Coccinelle: 10 years of automated evolution in the linux kernel. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pp. 601–614, Boston, MA, July 2018. USENIX Association. [www.usenix.org/conference/atc18/presentation/lawall](http://www.usenix.org/conference/atc18/presentation/lawall)

◇ Peter K. G. Williams  
60 Garden St. MS-20  
Cambridge, MA 02138  
USA  
`pwilliams (at) cfa dot harvard dot edu`  
<https://newton.cx/~peter/>  
ORCID 0000-0003-3734-3587

## IoT theatre presents: The Tempest

Island of T<sub>E</sub>X (developers)

### Abstract

2021 was a challenging year for the Island of T<sub>E</sub>X: roadmap changes, lack of resources, server limitations. Yet, resilience, persistence and a bit of good humour made the island even stronger, with new joiners, community support, bold plans and an even brighter future for the T<sub>E</sub>X ecosystem. And all just in time for celebrating 10 years of arara, our beloved bird!

### 1 Introduction

For those who do not know anything about us, the Island of T<sub>E</sub>X started as a pair of friends trying to improve the T<sub>E</sub>X ecosystem. Nowadays, the island acts as a friendly hub to community-based T<sub>E</sub>X-related projects, still mostly focused on the tooling side of things.

The last year has been challenging for us: roadmap changes, lack of resources, server limitations. Yet, resilience, persistence and, of course, a bit of good humour made the island even stronger, with new joiners, community support, bold plans and an even brighter future for the T<sub>E</sub>X ecosystem.

### 2 Docker images

The Island of T<sub>E</sub>X provides Docker images for easily reproducible builds as well as an official response to the need for continuous integration. Our images were among the first using vanilla T<sub>E</sub>X Live, providing the required tools for running software included in T<sub>E</sub>X Live—for example, Java Virtual Machine, Python, and so forth. Additionally, we provide T<sub>E</sub>X Live releases from 2014 on and let the user decide whether they want to pull all the documentation and source files into their CI configuration.

We officially publish our images at Docker Hub: [hub.docker.com/r/texlive/texlive](https://hub.docker.com/r/texlive/texlive). Docker Hub is the world's largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors building and distributing their code in containers.

We feel honoured to maintain and provide one of the most complete and comprehensive T<sub>E</sub>X Live Docker images available in the T<sub>E</sub>X ecosystem. Several organisations now base their images on ours—for instance, DANTE e.V., the German-speaking T<sub>E</sub>X user group. Thank you very much for trusting our builds!

More recently, given certain characteristics of a typical T<sub>E</sub>X Live install, our shared CI runners could not meet the requirements to properly build and generate entries in our registry for historic and current images (e.g., failure due to timeout and disk space constraints). At first, we believed this was just a temporary issue (e.g., having a job assigned to a weak CI runner) and subsequent jobs would eventually have our images correctly built. Alas, we had no luck—failure rates were increasing at an alarming rate.

Vít Novotný reached out to us to discuss potential build improvements to our images and joined our Docker team. A new islander! His contributions were amazing—the build process soon became way more robust and reliable than ever. However, the blocking issue with our shared CI runners was still haunting us. We had to do something.

We agreed to reach out to the T<sub>E</sub>X community and ask for any spare computational resource that could host one of our CI runners. We then wrote an e-mail to the T<sub>E</sub>X Live mailing list on May 12 and hoped for the best.

In less than 30 minutes, we got three replies offering help! In that same day, the island already had a new dedicated CI runner. A couple of days later, two more runners were added to our pool. The response from the T<sub>E</sub>X community was fantastic.

Special thanks to Uwe Ziegenhagen, Erik Braun, DANTE, Marei Peischl, Paul Gessler, and the Institute of Mathematics of the Czech Academy of Sciences. We really appreciate it—our Docker images are saved because of you! Also, thanks to all who sent us messages of encouragement, from electronic mail to public support via our Matrix chat room.

Incidentally, Marei Peischl became an islander as well, joining our team of Docker experts. Thanks, Marei! Apparently, dragons are very good at building images!

Still, there is a long road to go. The T<sub>E</sub>X Live images are becoming more and more versatile but we still only feature full-fledged installations. So there are new frontiers waiting for us: splitting the images by scheme and improving layer-friendliness. Docker experts are invited to join our quest to solve these issues.

### 3 T<sub>E</sub>Xdoc online

From all those projects using our T<sub>E</sub>X Live images, T<sub>E</sub>Xdoc online is our most prominent contribution in the field—an online T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X documentation lookup system based on CTAN's JSON API. Alas, T<sub>E</sub>Xdoc online has not seen much activity in the last

year. It has a long way to go until it is not only a successor of the former `texdoc.net` but an even better replacement.

Some of our ideas include a source code lookup mode, based on a combination of `texdoc`, `kpsewhich` and file output, and runtime macro definition capabilities with a frontend for `texdef`. In general, the frontend should get a UI overhaul at some point in order to make it friendlier, more responsive, and more accessible.

We are also planning to include an optional analytics layer based on anonymous browsing and queries. But do not worry; at this point we are only interested in counting visits for starters — something which does not track anyone. Our generous sponsor of the world’s most accessed `TEXdoc` online instance at `texdoc.org`, Stefan Kottwitz, would love to see this feature implemented as well. By the way, thanks for updating and maintaining `texdoc.org`, Stefan!

If anyone is interested in tackling one (or more) of the aforementioned features, definitely get in touch with us. All you need is some kind of programming background. Picking up Kotlin and contributing is something we can help you with. And if you have further ideas for improvement for this or any of our projects, our issue trackers are open 24/7. We believe that community feedback is key to building software the community actually finds useful.

#### 4 CLI tooling

Apart from the web-based end, we should expect improvements to our CLI tooling in the near future as well. Take, for instance, our lovely Albatross, a tool for finding system fonts that provide a certain glyph (`ctan.org/pkg/albatross`).

Some of our ideas include report customisation, support for output formats (e.g., JSON or CSV), and caching. We also plan extending the glyph lookup based on specific files and directories, so any font in the filesystem — even ones not installed — could be properly inspected.

We’ve released a patched version of `checkcites`, our tool for checking missing or unused references, in order to address an outstanding issue due to a breaking `BIBATEX` update. (`ctan.org/pkg/checkcites`)

This tool also has an interesting roadmap. Some of our ideas include a complete rewrite from Lua to Kotlin (work in progress) and moving to a modular approach, in which we have a proper `BIBATEX` parser and a command line interface.

The ultimate goal for our `BIBATEX` parser is to produce native code for all major operating systems alongside a Java Virtual Machine-compliant bytecode for major vendors, as well as a JavaScript backend.

This would be the perfect place to talk about our plans for `arara` as they share many goals. But before we do that, we want to celebrate with you. So let us go back in time for a bit.

### 5 Ten years of `arara`

Ten years ago, the very first version of `arara`, the cool `TEX` automation tool, was released. It was a humble flight for such a little bird. Little did we know, a delightful story about friendship, `TEX`, community and noisy birds was about to be written.

#### 5.1 Version 1

There is a famous quote along the lines of

*If at first you do not succeed,  
call it version 1.0.*

Version 1 of `arara` was also the first public release, dated April 4, 2012. Nothing much was there, besides the core concepts that still exist today: rules and directives.

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.

#### 5.2 Version 2

The first version had a serious drawback: compilation feedback was not in real time and, consequently, no user input was allowed. For version 2, real time feedback was introduced when the tool was executed in verbose mode.

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.

#### 5.3 Version 3

So far, `arara` was only a tiny project with a very restricted user base. However, for version 3, 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 just about doubled from previous releases.

When the counter stopped at version 3, we decided it was time for `arara` to graduate and finally be released in `TEX Live`. Then things really changed in our lives. Given the worldwide coverage of `TEX` distributions, `arara` silently became part of the daily typographic tool belt of many users.

#### 5.4 Version 4

Version 4 was definitely a quantum leap from previous releases. New features included a REPL workflow (i.e., rule evaluation on demand as opposed to prior to execution), an improved rule format, support for multiline directives, partial directive extraction mode, commands and triggers as abstraction layers, and an improved lookup strategy for configuration files.

#### 5.5 Version 5

Version 5 featured a major rewrite from Java to Kotlin. We mainly worked on features from user feedback, especially directory support and the processing of multiple files.

We got hooked by the idea of aligning release schedules of arara with T<sub>E</sub>X Live releases enabling us to make (small) breaking changes more often. We had big plans and started to work on version 6 right after releasing version 5. As with checkcites, we walked the extra mile and arara went from a monolithic implementation to a modular one, plus an enhanced feature set and optimized workflow.

#### 5.6 Version 6

Version 6 was split between an API, a core implementation, the engine and the CLI application to separate concerns. This was the first step in the direction of splitting out components that are bound to one platform. New features included specifying command line options to be passed to arara's session map, default preambles, expansion within directives, safe mode, and rule improvements (towards safety and optimization).

#### 5.7 Version 7

For version 7, we wanted to take larger steps towards platform-independence. Some components had to be rewritten, some needed different interfaces for different platforms. It is still a heavy work in progress, but in the end, we hope to provide an even more multitalented tool.

Version 7 still targets JVM, but the tool is already being shaped towards platform-independence. New features include a new interface for the most common file operations (as a means to supersede Java's I/O API in the future), better error messages to indicate potential encoding problems, header mode enabled by default, and a brand new project specification.

## 6 The future

As previously stated, we are planning the stabilization of the current projects and implementation of new modular components, as well as improving support for our tools by producing native executables by means of Kotlin/Native, a technology for compiling Kotlin code to native binaries without the need of a JVM, languages like Rust and similar modern technology.

We also have challenges. For starters, hardware limitations for development and testing. The development happens on GNU/Linux machines. Incidental issues specific to Windows or macOS are handled through voluntary testing from users who sometimes do not have development expertise.

The lack of such systems in the development pool poses a problem and can hinder the long-term goals for better coverage and interoperability. See, for instance, the recent issues regarding Windows support in version 7 — we had to release three patches in quick succession to properly address these issues!

Also, the island has no documentation team, so we need to cover all fronts during development and release, not to mention the limited number of active developers and contributors.

Again, we kindly ask the T<sub>E</sub>X and software community for help, in any way possible. And, as always, thanks for the patience with us.

Also, a special thanks to our new members Jonathan Spratte, Marei Peischl and Vít Novotný! We are fortunate to have you on the team!

We have many plans and hope to realize as much as possible. The island is a vibrant environment for the development of T<sub>E</sub>X-related tools; we want to enhance the user experience, from newbie to expert, and promote use and diffusion of modern methodologies and technologies. If you are interested in helping us develop ideas or even implementing some code, visas for the island are free and no bureaucracy is involved, so feel free to reach out.

The Island of T<sub>E</sub>X is hosted at GitLab, whom we thank for providing us with a premium plan. It's highly appreciated.

If you are a T<sub>E</sub>X ecosystem tool author and want to join us, you and your projects are always welcome. If you want to become a tool author, or rewrite an existing tool, you are welcome as well!

◇ Island of T<sub>E</sub>X (developers)  
<https://gitlab.com/islandoftex>

---

## Using *knitr* and $\LaTeX$ for literate laboratory notes

Boris Veytsman

### 1 Introduction

Many years ago I worked in a lab that hired a new student. His assignment seemed to be easy: to reproduce the results of another student and to expand on them. The work in question included mathematical modeling, computer simulations, and so on. To his dismay, the new student found out that the programs used undocumented libraries, the scripts had incomprehensible options, and the models had unstated assumptions. Deciphering all this turned out to be very difficult. While the original author was willing to help, he could not do much: the author started to forget the details of his research soon after graduation.

This experience had a profound influence on me. The question of whether I would be able to understand my own research in a decade or two became an obsession. Being a scientist, I asked myself how the problem was solved elsewhere. In experimental and applied sciences the research may cost millions of dollars. To preserve it, the researchers are required to keep detailed logs of their activity in the laboratory notes. The notes include the details of the experiments and their results, and also the hypotheses tested. There are courses [14] and books [6] about laboratory notes. It is fascinating to study laboratory notes of great scientists, for example, Linus Pauling. Pauling's notes comprise 46 notebooks spanning from 1922 to 1992, digitized by Special Collections & Archives of the Oregon State University [13]. His beautiful notes have a definite aesthetic value.

I would argue that the concept behind the practice of laboratory note keeping is somewhat akin to the concept of literate programming [8]. Knuth understood that code is just part of a programmer's output. The programmer's thoughts about these programs are even more important. Similarly, an important insight for science is that papers, preprints, presentations are not the research, but "an advertisement of the research" [15]. We must preserve the research itself [11, 15]. Laboratory notes are the means for this preservation.

Classic laboratory notes are physical notebooks like those of Linus Pauling. Unfortunately, this format has a number of flaws:

1. Physical notes are not searchable. While it is recommended to add a table of contents to a notebook [6, 14], it is time-consuming to keep it current, and has only limited value for a search.

2. Copying from physical notes is not easy. This is especially frustrating with code: retyping from the printouts pasted to the notebook pages is time-consuming and error-prone.
3. Physical notebooks are bulky.

Electronic notebooks may be searchable, allow easy copying and pasting, can be stored infinitely (if the proper backups are kept), and take miniscule space. On the other hand, physical notebooks are versatile, and one can write and doodle in them very quickly. It is difficult to match their convenience for the laboratory record keeping.

### 2 What should electronic laboratory notes store?

Electronic laboratory notes should allow the user to easily store a number of disparate items. As a theoretician, I put in my notes:

1. Prose. Sometimes short texts, sometimes longer paragraphs.
2. Equations, both inline and displayed.
3. Code snippets of various length.
4. Tables, sometimes produced automatically by code.
5. Plots, often produced automatically by code.
6. Sketches, doodles, diagrams, etc.
7. Bibliographies.

The standard  $\LaTeX$  features like automatic numbering of objects, font changes, etc., may help to make the notes more readable and expressive.

### 3 Examples of notebook interfaces

Many commercial and free programs have so-called "notebook interfaces" for exploratory studies. These interfaces try to solve the same problem as laboratory notebooks: documentation of research. In this section we discuss two free programs. The first, *wxMaxima* [20], is a document-based interface for the famous computer algebra system *Maxima* [10]. The second, *Jupyter notebooks* [7], is intended "to support interactive data science and scientific computing". Jupyter notebooks were initially developed for Python programming, but have been extended to more than 40 computer languages.

Both these solutions are based on similar ideas, which are also used in many other notebook interfaces. They have a linear sequence of "cells" of different kinds. A text cell contains documentation. A code cell contains a program snippet. This cell could be "run": the program snippet is executed, and an output cell is added to the notebook. There are other types of cell to introduce metadata, section

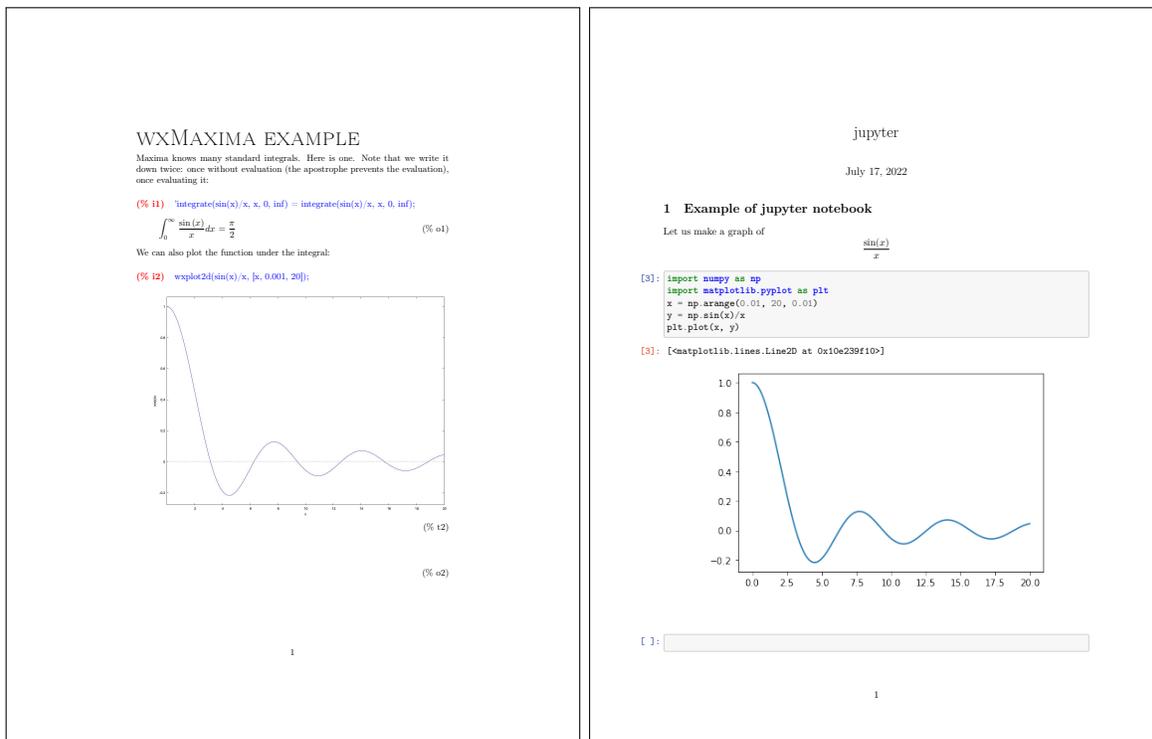


Figure 1: Examples of the output of wxMaxima and Jupyter notebooks

headers, etc. The text cells in Jupyter notebooks use the Markdown language [4], including  $\text{\LaTeX}$  math syntax. These programs can export a document with the record of the session in either PDF or  $\text{\TeX}$  format (Figure 1).

These notebooks are useful, and they are much better than no documentation at all. However, they do not satisfy many of the requirements stated in Section 2. We cannot easily number and reference objects. The output cells are not typically rendered as Markdown code, so we cannot typeset tables created by the code.

There are two reasons for these deficiencies. First, Markdown is not as expressible as  $\text{\LaTeX}$ . There are extensions like *bookdown* [22] which alleviate this problem. However, as far as I know, most notebook interfaces do not support these extensions. Also, Markdown extensions make the language more  $\text{\LaTeX}$ -like, which questions the whole premise of a simple typesetting language. Some readers may recall the famous phrase by Henry Spencer about the people who are condemned to reinvent Unix [19].

The second reason is more deep. The notebook interfaces are primarily records of the interaction between the user and the computer. The ideas of literate science, like the ideas of literate programming, suggest the centrality of the interaction between the user and other humans. Notebooks are basically code

with text inserts. A literate interface should be the opposite: text with code inserts. This approach is discussed in the next session.

#### 4 knitr-based notebooks

It is interesting that most notebook interfaces use  $\text{\TeX}$  as the back end typesetting engine, even when Markdown is the front end. This leads to the idea of  $\text{\LaTeX}$  as the laboratory notebook language. A set of  $\text{\TeX}$  files is easily searchable with standard utilities such as `grep` and `find`, while PDF output provides readable documents. This approach satisfies almost all requirements listed in Section 2, with one exception: we want to add both snippets of programs *and* their output as tables and plots. While adding program code can be achieved within  $\text{\LaTeX}$  (using, for example, the *listings* package [5]), the automatic addition of its output requires other means. The *knitr* package [21], based on the ideas of *Sweave* [9], can be used to create literate science [18].

A document in this case is a  $\text{\LaTeX}$  file (with extension `.rnw`) that contains “chunks” of code. Initially, only R code was supported by knitr. Now knitr, like Jupyter notebooks, has been extended to other program languages, including Python. When processed by knitr, the chunks are typeset and their output is added to the document.

Using *knitr* and  $\text{\LaTeX}$  for literate laboratory notes

```
<<plot, dev='tikz', message=F>>=
library(tidyverse)
library(ggthemes)
theme_set(theme_bw())
data <-
  tibble(x=seq(0.01, 20, by=0.01)) %>%
  mutate(y=sin(x)/x)
ggplot(data) + geom_line(aes(x,y)) +
  xlab("$x$") +
  ylab("$\\sin(x)/x$")
@
```

**Figure 2:** knitr chunk (in R) for plotting  $\sin(x)/x$

For example, consider the chunk in Figure 2. It programs a plot. When processed by knitr, both the typeset code and the plot are included in the  $\text{\TeX}$  file (Figure 3). There are options to suppress the typeset code, change the graphics format, etc. [21]. For example, the chunk in Figure 2 uses the *tikz* format, so the plot has math typeset by  $\text{\TeX}$ .

An interesting feature of *knitr* is the ability to re-render the output in  $\text{\TeX}$ . This feature can be used to automatically produce tables. Consider, for example, the extrema of the function  $f(x) = \sin(x)/x$ . We can calculate them by solving numerically the equation  $x \cos(x) - \sin(x) = 0$ , obtained by differentiating  $f(x)$ . The R program in Figure 4 calculates the first six extrema at  $x \geq 0$ . It outputs six lines (for example: 5 & 14.07 & 0.07). To typeset the table we output these lines as raw  $\text{\TeX}$  code (with the chunk options `result='asis'`, `echo=F`), and wrap it in a `tabular` environment. The result is shown in Table 1.

## 5 Problems with the knitr-based solution

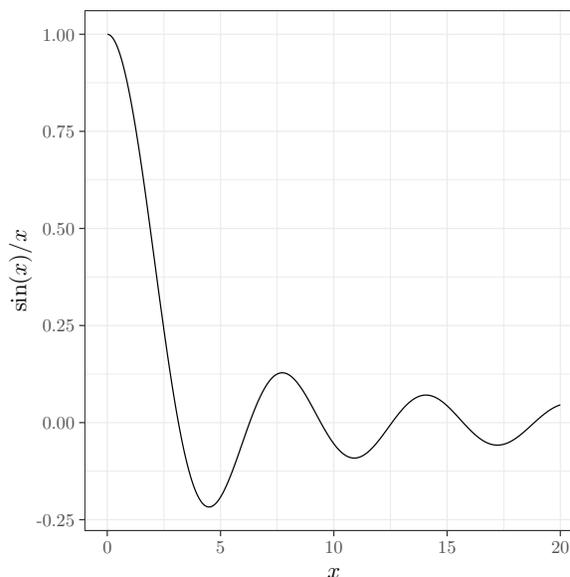
The solution based on knitr is powerful. However, it has its own problems.

The first set of problems is related to the deficiencies of PDF format. This format is static by design. Sometimes we want to include movies, animation or interactive plots. It is possible to do

Number	$x$	$f(x)$
1	0	1
2	4.49	-0.22
3	7.73	0.13
4	10.9	-0.09
5	14.07	0.07
6	17.22	-0.06

**Table 1:** First six extrema of  $f(x) = \sin(x)/x$  (see the code in Figure 4)

```
library(tidyverse)
library(ggthemes)
theme_set(theme_bw())
data <-
  tibble(x=seq(0.01, 20, by=0.01)) %>%
  mutate(y=sin(x)/x)
ggplot(data) + geom_line(aes(x,y)) +
  xlab("$x$") + ylab("$\\sin(x)/x$")
```



**Figure 3:** The typeset code (grayscaled for print) and plot produced by the chunk on Figure 2

```
find_extremum <- function(number) {
  result <-
    uniroot(
      function(x) {
        x*cos(x) - sin(x)
      },
      c((number-1)*pi,
        number*pi))
  x <- result$root
  f <- ifelse(x==0, 1,
             sin(x)/x)
  cat(number, "&", round(x, 2),
       "&", round(f, 2),
       "\\n\\n")
}
walk(1:6, find_extremum)
```

**Figure 4:** Code for calculation of extrema of  $\sin(x)/x$ . The results are shown in Table 1.

this using packages like *media9* [3], *animate* [2], and *FigPut* [1]. However, the recent debacle of Adobe Flash [12] makes one wary of PDF extensions.

Another problem is related to the speed of writing laboratory notes. I personally type prose *and* equations in L<sup>A</sup>T<sub>E</sub>X with the same speed I produce them. The same can be said about programming. However, doodling with a pen and paper is substantially faster than writing code in P<sub>S</sub>Tricks [17] or TikZ [16]. Thus sketching may require different solutions: from scanning handwritten images to the use of special programs for fast doodling.

## 6 Conclusions

L<sup>A</sup>T<sub>E</sub>X allows the production of detailed laboratory notebooks, that can be easily read, searched and indexed. The addition of knitr helps to integrate the notebooks with the inclusion of typeset code and its output, such as plots, tables, etc.

This has been my preferred format of laboratory notebooks for several decades. It is quite versatile, reasonably fast and provides notes of archival quality.

## References

- [1] R. Fairman. *FigPut. Interactive Figures for L<sup>A</sup>T<sub>E</sub>X*, 2022. [ctan.org/pkg/figput](https://ctan.org/pkg/figput)
- [2] A. Grahn. *The animate package*, 2022. [ctan.org/pkg/animate](https://ctan.org/pkg/animate)
- [3] A. Grahn. *The media9 package, v1.24*, 2022. [ctan.org/pkg/media9](https://ctan.org/pkg/media9)
- [4] J. Gruber. *Markdown*, 2004. [daringfireball.net/projects/markdown/](https://daringfireball.net/projects/markdown/)
- [5] C. Heinz, B. Noses, J. Hoffmann. *The Listings Package*, 2020. [ctan.org/pkg/listings](https://ctan.org/pkg/listings)
- [6] H.M. Kanare. *Writing the laboratory notebook*. American Chemical Society, Washington, D.C., 1985.
- [7] T. Kluyver, B. Ragan-Kelley, et al. Jupyter notebooks—a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides, B. Schmidt, eds., pp. 87–90. IOS Press, 2016.
- [8] D.E. Knuth. *Literate Programming*. No. 27 in CSLI Lecture Notes. Stanford, California, 1992.
- [9] F. Leisch, R Core Team. *Sweave User Manual*, 2022. [stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf](https://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf)
- [10] Maxima. A computer algebra system, 2022. [maxima.sourceforge.io/](https://maxima.sourceforge.io/)
- [11] J.P. Mesirov. Accessible reproducible research. *Science* 327(5964):415–416, 2010. [10.1126/science.1179653](https://doi.org/10.1126/science.1179653)
- [12] R.C. Moss. The rise and fall of Adobe Flash, 2020. [arstechnica.com/information-technology/2020/07/the-rise-and-fall-of-adobe-flash/](https://arstechnica.com/information-technology/2020/07/the-rise-and-fall-of-adobe-flash/)
- [13] L. Pauling. Research notebooks, 1922–1972. Special Collections & Archives Research Center, Oregon State University Libraries. [scarc.library.oregonstate.edu/coll/pauling/rnb/](https://scarc.library.oregonstate.edu/coll/pauling/rnb/)
- [14] P. Ryan. *Keeping a Lab Notebook*. National Institutes of Health, Office of Intramural Training and Education, 2012. [www.training.nih.gov/assets/Lab\\_Notebook\\_508\\_\(new\).pdf](https://www.training.nih.gov/assets/Lab_Notebook_508_(new).pdf)
- [15] M. Schwab, N. Karrenbach, J. Claerbout. Making scientific computations reproducible. *Computing in Science & Engineering* 2(6):61–67, 2000. [10.1109/5992.881708](https://doi.org/10.1109/5992.881708)
- [16] T. Tantau. *The TikZ and PGF Packages*, 2021. [ctan.org/pkg/pgf](https://ctan.org/pkg/pgf)
- [17] T. Van Zandt, R. Niepraschk, H. Voß. *P<sub>S</sub>Tricks. PostScript macros for Generic T<sub>E</sub>X*, 2007. [ctan.org/pkg/pstricks-base](https://ctan.org/pkg/pstricks-base)
- [18] B. Veytsman. Book review: Dynamic Documents with R and knitr, by Yihui Xie. *TUGboat* 35(1):115–119, 2014. [tug.org/TUGboat/tb35-1/tb109reviews-xie.pdf](https://tug.org/TUGboat/tb35-1/tb109reviews-xie.pdf)
- [19] Wikipedia contributors. Henry Spencer — Wikipedia, the free encyclopedia, 2022. [en.wikipedia.org/w/index.php?title=Henry\\_Spencer&oldid=1093428638](https://en.wikipedia.org/w/index.php?title=Henry_Spencer&oldid=1093428638)
- [20] wxMaxima, 2022. [wxmaxima-developers.github.io/wxmaxima/](https://github.com/wxmaxima-developers/wxmaxima/)
- [21] Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton; London; New York, second ed., 2015.
- [22] Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida, 2016. ISBN 978-1138700109. [bookdown.org/yihui/bookdown](https://bookdown.org/yihui/bookdown)

◇ Boris Veytsman  
Systems Biology School  
George Mason University  
Fairfax, VA 22030  
[borisv \(at\) lk dot net](mailto:borisv(at)lk(dot)net)  
<http://borisv.lk.net>

## ***y*Ex: a T<sub>E</sub>X-alike typesetter in Python**

Marnanel Thurman\*

### **Abstract**

*y*ex is an implementation of the core T<sub>E</sub>X system in pure Python. This article gives an overview of its development, the challenges faced, and possible future directions for the project.

### **1 Introduction**

*y*Ex is a T<sub>E</sub>X emulation written in pure Python. It aims to be as faithful a recreation of the T<sub>E</sub>X core as is possible. It has a strong test suite and plentiful inline documentation.

While I began the project as a means of learning T<sub>E</sub>X better, it has grown beyond that in every direction and continues to be an ongoing project. Because of this origin, I’m using *The T<sub>E</sub>Xbook* as a spec rather than working from the WEB sources.

Any reimplementaion, especially one which reimplements a compiled system in an interpreted language, necessarily has a set of goals which differ from the original: this serves to add diversity and robustness to the T<sub>E</sub>X ecosystem. Here are mine.

Firstly, *y*Ex aims to get things right before aiming for completeness: that is, it values *depth over breadth*. The most commonly-used functionality is implemented, and it can set basic documents successfully.

Secondly, Python has a rich *existing infrastructure*: a wonderful resource. *y*Ex is making as much use of that as possible. PDF handling can be handled by Python’s existing PDF libraries; Markdown will be handled likewise; the library Beautiful Soup takes care of XML and HTML handling; and so on. *y*Ex itself is already available in the Python package index.<sup>1</sup>

Where possible, parts of *y*Ex will be split out into their own general-purpose libraries, so that they may better interact with other people’s projects. It’s a fine thing to create something beautiful. It’s a thousand times better to build a source of creativity for others.

Lastly, *HTML output* is a particular focus of *y*Ex. Although T<sub>E</sub>X is well-suited for producing this format, it is surprisingly underused. More on that shortly.

### **2 Where are we so far?**

- We can set documents!
- SVG output, for debugging

\* Thanks to Kit Thurman for proofreading.

<sup>1</sup> [pypi.org/project/yex/](https://pypi.org/project/yex/)

- Serialisation
- Basic HTML output

Next steps:

- Full coverage of T<sub>E</sub>X controls
- PDF output
- Caching

### **3 Serialisation**

All of *y*Ex’s internal data structures can be serialised to simple JSON types, as can all processed documents. There are command line switches to output these formats. This is useful in debugging, as well as in integrating third-party systems outside Python.

This also enables caching. At present, for example, `plain.tex` takes several seconds to process. This would make it impractical to use as a library. However, with serialisation, it can be stored and retrieved at speed.

### **4 Docstrings**

Python classes and functions can be documented inline, using a literate programming feature known as “docstrings”. These can be formatted using a markup system, such as Restructured Text or Markdown. The docstrings are converted to HTML by a tool called Sphinx, for user help systems. Examples can be seen on sites such as Read The Docs.<sup>2</sup>

*y*Ex will include a new Sphinx plugin to accept T<sub>E</sub>X formatting, so that *y*Ex can produce its own documentation.

### **5 Input filters**

It will also be possible to add input filters for other formatting systems to *y*Ex itself, so that *y*Ex documents can include inline HTML or Markdown. Each format will have a stylesheet of macros, one per tag type. The macros will be called to handle each tag, to represent their meaning to *y*Ex.

### **6 HTML output**

The current focus of development is HTML output. One of the historical barriers to HTML output from T<sub>E</sub>X has been wordwrap. HTML is built to reflow text on the fly — shoddily, compared to T<sub>E</sub>X’s characteristically careful breaking of paragraphs into lines.

However, modern HTML toolkits such as Bootstrap use a different approach. They divide available display devices into “breakpoint”<sup>3</sup> classes, based on the viewport width: small (for devices such as mobile phones), medium (for laptops), large, and extra

<sup>2</sup> See, for example, [yex.readthedocs.io](https://yex.readthedocs.io).

<sup>3</sup> See [getbootstrap.com/docs/5.2/layout/breakpoints](https://getbootstrap.com/docs/5.2/layout/breakpoints). This term “breakpoint” is unrelated to T<sub>E</sub>X’s use of the word.

large. This permits pages to adapt according to the device in use, an ability known as “responsiveness”. The system exists for the sake of more complex formatting than merely wordwrap, but it suits  $y^{\text{Ex}}$ ’s purposes well.

This allows us to add an `\everypar` rule to the HTML output stylesheet, causing each paragraph to be processed four times, each with a different `\hsize`:

```
\let\endgraf=\par \let\endline=\cr
\def\wip##1\par{\let\widthspara=\relax
\special{html.responsive.start}\endgraf
\hsize=432pt
##1\endgraf
##\special{html.responsive.again}\hsize=576pt
##1\endgraf
##\special{html.responsive.again}\hsize=744pt
##1\endgraf
##\special{html.responsive.again}\hsize=992pt
##1\endgraf
\special{html.responsive.done}%
\let\widthspara=\wip}%
\let\widthspara=\wip
\everypar={\widthspara}
```

The `\special` directives tell the output driver to treat these as four versions of the same paragraph. The driver will meld them together, as with Bootstrap breakpoints, so that one is used on mobiles, one on laptops, and so on. The CSS makes the choice of which version to use based on the width of the viewport.

The widths here are taken from Bootstrap’s breakpoint specification, where they are given in pixels.  $y^{\text{Ex}}$  allows widths to be specified in pixels (`px`) in addition to  $\text{T}\text{E}\text{X}$ ’s standard units. For interoperability, we give them in points, using W3C’s definition of 96 pixels to the inch:<sup>4</sup> this makes a pixel equal to exactly 49152sp.

## 7 Impedance mismatches

There have been many challenges to overcome so far, even beyond the work of reimplementing of a system as complex as  $\text{T}\text{E}\text{X}$ —not to mention the writing of a test suite to prove it all works!

One major factor has been the distance between the priorities of  $y^{\text{Ex}}$  and  $\text{T}\text{E}\text{X}$ , which reflects the forty-year distance between them. Unsurprisingly for a program designed in the late 1970s,  $\text{T}\text{E}\text{X}$  has a general assumption of scarcity. There are only so many registers of each kind. Python expects you to say what you need and assume that the resources

will be found. This difference in approach makes implementation far more interesting. For example, producing call stack traces for errors in  $\text{T}\text{E}\text{X}$  macros proved aggravatingly difficult. A  $\text{T}\text{E}\text{X}$  macro can be curried by omitting its final argument, thus:

```
\def\#1#2{Hello, #2}
\def\b{\a x}
{\b world}
```

Because of this, the state of  $\text{T}\text{E}\text{X}$ ’s stack can’t be mirrored by the Python stack. Further, any part of the  $\text{T}\text{E}\text{X}$  code above might be pushed onto  $\text{T}\text{E}\text{X}$ ’s token stack, and the call stack would still need to remain consistent. The solution involves a special class of token, `Internal`, which runs a given Python callback on being processed. These tokens can’t be generated by the tokeniser; they are used for macro prologues and epilogues to maintain the call stack.

Another example is encapsulation: an important principle in Python.  $\text{T}\text{E}\text{X}$  is not careful with namespaces. One of the headaches this causes is that the order of loading  $\text{T}\text{E}\text{X}$  libraries can easily affect the results. That means that libraries can’t be cached individually: the cached value of a library will vary according to which libraries were loaded before it. Thus  $y^{\text{Ex}}$  must wait until it’s seen all the initial `\input` commands before making any decisions about caching. When it knows the full list of libraries needed, it must reload them from the cache in order, as a group.

## 8 The future

$y^{\text{Ex}}$ ’s initial goal is to be able to typeset *The  $\text{T}\text{E}\text{X}$ -book*. That’s still a long way off, though of course the speed at which we get there depends on how many people share the work. Contributions are always welcome! Visit [gitlab.com/marnanel/yex/](https://gitlab.com/marnanel/yex/) for the source.

Beyond that, I have a goal to make sure  $y^{\text{Ex}}$  gives solid results with as many of the packages in the standard  $\text{T}\text{E}\text{X}$  distributions as possible. Processing  $\text{\LaTeX}$  will be a very important step, though a huge one.

There are also many other  $\text{T}\text{E}\text{X}$ -like projects whose ideas we can share, many of which are being discussed at this conference. I look forward to seeing how we can work together.

◇ Marnanel Thurman  
<https://gitlab.com/marnanel/yex>

<sup>4</sup> [www.w3.org/Style/Examples/007/units.en.html](http://www.w3.org/Style/Examples/007/units.en.html)

## Extracting information from (L<sup>A</sup>)T<sub>E</sub>X source files

Jean-Michel HUFFLEN

### Abstract

We present some tools that allow us to parse all or part of (L<sup>A</sup>)T<sub>E</sub>X source files and process suitable information. For example, we can use them to extract some metadata of a document. These tools have been developed in the Scheme functional programming language. Using them requires only basic knowledge of functional programming and Scheme. Besides, these tools could be easily implemented using a strongly typed functional programming language, such as Standard ML or Haskell.

### 0 Introduction

In many places, it has been told or written that T<sub>E</sub>X is a wonderful tool for typesetting texts. But it deals only with its own formats: that is well-known, too. However, the information contained in source file texts processed by T<sub>E</sub>X—or any format or engine built out of it—may be of interest for purposes other than typesetting, e.g., enriching the metadata usable by Web search engines.

Doing such jobs by means of (L<sup>A</sup>)T<sub>E</sub>X commands arranged into an option of a class or a package is possible, but we think that this is *misusing* T<sub>E</sub>X. From our point of view, this tool does not aim to be a universal multi-task program, able not only to typeset texts, but also to generate Web pages or fulfill any other purpose we can imagine. From a point of view related to theoretical computer science, T<sub>E</sub>X's language has the same expressive power as a Turing machine, so any function can be programmed using T<sub>E</sub>X's primitives,<sup>1</sup> but as with any specialised language, using it for a purpose other than its intended one is tedious.<sup>2</sup> In addition, this language's syntax is old, its parsing uses old-fashioned conventions, it does not provide advanced data structures, as we can find in many more recent programming languages.

Hereafter we describe a way to connect Scheme functions to T<sub>E</sub>X commands when a (L<sup>A</sup>)T<sub>E</sub>X source file is parsed and these commands recognised. Our basic idea is that often only a little information is relevant, e.g., the metadata of a document. Extracting them from (L<sup>A</sup>)T<sub>E</sub>X source files allows us to avoid *information redundancy*. Section 2 explains the ori-

<sup>1</sup> Interested readers can consult [2, 19] about this subject.

<sup>2</sup> As another accurate example, any programmer knows that using Prolog [4] outside logic programming is quite painful.

gins and reasons for our choices, discussed further in Section 3. Reading this article requires only basic knowledge about T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X commands [14, 18] and the division of a L<sup>A</sup>T<sub>E</sub>X source file into a *preamble* and *body*. Some basic notions of programming in Scheme are needed, too, as can be found in any good introductory book to this functional programming language, e.g., [23].

## 1 Our Scheme library

### 1.1 Why Scheme?

As mentioned above, we aim to extract accurate information from (L<sup>A</sup>)T<sub>E</sub>X source files; we are not interested in processing the whole of such a file; we do not want to put a ‘new T<sub>E</sub>X program’ into action.

Now let us recall that in functional programming, functions are first-class objects, just like other data. So functions can be arguments or results of a computation. This feature allows us to write *generators* of functions. Our tool is a wonderful example of such a generator. You choose which information you would like to retain and how you plan to process it. This step is done by a computation which returns a function. This second function's argument is the input filename to be parsed.

In Section 2, we will see that some parts have already been written using Scheme [22] for several years. Let us recall that within this functional programming language—as within any Lisp dialect—data and programs have the same format. Hereafter, the description of our library's main features emphasises that functions and other data are mixed by means of a unique format.

### 1.2 How to use our library

Building a function parsing a (L<sup>A</sup>)T<sub>E</sub>X source file is done by the construct:<sup>3</sup>

```
(g-mk-tex-parsing-f directive ...)
```

with any number of *directives*.<sup>4</sup> There are *two* kinds of directives:

```
(g-retain-command command-name arg-nb
  optional-arg? top-level?
  recursive? preamble?
  occ-nb-info function)
(g-retain-match command-name s top-level?
  recursive? preamble?
  occ-nb-info function)
```

<sup>3</sup> Let us recall that Scheme systematically uses *prefixed* syntax. All the definitions introduced by our library are prefixed by ‘g-’.

<sup>4</sup> This is the terminology used within our source files. You can use `g-mk-tex-parsing-f` without arguments—that is, *no directive*—in which case the result will just move along the file's preamble without performing any other operation.

where:

*command-name* is the name of the command to be caught, without the initial ‘\’ character;

*arg-nb* is the argument number for this command;

*optional-arg?* is true<sup>5</sup> if the first argument is optional, surrounded by square brackets,<sup>6</sup> false otherwise;

*top-level?* is true if we have to look for this command only at the top level, false otherwise;

*recursive?* is used when `\input` commands are encountered: if it is true, corresponding files are searched recursively, otherwise such an `\input` command is just skipped;

*preamble?* stops searching after a preamble if it is bound to true; otherwise, search goes on;

*occ-nb-info* may be bound to:

- 0 or the false value: we check that this command does not appear within files;
- a positive integer *n*: the first *n* occurrences of this command are processed, and following ones are ignored;
- the true value: all the occurrences of this command are processed;

*function* the Scheme function to call; it *must* accept the same number of arguments than the `\command-name` command. All the arguments of such a function are supposed to be *strings*.

We can see that the directives introduced by the `g-retain-command` function are suitable for most  $\LaTeX$  commands, possibly with a leading optional argument. More difficult cases are handled by the `g-retain-match` function: its second argument is the command’s *pattern*, given as a string, according to  $\TeX$ ’s conventions used by the `\def` primitive, the command’s name being omitted. Here are two examples:

```
\csname ← "#1\endcsname"
\ifx ← "#1#2#3\else#4\fi"
```

All the other arguments of this `g-retain-match` function have the same meaning as the namesake arguments of `g-retain-command`. Let us notice that `g-retain-match` and `g-retain-command` are functions, whereas `g-mk-tex-parsing-f` is a *macro*.<sup>7</sup>

<sup>5</sup> Let us recall that the boolean values *true* and *false* are expressed in Scheme by the expressions `#t` and `#f` respectively.

<sup>6</sup> That is, according to  $\LaTeX$ ’s conventions [15].

<sup>7</sup> Let us recall that Scheme uses a *call-by-value* strategy for functions: arguments are evaluated before applying the function. Defining `g-mk-tex-parsing-f` as a macro allows us to install the structures we need, before applying the directives to populate these structures, and finally building the parsing function. The process put into action by that macro may be viewed as a kind of *compiling*.

The result of `g-mk-tex-parsing-f` is a function that applies to a filename. It parses this file by performing *one* pass and returns:

**false** if something went wrong, or a forbidden command is included into the file;

**true** in all other cases.

You have to use Scheme functions interfaced with  $\TeX$  constructs to update your own structures when a file is parsed. Beware that if an error occurs, these structures may be in an inconsistent state.

### 1.3 Other functions

Scheme’s initial library and our basic functions include a rich set of functions dealing with strings. For example, *s* being a string:

(`normalize-space s`) whitespace-normalises the *s* string, that is, leading and trailing spaces are stripped, multiple occurrences of whitespace are replaced by a single space character; the result is a newly allocated string.

The next two functions can be useful to destructure an argument of a  $\TeX$  command; the successive characters of the *s*<sub>0</sub> string are supposed to be a comma-separated list, *s*<sub>1</sub> is any string:

(`g-parse-to-list s`<sub>0</sub>) returns its elements within a linear list, e.g.:

```
(g-parse-to-list "New-York, New-York")
⇒ ("New-York" "New-York")
```

(`g-parse-to-alist s`<sub>0</sub> *s*<sub>1</sub>) returns the successive pairs *key=value* of *s*<sub>0</sub> within an *association list*; if a key is given without a value, this missing value is replaced by *s*<sub>1</sub>, e.g.:

```
(g-parse-to-alist "town=LA,state" "CA")
⇒ (("town" . "LA")
    ("state" . "CA"))
```

In both cases, the original order is preserved.

### 1.4 A simple example

As a simple example, let us consider a source text for  $\LaTeX$ . We would like to know:

- its title,
- the options given to the `babel` package<sup>8</sup> [18, Ch. 9] if it is loaded,
- the number of occurrences of the `\emph` command.

The function we build and run is given in Fig. 1. Some remarks:<sup>9</sup>

<sup>8</sup> We do not consider the ‘`main=...`’ construct.

<sup>9</sup> You may notice that we specify the commands of interest in alphabetical order. This is just a personal habit; the order of directives inside the `g-mk-tex-parsing-f` macro is irrelevant.

```

(define tug-2022-example-emph-occ-nb '0) ; Initialisations needed.
(define tug-2022-example-language-name-list '*dummy-value* ; (...)
(define tug-2022-example-title '*dummy-value*)

(define tug-2022-example-function
  (g-mk-tex-parsing-f (g-retain-command "emph" 1 ; One argument.
                                #f ; No optional argument.
                                #f ; May be located at any level.
                                #f ; Look for it recursively.
                                #f ; Search the preamble and body.
                                #t ; Process all the occurrences of this command.
                                (lambda (ignored-s) ; The argument is ignored.
                                  (set! tug-2022-example-emph-occ-nb
                                    (+ tug-2022-example-emph-occ-nb 1))))
    (g-retain-command "title" 1 #f ; One non-optional argument.
                      #t ; Top level only.
                      #f ; Recursively search.
                      #f ; Search the preamble and body.
                      1 ; Process only the first occurrence.
                      (lambda (title-s)
                        (set! tug-2022-example-title (normalize-space title-s))))
    (g-retain-command "usepackage" 2 #t ; Two arguments, including an optional one.
                      #t ; Search only at the top level.
                      #t ; Recursive search.
                      #t ; Search only the preamble.
                      #t ; Process all the occurrences.
                      (lambda (option-s package-names-s)
                        (when (member "babel" (g-parse-to-list package-names-s)
                                      string=?))
                          (set! tug-2022-example-language-name-list
                            (g-parse-to-list option-s))))))

(tug-2022-example-function "<this article's source file>") => #t ; Parsed successfully!

tug-2022-example-emph-occ-nb => 39 ; Variables updated.
tug-2022-example-language-name-list => ("french" "english") ; (...)
tug-2022-example-title => "Extracting Information from \AllTeX\ Source Files"

```

Figure 1: Example of using our Scheme functions.

- the `\title` command may or may not be given in the preamble, but is unique;
- if `babel` package is loaded, it can only be located in the preamble; but there may be *several* `\usepackage` commands, possibly for other packages;
- the innermost occurrences of the `\emph` command are processed first: some additional details about this point are given in App. A.

The evaluation given in Fig. 1 applies to the source of the present text. The three Scheme variables used are initialised at Fig. 1's top.

### 1.5 Types used

Scheme is a *dynamically typed* language. This property allows variables to be bound to a value being any type, *a priori*. Scheme is not *strongly* typed,

since variables are not given types, as in the C programming language [13]. This feature may be viewed as an advantage or drawback, depending on programmers' feelings. However we mention that our tool could be implemented using a strongly-typed functional programming language, such as Standard ML [20] or Haskell [21]. Let us recall that programmers of these languages do not have to put down the types associated with variables, but a type-checking mechanism is in charge of determining such types. If this operation fails, your program is rejected. So in practice, programmers of these languages pay great attention to types used.

When arguments of our directives are strings or booleans — true or false values — there is no problem. The information about the number of occurrences to be processed can be viewed as the *union* of natural numbers and boolean values. Since these

two sets are disjoint, modern strongly-typed functional programming languages can implement such a construct by means of a *disjoint union*:<sup>10</sup>

$$\text{Occ-nb-info-type} \stackrel{\text{def}}{=} \text{Boolean} \uplus \text{Natural}$$

The type of the functions connected to  $\text{T}_{\text{E}}\text{X}$  commands can be specified by a direct sum, too, due to a limitation of  $\text{T}_{\text{E}}\text{X}$ . Let us consider that all the possible results of such a function are encompassed into a type called *Result*. Let  $n$  be a natural number, the type of a function associated with a  $n$ -argument command is  $\text{String}^n \rightarrow \text{Result}$ , where *String* is the type of strings.<sup>11</sup> Since the greatest argument number for a  $\text{T}_{\text{E}}\text{X}$  command is ‘#9’ [14],<sup>12</sup> the complete functions are finally of the type:

$$\text{Function-for-}\text{T}_{\text{E}}\text{X} \stackrel{\text{def}}{=} \bigsqcup_{0 \leq i < 10} (\text{String}^i \rightarrow \text{Result})$$

## 2 History

### 2.1 Genesis

Let us recall that we implemented  $\text{MIBIB}\text{T}_{\text{E}}\text{X}$ <sup>13</sup>, a possible successor of  $\text{BIB}\text{T}_{\text{E}}\text{X}$ , the bibliography processor that was commonly associated with  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  for a long time. In particular,  $\text{MIBIB}\text{T}_{\text{E}}\text{X}$  has aimed to ease the production of *multilingual* bibliographies.

When we put  $\text{MIBIB}\text{T}_{\text{E}}\text{X}$ ’s first public version into action [9], we realised that we needed to parse the beginning of source *.tex* files, in order to get the way to process the languages used throughout a document; this information was not given in *.aux* files.<sup>14</sup> There was at most one occurrence of loading the *babel* package or an *ad hoc* package such as *french* or *polski*.<sup>15</sup> Such a load order could be located in a subfile grouping the packages for the set up of a document. On another point, we did not have to parse the whole of a  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  document: we stopped either after encountering such a load order, or en-

<sup>10</sup> Let  $S_0$  and  $S_1$  be two sets, the **disjoint union** [8] of  $S_0$  and  $S_1$  is defined by:

$$S_0 \uplus S_1 \stackrel{\text{def}}{=} (\{0\} \times S_0) \cup (\{1\} \times S_1)$$

If we connect this formula with an abstract data type definition, ‘0’ and ‘1’ may be viewed as the *constructors* of this data type.

<sup>11</sup> This definition includes zero-argument commands, since a zero-argument function  $f_0 : \rightarrow \text{Result}$  may be viewed as  $f_0 : \{\emptyset\} \rightarrow \text{Result}$ , as mentioned by [6]. In programming languages such as Standard ML or Haskell, the  $\{\emptyset\}$  set is implemented by the *unit* type, containing only the () value.

<sup>12</sup> There are workarounds if more arguments are needed, as explained in [14]. This point is obviously out of this article’s scope.

<sup>13</sup> **M**ulti**L**ingual **B**IB $\text{T}_{\text{E}}\text{X}$ .

<sup>14</sup> Incidentally,  $\text{BIB}\text{T}_{\text{E}}\text{X}$  only parses *.aux* files and *never* reads *.tex* files.

<sup>15</sup> At this time, the *polyglossia* package [3] had not yet come out, and *babel* did not yet support the Unicode  $\text{T}_{\text{E}}\text{X}$  engines.

countering ‘ $\backslash\text{begin}\{\text{document}\}$ ’, that is, at the end of the document’s preamble. When we designed the second version [10], we needed to get the encoding used through a document. To do that we proceeded in an analogous way. In other words, we had already created a kind of *mini- $\text{T}_{\text{E}}\text{X}$*  parser, possibly recursive.

### 2.2 Apotheosis

In December 2020, we became the new editor of the *Cahiers GUTenberg*, the journal of the French-speaking  $\text{T}_{\text{E}}\text{X}$  user group.<sup>16</sup> For many reasons, we decided to revise the class used for this journal and discovered that the previous version was used to build other files, such as metadata for Web search engines. On another point, we also decided to automate as many tasks as possible. For example, we plan to extract the information about the title, author(s), and pages from each article’s source file, in order to build the table of contents of an issue. In addition, we wished to check the succession of page numbers for successive articles.

We did not implement the production of metadata from issues of *Cahiers GUTenberg*. But we adapted our mini-parser into a library customisable as shown in §1.2 and we succeeded in generating automatically the table of contents of [1], although several engines were used for separate articles.

## 3 Discussion

Coupling engines based on  $\text{T}_{\text{E}}\text{X}$ ’s kernel with a more modern programming language has shown increased interest for more than a decade. The best-known example is  $\text{Lua}\text{T}_{\text{E}}\text{X}$  [7], where the engine can call procedures written using the Lua language [12], other experiments connect  $\text{T}_{\text{E}}\text{X}$  with Python [16]; applications based on such a *modus operandi* can be found in [17, 24].

Using functions written using the Lua programming language — as allowed by  $\text{Lua}\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  — for the tasks described in §2.2 was impossible: some articles of [1] needed  $\text{pdf}\text{T}_{\text{E}}\text{X}$  or  $\text{X}\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ , and compiling them with  $\text{Lua}\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  crashed. Besides, we confess that we were not disappointed. Extracting metadata from a source text is not tightly tied to typesetting texts — so it should work regardless of the engine used — and should be performed by a separate program.

An alternative could be given by the use of *regular expressions*<sup>17</sup> for most cases. However, let

<sup>16</sup> *GUTenberg* : **G**roupe francophone des **U**tilisateurs de **T** $\text{E}$ **X**.

<sup>17</sup> Interested readers can consult [5] for a good introduction to this field.

us notice that  $\text{T}_{\text{E}}\text{X}$ 's conditional and iterative expressions are not balanced as in modern programming languages, as we showed in [11]. So we are not sure that difficult matching cases can be reasonably handled by regular expressions, which are ‘naturally’ static. In addition, let us recall that our functions resulting from constructs performed by the `g-mk-tex-parsing-f` macro work in one pass, which seems to us to be more efficient than using several regular expressions.

In practice, we have applied such Scheme functions to examples in  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ , or close to this format, that is,  $\text{X}_{\text{Y}}\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  or  $\text{LuaL}^{\text{A}}\text{T}_{\text{E}}\text{X}$ . We think we could build functions able to parse plain  $\text{T}_{\text{E}}\text{X}$  or  $\text{ConT}_{\text{E}}\text{Xt}$  documents and extract suitable information from them, in which case the `g-retain-match` function will be used more intensively.

#### 4 Conclusion

Our contribution consists in a bridge between  $\text{T}_{\text{E}}\text{X}$  and more ‘classical’ programming. More experience will be needed in order to evaluate the relevance of our method. We can be told that using our tool requires mastering Scheme. But there is a price to pay for interesting applications outside typesetting texts. In other words, this program is not intended for end-users who just typeset texts. But we think that our tool may be enjoyed by  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  users who can program. Finally, we can observe that simple requirements can be put into action easily, as shown for getting an article’s title.

#### Acknowledgments

I thank Denis Bitouzé for his impressions about a first version of this article. I am also grateful to the very efficient proofreaders of *TUGboat*: Karl Berry and Barbara Beeton.

#### A How $(\text{L}^{\text{A}})\text{T}_{\text{E}}\text{X}$ files are parsed

You can discover the behaviour of the Scheme functions generated by the `g-mk-tex-parsing-f` macro by choosing some commands judiciously and associating them with functions that *trace* their arguments. Hereafter we give broad outlines of the complete process. Let us recall that a token recognised by  $\text{T}_{\text{E}}\text{X}$  may be a command name, a begin or end of a group, or a single character. Some groups of characters can be processed globally, e.g., two or more consecutive occurrences of end-of-line characters, equivalent to the `\par` command.

Our parser processes such tokens in turn. If a command is associated with a Scheme function, its arguments are parsed recursively, either by using the information about the argument number provided

by the `g-retain-command` function, or by processing the pattern introduced by the `g-retain-match` function. As soon as these arguments are built, the associated Scheme function is applied to these corresponding arguments. Getting such arguments causes tokens to be processed, so commands located with these arguments will be processed according to a kind of call by value.<sup>18</sup> So if we consider the following example:

```
An \emph{emph'd \emph{internal} text}.
```

if all the occurrences of the `\emph` command are to be processed, the ‘... $\bullet$ ’ occurrence will be processed first, then the ‘... $\circ$ ’ occurrence will be processed, according to a leftmost-innermost strategy. Of course, as soon as a Scheme function associated with a command is executed and returns its result, the process of exploring successive tokens in turn is resumed.

#### References

- [1] ASSOCIATION GUTENBERG : *Ils sont de retour!*, Vol. 58 de *Cahiers GUTenberg*. Septembre 2021. <https://www.gutenberg-asso.fr/-Cahiers-GUTenberg->
- [2] Pieter BELMANS: *T<sub>E</sub>X is Turing-Complete*. December 2010. Universiteit Antwerpen, <https://pbelmans.files.wordpress.com/2010/12/textalk.pdf>
- [3] François CHARETTE, Arthur REUTENAUER, Bastien BOUCARIÈS and Jürgen SPITZMÜLLER: *polyglossia: Modern Multilingual Typesetting With X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X and LuaL<sup>A</sup>T<sub>E</sub>X*. 18 July 2022. <https://ctan.org/pkg/polyglossia>
- [4] William F. CLOCKSIN and Christopher S. MELLISH: *Programming in Prolog*. 5th edition. Springer-Verlag. 2003.
- [5] Jeffrey E. F. FRIELD: *Mastering Regular Expressions*. 3rd edition. O’Reilly. August 2006.
- [6] George GRÄTZER: *Universal Algebra*. 2nd edition. Springer-Verlag. 1979.
- [7] Hans HAGEN: “Lua $\text{T}_{\text{E}}\text{X}$ : Howling to the Moon”. *Biuletyn Polskiej Grupy Użytkowników Systemu T<sub>E</sub>X*, vol. 23, pp. 63–68. April 2006. Also published in *TUGboat* vol. 26, no. 2, pp. 152–157. <https://tug.org/TUGboat/tb26-2/hagen.pdf>
- [8] Paul Richard HALMOS: *Naive Set Theory*. Undergraduate Texts in Mathematics. Springer-Verlag. 1987.
- [9] Jean-Michel HUFFLEN: “MIBIB $\text{T}_{\text{E}}\text{X}$ ’s Version 1.3”. *TUGboat*, vol. 24, no. 2, pp. 249–262. July 2003. <https://tug.org/TUGboat/tb24-2/tb77hufflen.pdf>

<sup>18</sup> If such commands are to be processed. Let us recall that we can restrict our process to work at the top level for a precise number of occurrences.

- [10] Jean-Michel HUFFLEN: “MIBIB $\TeX$  Now Deals with Unicode”. In: Tomasz PRZECHLEWSKI, Karl BERRY and Jerzy B. LUDWICHOWSKI, eds., *Premises, Predilections, Predictions. Proc. TUG@Bacho $\TeX$  2017*, pp. 39–41. April 2017. Also published in *TUGboat* vol. 38, no. 2, pp. 245–248. <https://tug.org/TUGboat/tb38-2/tb119hufflen-mlbibtex.pdf>
- [11] Jean-Michel HUFFLEN: “Which Success for  $\TeX$  as an Old Program?”. *Ars $\TeX$ nica*, vol. 30, pp. 24–30. In Proc. GUIT 2020 meeting. October 2020.
- [12] Roberto IERUSALIMSKY: *Programming in Lua*. 2nd edition. Lua.org. March 2006.
- [13] Brian W. KERNIGHAN and Dennis M. RITCHIE: *The C Programming Language*. 2nd edition. Prentice Hall. 1988.
- [14] Donald Ervin KNUTH: *Computers & Typesetting. Vol. A: The  $\TeX$ book*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1986.
- [15] Leslie LAMPOR: *L $\TeX$ : A Document Preparation System. User’s Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.
- [16] Mark LUTZ: *Programming Python*. O’Reilly & Associates. October 1996.
- [17] Henri MENKE: “Parsing Complex Data Formats in Lua $\TeX$  with LPEG”. *TUGboat*, vol. 40, no. 2, pp. 129–135. In Proc. TUG. 2019. <https://tug.org/TUGboat/tb40-2/tb125menke-lpeg.pdf>
- [18] Frank MITTELBACH and Michel GOOSSENS, with Johannes BRAAMS, David CARLISLE, Chris A. ROWLEY, Christine DETIG and Joachim SCHROD: *The L $\TeX$  Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.
- [19] Walter MOREIRA: *A Turing Machine in  $\TeX$* . April 2004. Montevideo, Uruguay. <http://www.cmat.edu.uy/%7Ewalterm/turing/turing.html#download>
- [20] Lawrence C. PAULSON: *ML for the Working Programmer*. 2nd edition. Cambridge University Press. 1996.
- [21] Simon PEYTON JONES, ed.: *Haskell 98 Language and Libraries. The Revised Report*. Cambridge University Press. April 2003.
- [22] Alex SHINN, John COWAN, and Arthur A. GLECKLER, with Steven GANZ, Aaron W. Hsu, Bradley LUCIER, Emmanuel MEDERNACH, Alexey RADUL, Jeffrey T. READ, David RUSH, Benjamin L. RUSSEL, Olin SHIVERS, Alaric SNELL-PYM and Gerald Jay SUSSMAN: *Revised<sup>7</sup> Report on the Algorithmic Language Scheme*. 6 July 2013. <https://small.r7rs.org/attachment/r7rs.pdf>
- [23] George SPRINGER and Daniel P. FRIEDMAN: *Scheme and the Art of Programming*. The MIT Press, McGraw-Hill Book Company. 1989.
- [24] Uwe ZIEGENHAGEN: “Combining L $\TeX$  with Python”. *TUGboat*, vol. 40, no. 2, pp. 126–128. In Proc. TUG 2019. <https://tug.org/TUGboat/tb40-2/tb125ziegenhagen-python.pdf>

◇ Jean-Michel HUFFLEN  
 FEMTO-ST (UMR CNRS 6174)  
 & University of Bourgogne  
 Franche-Comté  
 16, route de Gray  
 25030 BESANÇON CEDEX  
 France  
 jmhuffle (at) femto-st dot fr

## **L<sup>A</sup>T<sub>E</sub>X profiling of author submissions — completeness & usability checking**

Apu V, Rishi T, Aravind Rajendran

### **Abstract**

Due to the permissive nature of L<sup>A</sup>T<sub>E</sub>X, authors who prepare their manuscripts in L<sup>A</sup>T<sub>E</sub>X for publishing their research articles in academic journals often knowingly or unknowingly indulge in non-standard markup practices. Since article submission systems of most publishers are primarily designed for Microsoft Word-based articles, problems in L<sup>A</sup>T<sub>E</sub>X manuscripts go undetected during submission and review processes, and later cause avoidable delays and hardships in processing their submissions.

A tool for pre-submission check followed by requests to fix as much as possible at their end before submission will thus have benefits of earlier publication and reducing turnaround time considerably. T<sub>E</sub>XFolio Alpha is such a web-based tool for pre-submission profiling, completeness and usability checking of L<sup>A</sup>T<sub>E</sub>X manuscript submissions.

### **1 Introduction**

L<sup>A</sup>T<sub>E</sub>X is not just a markup language for document preparation. The programmable nature of L<sup>A</sup>T<sub>E</sub>X allows authors to customize every aspect of the document. When typesetting documents for personal use, this flexibility of L<sup>A</sup>T<sub>E</sub>X is a great advantage. But in the case of academic publishing, journal publishers prefer author-submitted documents to follow a common template and style so that the article publication process, which involves reviewing, editing, typesetting, proofing and final online and print deliverables, can be done within the turnaround time, and as cost effectively as possible.

Journal publishers often provide authors document templates in the preferred layouts and style for both L<sup>A</sup>T<sub>E</sub>X and Microsoft Word. Word being a WYSIWYG document preparation system with limited customization capabilities compared to L<sup>A</sup>T<sub>E</sub>X, manuscript processing at the typesetters' end is easier for a non-math-intensive document prepared using Word templates. Processing of L<sup>A</sup>T<sub>E</sub>X manuscripts depends on many factors, such as the author's expertise and coding style in L<sup>A</sup>T<sub>E</sub>X, the journal submission system's compatibility with L<sup>A</sup>T<sub>E</sub>X, the journal typesetter's expertise in handling L<sup>A</sup>T<sub>E</sub>X manuscripts, etc. Novice L<sup>A</sup>T<sub>E</sub>X authors may ignore compilation errors or may not follow instructions given in L<sup>A</sup>T<sub>E</sub>X templates, while authors expert in L<sup>A</sup>T<sub>E</sub>X may use fancy or cutting-edge packages that may not work with the submission system or that break journal house style.

This results in much communication between the author and typesetter which can cause avoidable delays and difficulties in processing authors' submission within a normal turnaround time. A pre-submission check tool that will ensure the requirements for fast article publishing are met would be a solution for these problems.

There are a number of reasons why a pre-submission profiling tool for L<sup>A</sup>T<sub>E</sub>X manuscripts is needed. First, it can help to ensure that the manuscript is properly formatted and meets all of the submission requirements by providing the facility to edit and compile the manuscript. Second, it can help to identify any potential problems with the manuscript before it is submitted by walking the authors through a checklist of the problems detected and providing instructions to fix them. Finally, it can help to save time and effort in the article production process by allowing the author to fix any problems before the submission process begins.

### **2 L<sup>A</sup>T<sub>E</sub>X submission profiling process**

L<sup>A</sup>T<sub>E</sub>X profiling is the process of analyzing an author's submitted manuscript files for errors, missing files, use of recommended class files and packages, use of unsupported L<sup>A</sup>T<sub>E</sub>X packages, author definitions of macros, use of mandatory items in the submitted manuscript required by the publisher, and then generating a consolidated checklist based on the analysis. The author needs to review the checklist, which reports both the items that are safe to move forward and the items that need action from the author to fix. Thus the profiling process checks the completeness and usability (C&U) of the manuscript.

T<sub>E</sub>XFolio Alpha is a cloud L<sup>A</sup>T<sub>E</sub>X profiling tool which can be used as a web application and a micro-service. Fig. 1 shows a high level block diagram of T<sub>E</sub>XFolio Alpha's workflow as a web application. Instead of directly uploading the manuscript files to the submission system, an author first uploads to T<sub>E</sub>XFolio Alpha. The C&U server on which T<sub>E</sub>XFolio Alpha is running processes the submitted files. This involves checking submitted L<sup>A</sup>T<sub>E</sub>X manuscripts and associated files using T<sub>E</sub>XFolio's analyzing scripts, written in Python and Perl, to detect the class files and packages used in the manuscript, missing input files and figures. If the author is not using the class file preferred by the publisher, T<sub>E</sub>XFolio Alpha will alert the author about the advantages of using that class file. If figures or input files are used in the manuscript but not uploaded to T<sub>E</sub>XFolio Alpha, they will be listed and the author prompted to upload them.

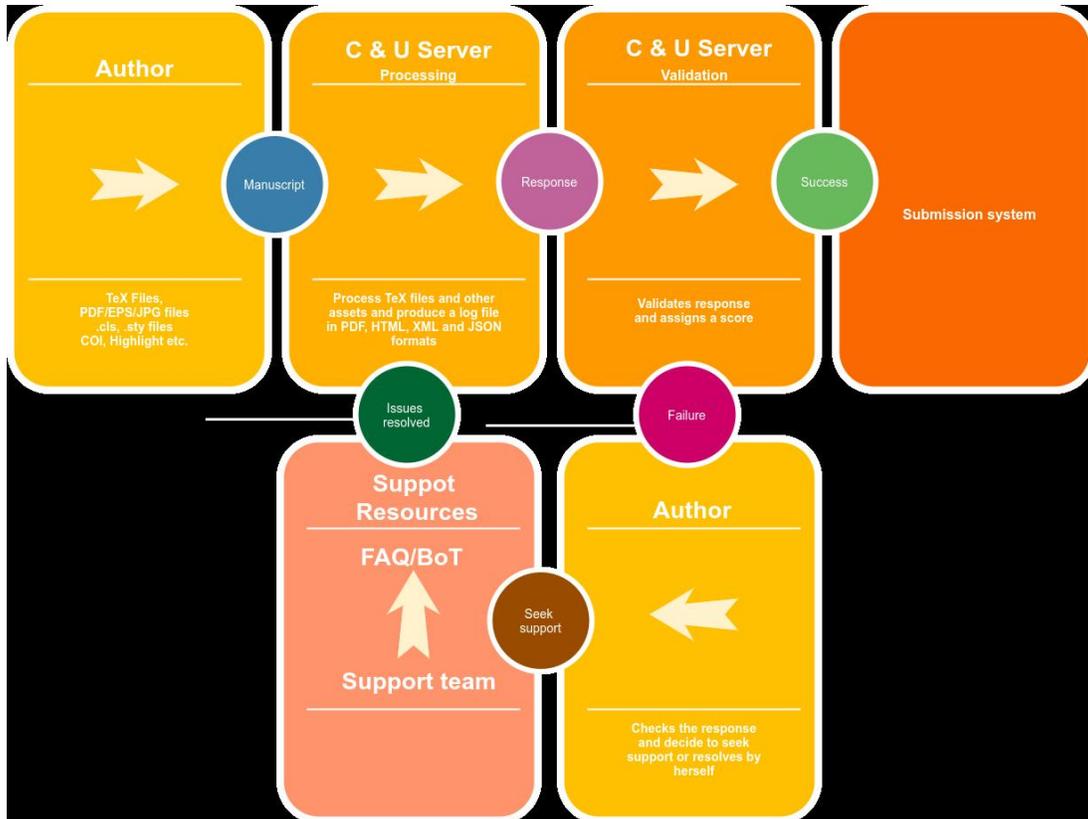


Figure 1: High level block diagram of TeXFolio Alpha.

Once the initial analysis and checking are done, the C&U server compiles the manuscript. If L<sup>A</sup>T<sub>E</sub>X compilation errors are encountered, the author will be prompted with details of the errors and possible

solutions to solve the errors. The author can edit the manuscript in TeXFolio Alpha’s L<sup>A</sup>T<sub>E</sub>X editor (Fig. 2). On the left side of the L<sup>A</sup>T<sub>E</sub>X editor interface there is a file manager. Authors can view the list of files

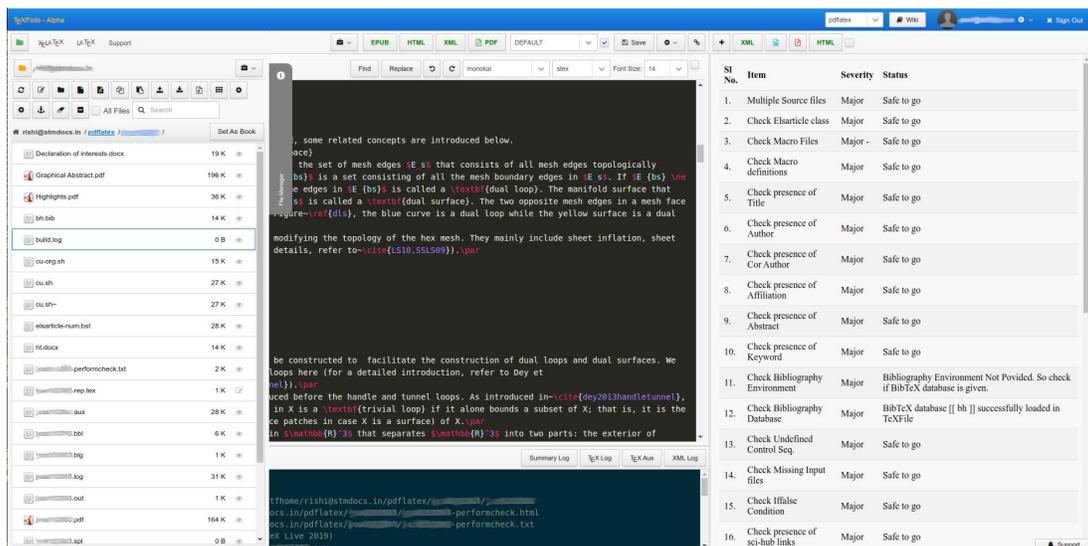


Figure 2: TeXFolio Alpha’s L<sup>A</sup>T<sub>E</sub>X editor.

**Table 1:** List of checks performed by  $\text{\TeX}$ Folio Alpha while profiling  $\text{\LaTeX}$  manuscripts.

No.	Items	Severity
1	Multiple source files	Major*
2	Check usage of Elarticle class	Major
3	Check missing Macro packages	Major
4	Check missing Macro definitions	Major
5	Check presence of Title	Major
6	Check presence of Author	Major
7	Check presence of Corresponding Author	Major
8	Check presence of Affiliation	Major
9	Check presence of Abstract	Minor <sup>†</sup>
10	Check presence of Keyword	Minor
11	Check Bibliography Environment	Minor
12	Check Bibliography Database	Minor
13	Check Undefined References	Minor
14	Check Undefined Control Seq.	Major
15	Check Multiply Def. Labels	Minor
16	Check Missing Input files	Major
17	Check Iffalse Condition	Minor
18	Check Nomenclature	Minor
19	Check Appendix/Supplementary materials	Minor
20	Check possible Overfull content	Minor
21	Check private email address(es)	Minor
22	Check Bibliographic citations — Crossref	Minor
23	Check other cross-references	Minor
24	Check presence of Highlights	Minor
25	Check presence of Conflict of Interest	Minor
26	Check presence of sci-hub links	Major
27	Check presence of notes	Minor
28	Check non-standard Math coding	Minor
29	Check missing Bibliographic citation	Minor
30	Check missing other cross-references	Minor

\* Major: Cannot proceed with submission. Author must correct the issue.

<sup>†</sup> Minor: Can proceed with submission. Typesetter will take care to correct the issue.

submitted and upload, and delete, rename or copy files.  $\text{\TeX}$  files can be opened in the editor in the middle of the interface and changes can be made. Below the editor, the log window will show compilation errors and prompt authors with instructions to fix the errors. On the right-hand side, the document viewer will display the pdf document generated from the manuscript.

Once the author corrects all compilation errors, the C&U server will start validating the manuscript. The document viewer will display the checklist generated by  $\text{\TeX}$ Folio Alpha after the validation process. This checklist is generated by several operations, such as processing the log and auxiliary files using Python and Perl scripts, and a checklist in XML format that

is generated during  $\text{\LaTeX}$  compilation. This checklist can be seen in the document viewer in Fig. 2. A few examples and more details about this XML checklist generation are given in Section 3. Items that are already satisfied by the manuscript will be listed with status ‘Safe to Go’. Items that need action by the author will be listed with details of the problem. Table 1 lists the current checks performed by  $\text{\TeX}$ Folio Alpha.

$\text{\TeX}$ Folio Alpha will also assign a score to the manuscript relating on the status of the items and its severity. If the score does not meet the threshold set by the publisher, the submission will not be moved to the publishers’ submission system; instead, the author will be asked to correct the critical problems.

To help the author solve problems, FAQs, chatbot or human support can be integrated into the T<sub>E</sub>XFolio interface. Since the issues and instructions to solve them are presented in a checklist format it will often be easier for the authors to correct the problems themselves. After the author completes the validation process with a sufficient validation score, the manuscript will be moved to the publisher’s submission system and a preprint pdf will be generated.

T<sub>E</sub>XFolio Alpha can be configured as a micro-service as well, which can be integrated with other applications and used with API calls.

### 3 Role of L<sup>A</sup>T<sub>E</sub>X in the manuscript validation process

As discussed above, T<sub>E</sub>XFolio Alpha uses Python and Perl for processing and analyzing L<sup>A</sup>T<sub>E</sub>X source files, logs and auxiliary files to generate the C&U checklist. Details like missing input files, checking for the use of recommended class files and packages, uncited references etc., can be detected using these methods. But a few critical ingredients for the checklist can only be extracted with the help of L<sup>A</sup>T<sub>E</sub>X during a compilation. T<sub>E</sub>XFolio Alpha uses L<sup>A</sup>T<sub>E</sub>X’s hook mechanisms and the `etoolbox` package’s patching commands to extract checklist details during compilation. All hooks and patching commands are kept in a configuration file and this file is loaded in the manuscript before `\documentclass` using `\input`. The example in this article uses a manuscript using the `elsarticle.cls`; this is a popular document class for preparing preprint pdfs and it is the recommended document class by one of the largest academic publishers.

In this example, `elsarticle-pre-hooks.tex` contains the hook macros and patching commands. This pre-hooks file contains publisher and journal specific configuration for documents generated using `elsarticle.cls`:

```
\input{elsarticle-pre-hooks}
\documentclass[final]{elsarticle}
```

This is the only change in the manuscript made for profiling in T<sub>E</sub>XFolio Alpha. By using hooks and patching there is no need to add any additional packages or macro definition in the preamble part of author manuscript for profiling. This makes sure the author’s manuscript is kept intact during T<sub>E</sub>XFolio Alpha’s profiling process.

We will next discuss three examples where we used L<sup>A</sup>T<sub>E</sub>X hook mechanism in T<sub>E</sub>XFolio Alpha.

#### 3.1 Detection of sensitive macro redefinitions

An author may redefine macros, either intentionally or unintentionally, that are defined in the class file. Although L<sup>A</sup>T<sub>E</sub>X normally reports ‘already defined’ errors, the author may use `\def` or `\renewcommand` to skip this error without considering the reasons for the error. In some cases a package loaded by the author may redefine macros without showing any errors.

We use a hook in the configuration which check for definitions by the author or by loaded packages that override already-defined macros in the class file, per publisher requirements. If such redefinitions go undetected during submission they will raise style error flags later, during journal production at the typesetter, causing delays in the article production as the typesetter must contact the journal and authors to query whether to keep the custom style. If such cases are detected at the submission stage, the author can explain the rationale behind changing the macro, or revert to the original macro defined in the class if the redefinition was unintentional and does not have any particular significance.

Let’s look at the implementation of the hook. We create a `clist` (comma-separated list) that holds the names of sensitive macros. In this example, two commands `\textmarker` and `\author` are the sensitive macros.

```
\clist_new:N \cu_macros_for_verification
\clist_gset:Nn \cu_macros_for_verification
  {textmarker,author}
```

Two property lists are created to hold a hash value of the macro definitions.

```
\prop_new:N \cu_elsarticle_macro_hash
\prop_new:N \cu_document_macro_hash
```

This hash value will be used to verify if the macro definition has been modified.

Using the hook mechanism, we add macro calls after the class file is loaded which compute the hash value of the meaning of the macros listed in `\cu_macros_for_verification` to the property list `\cu_elsarticle_macro_hash`.

```
\AddToHook{class/after}[elsarticle]{
  \clist_map_inline:Nn
    \cu_macros_for_verification
  { \str_if_eq:nnTF { #1 } { }
    { \clist_map_break: }
    { \prop_gput:Nnx
      \cu_elsarticle_macro_hash { #1 }
      { \tex_mdffivesum:D{\csmeaning{#1}} }
    }
  }
}
```

Next, to capture any cases of redefinition of the macros listed in `\g_macros_for_verification` we add an `\enddocument` hook to save hash values for the macro definitions to `\g_document_macro_hash`.

```
\AddToHook{enddocument}{
  \clist_map_inline:Nn
    \cu_macros_for_verification
  { \str_if_eq:nnTF { #1 } { }
    { \clist_map_break: }
    { \prop_gput:Nnx
      \cu_document_macro_hash { #1 }
      { \tex_mdffivesum:D{\csmeaning{#1}}
      }
    }
  }
}
```

Continuing in the same hook, we compare the hash values on both property lists. If hash values are different, authors need to verify if the macro redefinition is necessary.

```
\prop_map_inline:Nn
  \cu_elsarticle_macro_hash {
    \str_if_eq:nnTF { #1 } { }
    { \prop_map_break: }
    { \str_if_eq:eeTF { #2 }
      { \prop_item:Nn
        \cu_document_macro_hash { #1 } }
      { }
    }
  }
```

When we identify such a difference in hash value, we add a checklist item with description of the problem to a custom XML file.

```
{ \iow_now:Nx \cu_report
  {<checklist
    type="redefined-macros">}
  \iow_now:Nx \cu_report
  {<description id="redefined-#1">
    \cumacrodescription{#1}
  </description>}
  \iow_now:Nx \cu_report {</checklist>}
}
}
```

The XML elements are given attributes `@type` and `@id` to simplify generation of a report which can be viewed in the `TEX`Folio Alpha web interface.

### 3.2 Check for mandatory items in the manuscript

Here we look at another case, this time checking for the use of mandatory items in the document. When using the class file suggested by the publisher, some commands defined in the class file, for example, macros for keywords, corresponding author, etc., will

be mandatory for submission. One way to detect if authors have not omitted these commands is to add error messages if they are not used. But in the case of popular class files such as `elsarticle.cls`, authors use it for typesetting documents for personal use like lecture notes too. The implementation of errors specific for publisher requirements will create difficulties in those type of non-academic use cases. So using a hook configuration in the profiling tool is a better solution.

To ensure whether authors use these mandatory commands in manuscripts, a list of such macros are maintained:

```
\clist_new:N \cu_mandatory_macros
\clist_set:Nn \cu_mandatory_macros
  {corref,cortext}
```

`\corref` and `\cortext` are commands used to tag corresponding authors in `elsarticle.cls`.

We iterate through the list, adding a hook to each macro itself which sets a definition to indicate the macro has been used.

```
\clist_map_inline:Nn
  \cu_mandatory_macros
  {\str_if_eq:nnTF { #1 } { }
  { \clist_map_break: }
  { \AddToHook{cmd/#1/before}
    { \csgdef{cu_macro_#1_done}{1}} }
}
```

Then a hook at `\enddocument` verifies if the command has been used, and writes to the checklist XML.

```
\AddToHook{enddocument}{
  \clist_map_inline:Nn
    \cu_mandatory_macros {
      \str_if_eq:nnTF { #1 } { }
      { \clist_map_break: }
      { \ifcsundef { cu_macro_#1_done }
        { \iow_now:Nx \cu_report
          {<checklist
            type="mandatory-macros">}
          \iow_now:Nx \cu_report
          {<description
            id="mandatory-macro-#1">
            \cumandatorymacrodescription{#1}
          </description>}
          \iow_now:Nx \cu_report
          {</checklist>}
        } { }
      }
    }
}
```

An analogous hook can be used to check for use of mandatory environments: instead of hooking

to `cmd/⟨command name⟩/before`, we hook to `env/⟨environment name⟩/before`.

### 3.3 Preventing the use of incompatible or troublesome packages

Packages authors load in their manuscript may be incompatible with the L<sup>A</sup>T<sub>E</sub>X submission system of the publisher. For instance, packages that require the `--shell-escape` option will not be allowed with these online submission systems, as they poses a security risk. A commonly-used example of such a package is `minted.sty`.

Via the `package/⟨package name⟩/before` hook we can stop the compilation at the point where the package is loaded and instruct author to use a similar package that is compatible with the submission system, or suggest some alternate methods. A hook as below can be used to stop compilation at the point where `minted` is detected.

```
\AddToHook{package/minted/before}{
\AddToHook{enddocument}{
  \iow_now:Nx \cu_report
  {<checklist
   type="problem-package">}
  \iow_now:Nx \cu_report
  {<description
   id="problem-pkg-minted">
   \cuproblempkgdescription{minted}
   </description>}
  \iow_now:Nx \cu_report
  {</checklist>}
}
\AddToHook{begindocument/end}
  {\enddocument}
\endinput
}
```

## 4 Objectives and scope of T<sub>E</sub>XFolio Alpha

The main objective of T<sub>E</sub>XFolio Alpha is to ensure that authors submit the latest, single version source material right the first time. The reports generated by the checklists can be used by the publisher and supplier workflow management systems to improve customer experience. FAQs and chatbots can be integrated into the system to walk authors through the process. Being a cloud-based web service or micro-service, authors and publishers can use T<sub>E</sub>XFolio Alpha without the need of a local T<sub>E</sub>X installation.

### References

- [1] L<sup>A</sup>T<sub>E</sub>X's hook management. <https://mirror.ctan.org/macros/latex/base/lthooks-doc.pdf>
- [2] The L<sup>A</sup>T<sub>E</sub>X3 Sources. <https://mirror.ctan.org/macros/latex/contrib/l3kernel/source3.pdf>

- ◇ Apu V  
STM Document Engineering  
Trivandrum, Kerala, India  
[apu.v \(at\) stmdocs.in](mailto:apu.v@stmdocs.in)  
<https://stmdocs.in>
- ◇ Rishi T  
STM Document Engineering  
Trivandrum, Kerala, India  
[rishi \(at\) stmdocs.in](mailto:rishi@stmdocs.in)  
<https://stmdocs.in>
- ◇ Aravind Rajendran  
Independent Consultant,  
STM Document Engineering  
Trivandrum, Kerala, India  
[aravind \(at\) stmdocs.in](mailto:aravind@stmdocs.in)  
<https://stmdocs.in>

# L<sup>A</sup>T<sub>E</sub>X News

Issue 35, June 2022

*Contents*

**Introduction** 1

**Document metadata interface** 1

**The latex-lab bundle** 2

**A new mark mechanism for L<sup>A</sup>T<sub>E</sub>X** 2

**A key/value approach to option handling** 3

**New or improved commands** 3

- Floating point and integer calculations . . . . . 3
- CamelCase commands for changing arguments to csnames . . . . . 3
- Testing for (nearly) empty arguments . . . . . 4
- Better allocator for Lua command ids . . . . . 4
- Starred command version for `\ref`, `\Ref` and `\pageref` . . . . . 4
- Preparation for supporting PDF in backends . . . . . 4

**Code improvements** 4

- `\protected` UTF-8 character definitions . . . . . 4
- A small update to `\obeylines` and `\obeyspaces` . . . . . 4
- `doc` upgraded to version 3 . . . . . 4
- `doc` can now show dates in change log . . . . . 5
- `ltxdoc` gets options `nocfg` and `doc2` . . . . . 5
- Lua<sub>T</sub>E<sub>X</sub> callback improvements . . . . . 5
- Class `proc` supports `twoside` . . . . . 5
- Croatian character support . . . . . 5
- Cleanup of the Unicode declaration interface . . . . . 5
- New hook: `include/excluded` . . . . . 5
- Input support for normalized angle brackets . . . . . 5

**Bug fixes** 5

- Using `\DeclareUnicodeCharacter` with C1 control points . . . . . 5
- Fix `\ShowCommand` when used with `ltxcmd` . . . . . 6
- Make `\cite{}` produce a warning . . . . . 6
- Fix adding `cmd` hooks to simple macros . . . . . 6
- Warn if `shipout/lastpage` hook is executed too early . . . . . 6
- More consistent use of cramped math styles in Lua<sub>T</sub>E<sub>X</sub> . . . . . 6
- Fixed bug when setting hook rules for one-time hooks . . . . . 6

**Changes to packages in the amsmath category** 6

- `amsopn`: Do not reset `\operator@font` . . . . . 6
- `amsmath`: Error in `\shoveleft` . . . . . 6
- `amsmath` and `amsopn`: Robustify user commands . . . . . 6

- Changes to packages in the graphics category** 7

  - Color in formulas . . . . . 7
  - Fix locating files with `\graphicspath` . . . . . 7

- Changes to packages in the tools category** 7

  - `multicol`: Fix `\newcolumn` . . . . . 7
  - `bm`: Fix for `amsmath` operators . . . . . 7

*Introduction*

The 2022 June release of L<sup>A</sup>T<sub>E</sub>X is again focussing on improvements made for our multi-year project to automatically offer tagged PDF output [1]. These are the new document metadata interface, the new mark mechanism for L<sup>A</sup>T<sub>E</sub>X, a standard key/value approach for options, and the introduction of the `latex-lab` area for temporary code that can be optionally loaded by a document (when `\DocumentMetadata` is used with certain test keys). These additions are described in the first sections. Related to this effort there are updates to `hyperref` and `tagpdf`, both of which have their own distributions.

As usual, we also added a number of smaller improvements and bug fixes in various components of core L<sup>A</sup>T<sub>E</sub>X. Perhaps the most interesting ones (for some users) are direct support for floating point arithmetic (via `\fpeval`; see below) and the ability to properly color parts of math formulas without introducing spacing problems. For this we now offer the command `\mathcolor`; see the description near the end of the newsletter. There is also a new major release of the `doc` package that supports a more fine-grained classification of code elements and properly supports `hyperref`.

*Document metadata interface*

Until recently there was no dedicated location to declare settings that affect a document as a whole. Settings had to be placed somewhere in the preamble or as class options or sometimes even as package options. For some such settings this may cause issues, e.g., setting the PDF version is only possible as long as the PDF output file has not yet been opened which can be caused by loading one or the other package.

For the “L<sup>A</sup>T<sub>E</sub>X Tagged PDF project” [1] further metadata about the whole document (and its processing) need to be specified and again this data should be all placed in a single well-defined place.

For this reason we introduce the new command `\DocumentMetadata` to unify all such settings in one place. The command expects a key/value list that describes all document metadata for the current document. It is only allowed to be used at the very

beginning of the document, i.e., the declaration has to be placed *before* `\documentclass` and will issue an error if found later.

At this point in time we provide only the bare command in the format; the actual processing of the key/value is defined externally and the necessary code will be loaded if the command is used. This scheme is chosen for two reasons: by adding the command in the kernel it is available to everybody without the need to load a special package using `\RequirePackage`. The actual processing, though, is external so that we can easily extend the code (e.g., offering additional keys or changing the internal processing) while the above-mentioned project is progressing. Both together allows users to immediately benefit from intermediate results produced as part of the project, as well as offering the L<sup>A</sup>T<sub>E</sub>X Project Team the flexibility to enable such intermediate results (for test purposes or even production use) in-between and independently of regular L<sup>A</sup>T<sub>E</sub>X releases. Over time, tested and approved functionality can then seamlessly move into the kernel at a later stage without any alterations to documents already using it. At the same time, not using the new consolidated interface means that existing documents are in no way affected by the work that is carried out and is in a wider alpha or beta test phase.

Documentation about the new command and already existing keys are in `lmeta` (part of `source2e.pdf`) and `documentmetadata-support.pdf` and also in the documentation of the `pdfmanagement-testphase` package.

Package and class authors can test if a user has used `\DocumentMetadata` with `\IfDocumentMetadataTF`.

### *The latex-lab bundle*

We added a new `latex-laboratory` bundle in which we place new code that is going to be available only through a `\DocumentMetadata` declaration and that is—most importantly—work under development and subject to change without further notice. This means that commands and interfaces provided there may get altered or removed again after some public testing. The code can be accessed through the `\DocumentMetadata` key `testphase`. Currently supported values are `phase-I` and `phase-II` that enable code of the tagged PDF project (phase I is frozen, and phase II is the phase we are currently working on). With

```
\DocumentMetadata{testphase=phase-II}
you currently enable tagging for paragraphs and
footnotes; more document elements will follow soon.
```

For more detailed testing it is also possible to pass other values to `testphase`; for example, the first incarnation of a template design interface based on `l3keys` can be accessed through the value `prototype`, thus

```
\DocumentMetadata
  {testphase={phase-II,prototype}}
```

will enable all of phase-II plus the draft template interface (which is not yet integrated in phase-II).

Eventually, code will move (once considered stable) from the testphase into the L<sup>A</sup>T<sub>E</sub>X kernel itself. Tagging will continue to require a `\DocumentMetadata` declaration, but you will then be able to drop the `testphase` key setting.

### *A new mark mechanism for L<sup>A</sup>T<sub>E</sub>X*

The mark mechanism is T<sub>E</sub>X's way to pass information to the page-building process, which happens asynchronously, in order to communicate relevant data for running headers and footers to the latter, e.g., what is the first section on the page or the last subsection, etc. However, marks may also be used for other purposes. The new kernel module provides a generalized mechanism for marks of independent classes.

The T<sub>E</sub>X engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing `\mark` commands into the source document.

This mechanism works well for simple formats (such as plain T<sub>E</sub>X) whose output routines are only called to generate pages. It fails, however, in L<sup>A</sup>T<sub>E</sub>X (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. When that happens T<sub>E</sub>X's `\topmark` no longer reflects the situation at the top of the next page when that page is finally boxed.

Furthermore, T<sub>E</sub>X only offered a single mark while L<sup>A</sup>T<sub>E</sub>X wanted to keep track of more than one piece of information. For that reason, L<sup>A</sup>T<sub>E</sub>X implemented its own mark mechanism where the marks always contained two parts with their own interfaces: `\markboth` and `\markright` to set marks and `\leftmark` and `\rightmark` to retrieve them.

Unfortunately, this extended mechanism, while supporting scenarios such as chapter/section marks, was far from general. The mark situation at the top of a page (i.e., `\topmark`) remained unusable and the two marks offered were not really independent of each other because `\markboth` (as the name indicates) was always setting both.

The new mechanism now available in L<sup>A</sup>T<sub>E</sub>X starting with the 2022 release overcomes both issues:

- It provides arbitrary many, fully independent named marks, that can be allocated and from that point onwards used.
- It offers access for each such mark to retrieve its top, first, and bottom value separately.
- Furthermore, the mechanism is augmented to give access to marks in different “regions”, which may be other than full pages.

The legacy interfaces, e.g., `\markboth`, are kept. Thus classes and packages making use of them continue to work seamlessly. To make use of the extended possibility a new set of commands for the declaration of mark

classes, setting their values and querying their state (in the output routine) is now available in addition. You find the documentation for the new interfaces together with examples and further notes on the mechanism in the file `lmarks-doc.pdf`. Just call `texdoc lmarks-doc` to display it on your computer.

### A key/value approach to option handling

The classical  $\text{\LaTeX} 2_{\epsilon}$  method for handling options, using `\ProcessOptions`, treats each entry in the list as a string. Many package authors have sought to extend this handling by treating each entry as a key–value pair (keyval) instead. To date, this has required the use of additional packages, for example `kvoptions`.

The  $\text{\LaTeX}$  team have for some time offered the package `l3keys2e` to allow keyvals defined using the L3 programming layer module `l3keys` to act as package options. This ability has now been integrated directly into the kernel. As part of this integration, the syntax for processing keyval options has been refined, such that `\ProcessKeyOptions`

will now automatically pick up the package name as the key *family*, unless explicitly given as an optional argument:

```
\ProcessKeyOptions[family]
```

To support creating key options for this mechanism, the new command `\DeclareKeys` has been added. This works using the same general approach as `l3keys` or `pgfkeys`: each key has one or more *properties* which define its behavior.

Options for packages which use this new approach will not be checked for clashes by the kernel. Instead, each time a `\usepackage` or `\RequirePackage` line is encountered, the list of options given will be passed to `\ProcessKeyOptions`. Options which can only be given the first time a package is loaded can be marked using the property `.usage = load`, and will result in a warning if used in a subsequent package loading line.

Package options defined in this way can also be set within a package using the new command `\SetKeys`, which again takes an optional argument to specify the *family*, plus a mandatory one for the options themselves.

### New or improved commands

#### Floating point and integer calculations

The L3 programming layer offers expandable commands for calculating floating point and integer values, but so far these functions have only been available to programmers, because they require `\ExplSyntaxOn` to be in force. To make them easily available at the document level, the small package `xfp` defined `\fpeval` and `\inteval`.

An example of use could be the following:

```
\LaTeX{} can now compute:
\[\frac{\sin(3.5)}{2} + 2\cdot 10^{-3}
= \fpeval{\sin(3.5)/2 + 2e-3} \]
```

which produces the following output:

$\text{\LaTeX}$  can now compute:

$$\frac{\sin(3.5)}{2} + 2 \cdot 10^{-3} = -0.1733916138448099$$

These two commands have now been moved into the kernel and in addition we also provide `\dimeval` and `\skipeval`. The details of their syntax are described in `usrguide3.pdf`. The command `\fpeval` offers a rich syntax allowing for extensive calculations, whereas the other three commands are essentially thin wrappers for `\numexpr`, `\dimexpr`, and `\glueexpr`—therefore inheriting some syntax peculiarities and limitations in expressiveness.

```
\newcommand\calculateheight[1]{%
\setlength\textheight{\dimeval{\topskip
+ \baselineskip * \inteval{#1-1}}}}
```

The above, for example, calculates the appropriate `\textheight` for a given number of text lines.

([github issue 711](#))

#### CamelCase commands for changing arguments to csnames

It is sometimes helpful to “construct” a command name on the fly rather than providing it as a single `\dots` token. For these kinds of tasks the  $\text{\LaTeX} 3$  programming layer offers a general mechanism (in the form of `\exp_args:N...` and `\cs_generate_variant:Nn`). However, when declaring new document-level commands with `\NewDocumentCommand` or `\NewCommandCopy`, etc. the L3 programming layer may not be active, and even if it is, mixing CamelCase syntax with L3 programming syntax is not really a good approach. We have therefore added the commands `\UseName` and `\ExpandArgs` to assist in such situations, e.g.,

```
\NewDocumentCommand\newcopyedit{m0{red}}
{\newcounter{todo#1}%
\ExpandArgs{c}\NewDocumentCommand{#1}{s m}%
{\stepcounter{todo#1}%
\IfBooleanTF {##1}%
{\todo[color=#2!10]%
{\UseName{thetodo#1}: ##2}}%
{\todo[inline,color=#2!10]%
{\UseName{thetodo#1}: ##2}}%
}%
}
```

which provides a declaration mechanism for copyedit commands, so that `\newcopyedit{FMi}[blue]` then defines `\FMi` (and the necessary counter).

The command `\ExpandArgs` can be useful with the argument `cc` or `Nc` in combination with `\NewCommandCopy` if the old or new command name or both need constructing. Finally, there is `\UseName` which takes its argument and turns it into a command (i.e., a CamelCase version of `\@nameuse` ( $\text{\LaTeX} 2_{\epsilon}$ ) or `\use:c` (L3 programming layer)) which was also used in the example above.

([github issue 735](#))

*Testing for (nearly) empty arguments*

In addition to `\IfNoValueTF` to test if an optional argument was provided or not, there is now also `\IfBlankTF`, which tests if the argument is empty or contains only blanks. Based on the result it selects a true or false code branch. As usual, the variants `\IfBlankT` and `\IfBlankF` are also provided for use when only one branch leads to some action. Further details and examples are given in `usrguide3.pdf`.

*Better allocator for Lua command ids*

In Lua<sub>TeX</sub> we already had the `\newluafunction` macro which allocates a Lua function identifier which can be used to define commands with `\luadef`. But this always required two steps: `\newluafunction` defines the passed control sequence as an integer, which then has to be used to define the actual Lua command with `\luadef`. After that, the integer is no longer needed. This was inconsistent with other allocators. Therefore we added two new allocators `\newluacmd` and `\newexpandableluacmd` which directly define a control sequences invoking the allocated Lua function. The former defines a non-expandable Lua command, the latter an expandable one. Of course, the associated Lua function still has to be defined by assigning a function to the `lua.get_functions_table()` table. The required index is available in `\allocationnumber`.

An example could be

```
\newluacmd \greeting
\directlua {
lua.get_functions_table()
  [tex.count.allocationnumber]
  = function()
    local name = token.scan_argument()
    tex.sprint('Hello ', name, '!')
  end
}

\greeting{world}
```

(*github issue 536*)

*Starred command version for \ref, \Ref and \pageref*

For a long time `hyperref` has provided starred versions for the reference commands which do not create active links. This syntax extension required users and package authors to check if `hyperref` was loaded and adjust the coding accordingly or take the starred forms out if text was copied to a document without `hyperref`. The commands have now been aligned with the `hyperref` usage and always allow an optional star. The `showkeys` package has been updated to handle the starred versions too, both with `hyperref` or `nameref` and without. The commands are defined with `\NewDocumentCommand` and so no longer expand when written to auxiliary files. This reduces the number of compilations needed to resolve references in captions and sectioning commands. The package `ifthen` has been updated to ensure that `\pageref` can still be used inside tests like `\isodd`.

*Preparation for supporting PDF in backends*

At the current point in time, basic support for PDF in backends is not part of L<sup>A</sup>T<sub>E</sub>X core; it is provided by an external package like `hyperref`. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document. To enable class and package authors to support PDF-specific tasks like the creation of link targets without having to test first if `hyperref` has been loaded, dummy versions of the commands `\MakeLinkTarget`, `\LinkTargetOn`, `\LinkTargetOff` and `\NextLinkTarget` are provided.

*Code improvements**\protected UTF-8 character definitions*

The characters defined via `utf8.def` are now defined as `\protected` macros. This makes them safe to use in expansion contexts where the classic `\protect` mechanism is not enabled, notably L3 programming layer `e` and `x` arguments.

Related to this change `\MakeUppercase` and `\MakeLowercase` have been updated to use the Unicode-aware case changing functions `\text_lowercase:n` in place of the T<sub>E</sub>X primitive `\lowercase`. A similar change will be made in the `textcase` package.

Note: for technical reasons these low-level character handling changes will not be rolled back if the format version is rolled back using the `latexrelease` package rollback mechanism. (*github issue 780*)

*A small update to \obeylines and \obeyspaces*

The plain T<sub>E</sub>X versions of `\obeylines` and `\obeyspaces` make `~M` and `␣` active and force them to execute `\par` and `\space`, respectively. Don Knuth makes a remark in the T<sub>E</sub>Xbook that one can then use a trick such as

```
\let\par=\cr \obeylines \halign{...
```

However, redefining `\par` like this may lead to all kinds of problems in L<sup>A</sup>T<sub>E</sub>X. We have therefore changed the commands to use an indirection: the active characters now execute `\obeyedline` and `\obeyedspace`, which in turn do what the hardwired solution did before.

```
This • means • that • it • is • now • possible • to
• achieve • special • effects • in • a • safe • way.
• This • paragraph, • for • example, • was •
produced • by • making • \obeyedspace • generate
• {\␣\textbullet\␣} • and • enabling •
\obeyspaces • within • a • quote • environment.
```

Thus, if you are keen to use the plain T<sub>E</sub>X trick, you need to say `\let\obeyedlines=\cr` now. (*github issue 367*)

*doc upgraded to version 3*

After roughly three decades the `doc` package received a cautious uplift, as already announced at the 2019 TUG conference—changes to `doc` are obviously always done in a leisurely manner.

Given that most documentation is nowadays viewed on screen, `hyperref` support is added and by default enabled (suppress it with option `nohyperref` or alternatively

with `hyperref=false`) so the internal cross-references are properly resolved including those from the index back into the document.

Furthermore, `doc` now has a general mechanism to define additional “doc” elements besides the two `Macro` and `Env` it has known in the past. This enables better documentation because you can now clearly mark different types of objects instead of simply calling them all “macros”. If desired, they can be collected together under a heading in the index so that you have a section just with your document interface commands, or with all parameters, or ...

The code borrows ideas from Didier Verna’s `dox` package (although the document level interface is different) and it makes use of Heiko Oberdiek’s `hypdoc` package, which at some point in the future will be completely integrated, given that its whole purpose is to patch `doc`’s internal commands to make them `hyperref`-aware.

All changes are expected to be upward compatible, but if you run into issues with older documentation using `doc` a simple and quick solution is to load the package as follows: `\usepackage{doc}[=v2]`

#### *doc can now show dates in change log*

Up to now the change log was always sorted by version numbers (ignoring the date that was given in the `\changes` command). It can now be sorted by both version and date if you specify the option `reportchangedates` on package level and in that case the changes are displayed with

*<version> – <date>*

as the heading (instead of just *<version>*), when using `\PrintChanges`. *(github issue 531)*

#### *ltxdoc gets options nocfg and doc2*

The  $\LaTeX$  sources are formatted with the `ltxdoc` class, which supports loading a local config file `ltxdoc.cfg`. In the past the  $\LaTeX$  sources used such a file but it was not distributed. As a result reprocessing the  $\LaTeX$  sources elsewhere showed formatting changes. We now distribute this file which means that it is loaded by default. With the option `nocfg` this can be prevented.

We also added a `doc2` option to the class so that it is possible to run old documentation with `doc` version 2, if necessary.

#### *LuaTeX callback improvements*

The  $\LaTeX$  callbacks `hpack_quality` and `vpack_quality` are now `exclusive` and therefore only allow a single handler. The previous type `list` resulted in incorrect parameters when multiple handlers were set; therefore, this only makes an existing restriction more explicit.

Additionally the return value `true` for `list` callbacks is now handled internally and no longer passed on to the engine. This simplifies the handling of these callbacks and makes it easier to provide consistent interfaces for user-defined `list` callbacks.

#### *Class proc supports twoside*

The document class `proc`, which is a small variation on the `article` class, now supports the `twoside` option, displaying different data in the footer line on recto and verso pages. *(github issue 704)*

#### *Croatian character support*

The default `inputenc` support has been extended to support the 9 characters `DŽ`, `Dž`, `dž`, `LJ`, `Lj`, `lj`, `Nj`, `Nj`, `nj`, input as single UTF-8 code points in the range `U+01C4` to `U+01CC`. *(github issue 723)*

#### *Cleanup of the Unicode declaration interface*

When declaring encoding specific commands for the Unicode (TU) encoding some declarations (e.g., `\DeclareUnicodeComposite`) do not have an explicit argument for the encoding name, but instead use the command `\UnicodeEncodingName` internally. There was one exception though: `\DeclareUnicodeAccent` required an explicit encoding argument. This inconsistency has now been removed and the encoding name is always implicit. To avoid a breaking change for a few packages on CTAN, `\DeclareUnicodeAccent` still accepts three arguments if the second argument is TU or `\UnicodeEncodingName`. Once all packages have been updated this code branch will get removed.

At the same time we added `\DeclareUnicodeCommand` and `\DeclareUnicodeSymbol` for consistency. They also use `\UnicodeEncodingName` internally, instead of requiring an encoding argument as their general purpose counterparts do. *(github issue 253)*

#### *New hook: include/excluded*

A few releases ago we introduced a number of file hooks for different types of files; see [2] and in particular [4]. The hooks for `\include` files now have an addition: if such a file is not included (because `\includeonly` is used and its *<name>* is not listed in the argument) then the hooks `include/excluded` and `include/<name>/excluded` are executed in that order—of course, only if they contain code. This happens after  $\LaTeX$  has loaded the `.aux` file for this include file, i.e., after  $\LaTeX$  has updated its counters to pretend that the file was seen.

#### *Input support for normalized angle brackets*

Source files containing `<` or `>` directly written as Unicode codepoints `U+2329` and `U+232A` no longer break when the source file gets normalized under Unicode normalization rules. *(github issue gh/714)*

#### *Bug fixes*

##### *Using \DeclareUnicodeCharacter with C1 control points*

An error in the UTF-8 handling for non-Unicode  $\TeX$  has prevented `\DeclareUnicodeCharacter` being used with characters in the range hex 80 to 9F. This has been corrected in this release. *(github issue 730)*

*Fix `\ShowCommand` when used with `ltxcmd`*

When `\ShowCommand` support was added for `ltxcmd` in the previous release [3], a blunder in the code made it so that when `\ShowCommand` was used on a command defined with `ltxcmd`, it only printed the meaning of the command in the terminal, but didn't stop for interaction as it does elsewhere (mimicking `\show`). The issue is now fixed. *(github issue 739)*

*Make `\cite{}` produce a warning*

When the `\cite` command can't resolve a citation label it issues a warning "Citation '*label*' on page *page* undefined". However, due to some implementation details a completely empty argument was always silently accepted. Given that there are probably people who write `\cite{}` with the intention to fill in the correct label later it is rather unfortunate if that is not generating a warning that something in the document is still amiss. This has finally been corrected and a warning is now generated also in this case. *(github issue 790)*

*Fix adding cmd hooks to simple macros*

A bug in how  $\LaTeX$  detected the type of a command caused a premature forced expansion of such commands, which, depending on their definition, could be harmless or could cause severe trouble. This has been fixed in the latest release. *(github issue 795)*  
*(https://tex.stackexchange.com/q/637565)*

*Warn if `shipout/lastpage` hook is executed too early*

The hook `shipout/lastpage` is intended to place `\specials` into the last page shipped out. This is needed for some use cases, e.g., tagging. If that hook is nonempty and the user has added additional pages since the last run, then  $\LaTeX$  executes this hook too early, but until now without giving any indication that the document needs rerunning. This has now been corrected and an appropriate warning is given. *(github issue 813)*

*More consistent use of cramped math styles in `LuaTeX`*

Using `LuaTeX`'s `\Udelimiterover` to place a horizontally extensible glyph on top of a mathematical expression now causes the expression to be set in cramped style, as used in similar situations by traditional  $\TeX$  math rendering. Similarly, cramped style is now used for expressions set under such a delimiter using `\Underdelimiter`, but is no longer used when setting an expression on top of such extensible glyphs using `\Overdelimiter`. This new behavior follows  $\TeX$ 's rule that cramped style is used whenever something else appears above the expression. Additionally the math style of these constructs can now be detected using `\mathstyle`.

The old behavior can be restored by adding

```
\mathdefaultsmode=0
```

to a document.

*Fixed bug when setting hook rules for one-time hooks*

If a `\DeclareHookRule` command is set for a one-time hook, it has to come *before* the hook gets used, because otherwise it never applies—after all, the hook is used

only once. There was a bug in the implementation in that the sorting mechanism was still applied if the `\DeclareHookRule` declaration appeared while the one-time hook was executed, causing the spurious typesetting of the code labels and the hook name. This bug is now fixed and an error is raised when a new sorting rule is added to an already-used one-time hook.

A possible scenario in which this new error is raised is the following: package AAA declares a hook rule for `begindocument` (i.e., `\AtBeginDocument`) to sort out the behavior between itself and some other package. Package BBB wants to load package AAA but only if it hasn't been loaded in the preamble, so delays the loading to `begindocument`. In that case the hook rule declared by AAA can no longer be applied and you get the error. If that happens the solution is to load the package in `begindocument/before`, which is executed at the very end of the preamble but before `begindocument` is processed. *(github issue 818)*

*Changes to packages in the `amsmath` category**`amsopn`: Do not reset `\operator@font`*

The package `amsopn` used to define `\operator@font` but this command has been provided by the  $\LaTeX$  format for at least 14 years. As a result the definition in `amsopn` is equivalent to a reset to the kernel definition, which is unnecessary and surprising if you alter the math setup (e.g., by loading a package) and at a later stage add `amsmath`, which then undoes part of your setup. For this reason the definition was taken out and `amsmath/amsopn` now relies on the format definition.

In the unlikely event that you want the resetting to happen, use

```
\makeatletter
\def\operator@font{\mathgroup\symoperators}
\makeatother
```

after loading the package. *(github issue 734)*

*`amsmath`: Error in `\shoveleft`*

If `\shoveleft` started out with the words "plus" or "minus" it was misunderstood as part of a rubber length and led either to an error or was swallowed without trace. By adding a `\relax` this erroneous scanning into the argument of `\shoveleft` is now prevented. *(github issue 714)*

*`amsmath` and `amsopn`: Robustify user commands*

Most user-level commands have been made robust in the  $\LaTeX$  kernel during the last years, but variant definitions in `amsmath` turned them back into fragile beings. We have now made most commands in `amsmath` and `amsopn` robust as well to match the kernel behavior. This also resolves a bug recently discovered in the `mathtools` package, which was due to `\big` not being robust after `amsmath` was loaded. *(github issue 123)*

### Changes to packages in the graphics category

#### Color in formulas

While it is possible to color parts of a formula using `\color` commands the approach is fairly cumbersome. For example, to color a summation sign, but not its limits, you need four `\color` commands and some seemingly unnecessary sets of braces to get coloring and spacing right:

```
\[ X = \color{red} \sum
% without {{ the superscript below is misplaced
      _{{\color{black} i=1}}
% without {{ the \sum is black
      ^{{\color{black} n}}
      \color{black} % without it the x_i is red
      x_i          \]
```

Leaving out any of the `\color` commands or any of the `{\dots}` will give you a wrong result instead of the desired

$$X = \sum_{i=1}^n x_i$$

So even if this is possible, it is not a very practical solution and furthermore there are a number of cases where it is impossible to color a certain part of a formula, for example, an opening symbol such as `\left(` (but not the corresponding `\right)`).

We have therefore added the command `\mathcolor` to the `color` and `xcolor` package, which has the same syntax as `\textcolor`, but is specially designed for use in math and handles sub and superscripts and other aspects correctly and preserves correct spacing. Thus, the above example can now be written as

```
\[ X = \mathcolor{red}{\sum}_{i=1}^n x_i \]
```

This command is *only* allowed in formulas. For details and further examples, see `mathcolor.pdf`.

#### Fix locating files with `\graphicspath`

If a call to `\includegraphics` asked for a file (say, `image`) without extension, and if both `A/image.pdf` and `B/image.tex` existed (both `A/` and `B/` in `\graphicspath`, but neither in a folder searched by `\TeX`), then `A/image.pdf` would not be found, and a “file not found” error would be incorrectly thrown. The issue is now fixed and the graphics file is correctly found.

([github issue 776](https://github.com/TeXmacs/TeXmacs/issues/776))

(<https://tex.stackexchange.com/q/630167>)

### Changes to packages in the tools category

#### `multicol`: Fix `\newcolumn`

The recently added `\newcolumn` didn’t work properly if used in vertical mode, where it behaved like `\columnbreak`, i.e., spreading the column material out instead of running the column short.

(<https://tex.stackexchange.com/q/624940>)

#### `bm`: Fix for `amsmath` operators

An internal command used in the definition of operator commands such as `\sin` in `amsmath` has been guarded in `\bm` to prevent internal syntax errors due to premature expansion. ([github issue 744](https://github.com/TeXmacs/TeXmacs/issues/744))

### References

- [1] Frank Mittelbach and Chris Rowley: *L<sup>A</sup>T<sub>E</sub>X Tagged PDF—A blueprint for a large project*. <https://latex-project.org/publications/indexbyyear/2020/>
- [2] L<sup>A</sup>T<sub>E</sub>X Project Team: *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> news 32*. <https://latex-project.org/news/latex2e-news/1tnews32.pdf>
- [3] L<sup>A</sup>T<sub>E</sub>X Project Team: *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> news 34*. <https://latex-project.org/news/latex2e-news/1tnews34.pdf>
- [4] Frank Mittelbach, Phelype Oleinik, L<sup>A</sup>T<sub>E</sub>X Project Team: *The l<sup>A</sup>T<sub>E</sub>X filehook documentation*. Run `texdoc lATEX-filehook-doc` to view.

## Introductory L<sup>A</sup>T<sub>E</sub>X workshop en français

Éric Guichard, Jean-Michel Huffle

We have greatly enjoyed organizing the first French workshop about L<sup>A</sup>T<sub>E</sub>X for the TUG 2022 conference. It can be viewed online at [youtube.com/watch?v=1UssT1N1cfU](https://youtube.com/watch?v=1UssT1N1cfU); the duration is about three hours.

It was a course for beginners, and we felt that we had to offer a different presentation than those found online or in books, tutorials, etc. All of those are very well made, and it did not seem useful to copy them. For this reason, we decided to begin with a historical and philosophical presentation of writing, to explain, rather than the qualities and easiness of L<sup>A</sup>T<sub>E</sub>X, the invention of Donald Knuth in a wide frame — from Euclid and Aristotle through to Frutiger and digital writing.

Why? Generally, people who write a lot, or who edit books or reviews, split between Word, InDesign or L<sup>A</sup>T<sub>E</sub>X users, and often debate endlessly about the qualities of their preferred tool. But writing is more complicated than what we think. And it seemed interesting to remind viewers that (for instance before the invention of computers and of Word) it is a *technology of the intellect*, as Jack Goody described it ([en.wikipedia.org/wiki/Jack\\_Goody](https://en.wikipedia.org/wiki/Jack_Goody)).

Goody showed that writing is a technology without which we cannot think in a sophisticated way (no Hilbert spaces without writing) and which sometimes formats our thoughts and representations (e.g., law, religions) and of course which is essential in the notion of culture (from our aesthetic tastes till our analysis of Greek philosophers, which we know through written commentaries of the analysts of their analysts).

Astonishingly, this understanding of the link between matter and spirit (technology and thought) is very recent, compared to the 5300 years of history of writing. This fact may explain why we have so many difficulties to understand the contribution of Donald Knuth, who had the same analysis as Goody.

After this presentation, we began to introduce L<sup>A</sup>T<sub>E</sub>X, focusing mostly on beginners in social sciences and humanities. Here is the summary of our presentations:

1. Thinking L<sup>A</sup>T<sub>E</sub>X, thinking with L<sup>A</sup>T<sub>E</sub>X, by E. Guichard.  
After a short story of writing, concepts and advantages of T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X.
2. L<sup>A</sup>T<sub>E</sub>X, first steps for beginners, by E. Guichard.  
How to put a first document into action with L<sup>A</sup>T<sub>E</sub>X; interactive show with simple examples.
3. Questions of typography, by J.-M. Huffle.

We did not attempt to be exhaustive about the points specific to French typography, but just showed that most are now well-handled, by packages such as `babel` and `polyglossia`. We also separated the points which should be addressed last — e.g., improperly hyphenated words — and the precautions we should get used to adopting systematically — e.g., signalling unbreakable space characters.

4. L<sup>A</sup>T<sub>E</sub>X in humanities contexts (writers, designers, publishers), by E. Guichard.

Standards, easiness for reading, dialogues with publishers, design, communication with other typesetting systems.

5. Bibliographies, by J.-M. Huffle.

First we show that generating ‘References’ sections manually is error-prone and leads to bibliographies that are difficult to reuse, because there are far too many possible layouts. Then we explain why several passes are needed when a bibliography processor is used in conjunction with L<sup>A</sup>T<sub>E</sub>X. This demonstration uses BIB<sub>T</sub>E<sub>X</sub> but is suitable for any other bibliography processor. After an example of *cross-references* among bibliographical entries, additional demonstrations aim to illustrate the expressive power of advanced bibliography styles used in conjunction with L<sup>A</sup>T<sub>E</sub>X packages such as `natbib` and `jurabib`. This part ends with an introduction to the `biblatex` package and the `biber` bibliography processor.

### Rappel du programme en français

1. Penser L<sup>A</sup>T<sub>E</sub>X, penser avec L<sup>A</sup>T<sub>E</sub>X (EG) — Thèmes : histoire de l’écriture, concepts et apports de T<sub>E</sub>X et L<sup>A</sup>T<sub>E</sub>X.
2. L<sup>A</sup>T<sub>E</sub>X, premiers pas (EG) — Thèmes : réaliser un premier document en L<sup>A</sup>T<sub>E</sub>X; présentation illustrée d’exemples simples.
3. Points de typographie (JMH) — Thèmes : généralités, coupures, polices, langues.
4. L<sup>A</sup>T<sub>E</sub>X en milieu littéraire (EG) — Thèmes : normes, confort de lecture, *design*, dialogue avec les éditeurs, communication avec d’autres systèmes éditoriaux.
5. Bibliographies (JMH) — Thèmes : processeurs de bibliographies, styles de base, exemples.

- ◇ Éric Guichard  
Eric.Guichard (at) ens-lyon dot fr
- ◇ Jean-Michel Huffle  
jmhuffle (at) femto-st dot fr

---

## Self-publishing, L<sup>A</sup>T<sub>E</sub>X, and Markdown

Lloyd Prentice

### Abstract

Considering L<sup>A</sup>T<sub>E</sub>X plus markdown for serious self-publishers — possibilities and challenges. Crying need, promise on the horizon, but much work to do.

### 1 introduction to self-publishing

Step into your virtual time machine. Zip back to the much-storied age of expatriate Left Bank writers in Paris. Meet and greet the artisans, craftspeople, and marketing specialists toiling to bring *Ulysses*, say, to the reading public. Drop into the typesetting shop churning out galleys. Imagine the sound of clanking brass matrices; smell of molten lead.

Now, flit forward. 2019. Bowkers tells us that 1.68 million ISBNs were issued to U.S. self-publishers in 2018, up 40 percent from the year before [1]. Wordsrated tells us that as of 2022, more than 300 million self-published books are sold each year [2].

Here's the thing. A self-publisher is an army of one striving to publish and market books single-handedly, that is, to functionally replicate the Shakespeare and Company workflow that brought us *Ulysses* and so many other wonderful books of the time.

Writing a book is challenge enough. But self-publishers soldier on under a slew of hats — author, development editor, copy editor, book designer, typesetter, proofreader, publicist, book sales rep, book-seller, . . .

A remarkable number of indie writers manage to launch and market their books within constraints of shoestring budgets with zero line-items for freelance consulting help.

For the intrepid self-publisher it's safe to say: *Time is her most precious asset*.

The personal computer and the Internet are indispensable tools that make self-publishing possible. For the vast majority of self-publishers, I'd venture, it's the writing that counts. Knowing which GUI icons to click to produce the PDF and HTML files they send off to KDP is all they care to know about the black magic of bringing their books to folks they hope will buy and read.

But here's the downside — all too many poorly designed and formatted books that pop up on Amazon and elsewhere give self-publishing a bad rep on the street.

Muse away — 300 million self-published books sold each year. That's a staggering number of titles eagerly seeking readers willing to shell out the cover price.

Lloyd Prentice

doi.org/10.47397/tb/43-2/tb134prentice-selfpub

How many book sales lost, one wonders, due to amateurish book design and typography? How many hopes and dreams crushed?

### 2 In my humble experience

I'm a self-publisher. Life-long book lover. Writing is the core of my checkered career in magazine publishing, corporate communication, academia, and software development. I'm a sloppily organized generalist striving at publishing tasks that demand meticulous attention to minute detail.

At this point I've self-published two novels and a technical programming book — all available on Amazon, the programming book also available on Leanpub. I have more books in various stages of development; fear that there's not enough time left in my life to bring them all to print and screen. *Time is my most precious asset*.

This, then, brings me to L<sup>A</sup>T<sub>E</sub>X.

I typeset my novels with L<sup>A</sup>T<sub>E</sub>X. Found L<sup>A</sup>T<sub>E</sub>X comfortable and efficient. Novels turned out well enough with default L<sup>A</sup>T<sub>E</sub>X settings. I set out to write the programming book in L<sup>A</sup>T<sub>E</sub>X. But code boxes looked atrocious. L<sup>A</sup>T<sub>E</sub>X consultants Amy Henderson and Kathryn Hargreaves bailed me out.

I needed to stand on my own two feet so I dove cold turkey into L<sup>A</sup>T<sub>E</sub>X — much appreciated the creative potential for book styling, but learned quickly that tedious text markup was not congenial with my clumsy fingers and skip-about mind.

In all likelihood I would have pushed L<sup>A</sup>T<sub>E</sub>X to not now, maybe not ever.

The discovery of Vít Novotný's L<sup>A</sup>T<sub>E</sub>X markdown package was lightning out of the blue. I knew markdown from software development experience — loved the simplicity and readability.

Markdown! I can have my cake and eat it too.

So I set out to write a book about L<sup>A</sup>T<sub>E</sub>X markdown for self-publishers.

### 3 Markdown

Now, more than a year of hard work later, the book is 95% ready for launch — nonfiction totally composed on a plain-vanilla text editor, marked up for styling with markdown and, with generous help from Vít and Tereza Vrabcová, styled with L<sup>A</sup>T<sub>E</sub>X. But, I decided to defer publication. Why? I don't understand the templates. On me and my inexperience, yes. But the templates feel like a tech-stretch too far for tech-wary self-publishers.

Here's what I've learned plus a few no-doubt naive thoughts on how L<sup>A</sup>T<sub>E</sub>X and markdown can inspire self-publishers to publish more beautiful books:

- Markdown imposes minuscule cognitive friction. The author can efficiently mark up work for styling simultaneously with creative composition. A big productivity win.
- The markdown package is still a work-in-progress, but quite productive for simply-styled works of fiction and nonfiction.
- There’s a disconnect, however, between the productive efficiency of the current version of markdown and the book styling potential of  $(\LaTeX)$ .
- The  $(\LaTeX)$  ecosystem is arguably too vast and daunting for the average self-publisher, but with will and work this can be overcome with benefits to all. . .

#### 4 Will? Work?

Who has time for that?

Actually, it’s a matter of bringing work that’s already business as usual across the broad  $\TeX$  community into tighter focus. It starts, I believe, with a question: “What can we do to make  $(\LaTeX)$ -based workflow for book publishers as efficient and productive as possible?”

Note that I’ve broadened the constituency beyond self-publishers to include indie and traditional publishers as well. Here, however, I’m mainly thinking self-publishers.

There are opportunities here for many across the greater  $(\LaTeX)$  community to contribute experience, insights, technical know-how, and sound pedagogy.

So let’s don our virtual reality glasses to consider the competencies and needs of Cindy, Marcos, and Sophia — fictional self-publishers all.

Cindy is a talented 22-year-old single mother striving to earn more by self-publishing romance novels. Her PC runs Windows and Word. She brings several years of social media engagement to her self-publishing venture.

Marcos is a 36-year-old software engineer. He spends his working day developing code on a Ubuntu Linux workstation. His ambition is to write the definitive work on Haskell for game programmers.

Sophia is a tenured astrophysicist with considerable  $\LaTeX$  competencies hard-earned while writing her thesis. She needs an up-to-the-minute textbook for her sparsely attended graduate seminars. Traditional textbook publishers across the board tell her that the potential market is too niche to touch.

Our core question now boils down to this: “How can we help Cindy, Marcos, and Sophia publish commercial-quality books?”

I see three areas worthy of thought and attention: outreach, tools, documentation.

#### 5 Outreach

All three of our self-publishing avatars need to understand:

- the principles and benefits of competent book design and readable typography;
- that  $\LaTeX$  offers a superior typesetting option;
- how to install a  $\TeX$  distribution, likely  $\TeX$  Live, Mac $\TeX$ , or MiK $\TeX$ ; compose with markdown; and style with  $\LaTeX$ .

It seems evident that, given the vast population of self-publishers, the Internet and social media are the only wheels in town. We need crisp, clear how-to blogs and tutorials that invite self-publishers to experiment with markdown and  $(\LaTeX)$  and guide them to quick and dramatic success.

#### 6 Tools

Two or perhaps all three of our avatars need to install  $\TeX$  Live. Indeed, they need to know how to update  $\TeX$  Live every year to keep abreast of technical and security concerns. And this, in my experience, is a challenging obstacle.

In effort to serve every popular operating system and every niche in the  $\LaTeX$  ecosystem,  $\TeX$  Live installation docs require meticulously close reading and confident computer skills. Contrast with MS Word, which comes batteries-included with the PC, and Ubuntu Linux which installs software with a single command “`sudo apt install myprogram`”.

A second concern is that the full  $\TeX$  Live system installs several GB of files on a poor author’s hard drive, including all too many packages that she’ll never in her lifetime use.

Turns out that there’s a promising mitigation, that is,  $\TeX$  Live schemes [3].

##### 6.1 Schemes

Schemes are subsets of the full collection of  $\TeX$  Live package offerings.

Karl Berry has taken the first step by creating a  $\TeX$  Live scheme specifically dedicated to book design, styling, and publishing (`scheme-bookpub`). The goal: significantly reduce installed hard disk footprint (180 MB), pave the way toward more intuitive installation, and provide the foundation for a welcoming and friendly  $\LaTeX$  ecosystem for self-publishers.

Peter Flynn and Vít Novotný have contributed package ideas. *TUGboat* readers can contribute by reviewing the selection of packages and adding their own must-have packages related to book publishing.

## 6.2 Dedicated website

But we need more to welcome in self-publishers. So I further contend that we need a website under TUG auspices explicitly devoted to (L<sup>A</sup>)T<sub>E</sub>X in book publishing. The website must be simple, attractive, and engaging. It should provide dead-simple, clear, and concise T<sub>E</sub>X Live book scheme download instructions as well as links to tutorials and resources.

We have a domain: `texlibro.net`; work is in progress. Our hope is to bring up a beta site by the end of October.

## 6.3 The L<sup>A</sup>T<sub>E</sub>X markdown package

The L<sup>A</sup>T<sub>E</sub>X markdown package in my view is worthy of center stage in book publishing workflow. But first, we need an elegant solution to a major disconnect.

A beautiful book harmonizes three elements — content, structural design, and page layout. Markdown facilitates composition of content. L<sup>A</sup>T<sub>E</sub>X facilitates structural design and, arguably, page layout. But in my experience there is a missing link — clean and elegant harmonization between these two realms.

HTML incarnations of markdown seamlessly interface with CSS — a standardized and extensively documented digital page styling language. Is such possible for (L<sup>A</sup>)T<sub>E</sub>X plus markdown?

Vít Novotný has proposed several technical fixes to bridge the gap:

- Markdown options
- Referrers
- Referrer prototypes
- Themes
- Snippets

To my limited understanding they each show promise in their own way. But can I integrate them into my workflow? To do so, it seems, I need to have deep experience with T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, markdown intricacies and, arguably, Lua. I hardly know where to begin; I do know that all too many self-publishers would throw up their hands.

Naive thought: HTML has a simple bridge to CSS that cleanly separates content from structure and style. CSS, in turn, is a fairly easy to learn styling language. So, question, is a similar innovation possible for L<sup>A</sup>T<sub>E</sub>X markdown and L<sup>A</sup>T<sub>E</sub>X?

This brings me to documentation.

## 7 Documentation

The T<sub>E</sub>X ecosystem is extensively documented in books, websites, tutorials, and forum snippets.

But in my experience it's overwhelming. Simple formatting question? There's no end of outstanding info and guidance out there. But where to start?

How to separate sheep from goats from inside-T<sub>E</sub>X-baseball? You've no doubt been there, and know what I mean.

As noted earlier, self-publishers care about their words and how to publish their works as efficiently as possible. They need concise and well-structured how-to tutorials with recipes they can follow along and implement to experience eye-opening success.

We need models for such tutorials, many tutorials to cover the book design and formatting challenges of fiction, nonfiction, and academic books, and a well-organized, curated, portal to this font of wisdom, experience, and how-to lore. The website proposed above would be the perfect portal.

Indeed, we need more — a few good books that skip over history and the inside-T<sub>E</sub>X-baseball to inspire the imagination and creative energies of Cindy, Marcos, Sophia and the countless other self-publishers struggling with book publishing challenges of their own.

Here's where L<sup>A</sup>T<sub>E</sub>X professionals and corporate publishers can step in — by contributing know-how; sponsoring the work outlined above.

Yes, there's much work to do. But if it enables future literary talents in the tradition of Joyce, Hemingway, and Fitzgerald to bring beautiful books into the world, then it's well worth the effort.

## References

- [1] Number of Self-Published Titles Jumped 40% in 2018. [www.publishersweekly.com/pw/by-topic/industry-news/publisher-news/article/81473-number-of-self-published-titles-jumped-40-in-2018.html](http://www.publishersweekly.com/pw/by-topic/industry-news/publisher-news/article/81473-number-of-self-published-titles-jumped-40-in-2018.html)
- [2] Self-published Books & Author Sales Statistics. [wordrated.com/self-published-book-sales-statistics/](http://wordrated.com/self-published-book-sales-statistics/)
- [3] The T<sub>E</sub>X Live Guide — 2022, “Selecting what is to be installed”. [tug.org/texlive/doc/texlive-en/texlive-en.html#x1-240003.2.2](http://tug.org/texlive/doc/texlive-en/texlive-en.html#x1-240003.2.2)
- [4] Book publication topic on CTAN. [ctan.org/topic/book-pub](http://ctan.org/topic/book-pub)

◇ Lloyd Prentice  
lloyd (at) writersglen dot com  
texlibro.net

## The story of a silly package

Paulo Cereda, Phelype Oleinik

### Abstract

In this article, Paulo and Phelype recollect the untold story of two friends writing a silly package just for the fun of it. The story, however, takes a turn when the T<sub>E</sub>X community decides to embrace silliness. You are invited to gather around to learn about T<sub>E</sub>X, friendship, community, silly walks, and, of course, the air speed velocity of an unladen swallow.

### 1 A silly introduction

The story begins with two friends, Phelype and Paulo, talking about random things, as usual. At a certain point, the conversation shifted to flipbooks. You probably know what a flipbook is — even if the name does not ring a bell at all, you most certainly have seen one in action.

A *flipbook* is a booklet with a series of images that very gradually change from one page to the next, so that when the pages are viewed in quick succession, the images appear to animate by simulating motion or some other change. These images are typically placed at page corners.

What if they implement such concept in T<sub>E</sub>X? Granted, it is not a novel idea, but it is definitely a new use case of how powerful and expressive T<sub>E</sub>X is. And it could bring a lot of fun too!<sup>1</sup>

Of course, Phelype and Paulo were thrilled with the idea of having a T<sub>E</sub>X implementation of a flipbook. It should be simple enough, as they could exploit the document page numbering to achieve the desired visual effect. But they definitely needed way more than just a good plan: they needed an actual series of images to be animated.

### 2 A silly brainstorming

The two friends had lots of ideas, but they all lacked the sort of expressiveness principle Phelype and Paulo were looking for. If this package was supposed to reach an international audience just counting on a series of images, the visual message should be easily identified and understood regardless of language or any other cultural aspect.

They needed something... completely different. And, unsurprisingly, they found it in an emblematic sketch from Monty Python's Flying Circus.

For those unfamiliar with the name, *Monty Python's Flying Circus* was a British sketch comedy

series created by and starring Graham Chapman, John Cleese, Eric Idle, Terry Jones, Michael Palin and Terry Gilliam. They relied on a form of a humour called *surreal*, which is based on deliberate violations of causal reasoning, thus producing events and behaviours that are obviously illogical.

The sketch that inspired the two friends is named *The Ministry of Silly Walks*, first aired on September 15, 1970. The sketch, as originally depicted in the series, begins with John Cleese playing Mr. Teabag, a civil servant who, after purchasing *The Times* from the newsagent in the previous sketch, walks through the streets of London (at the crossing of Thorpebank Road and Dunraven Road) in a very peculiar manner. He eventually arrives at his place of business: *The Ministry of Silly Walks*, on the northern end of Whitehall. In the hallway, he passes other employees all exhibiting their own silly walks before arriving at his office. Once there, he finds Mr. Putey (portrayed by Michael Palin) waiting for him and apologizes for the delay, explaining that his walk has become particularly silly of late and it takes longer for him to reach his destination.

On January 7, the *International Day of Silly Walks* is celebrated around the world by all loyal fans of Monty Python's Flying Circus. In the Czech city of Brno, a *Silly Walk City March* is held annually since 2012.

### 3 A silly implementation

So far they had a great, if not the greatest, idea for a package, but they were still in need of proper assets. Thankfully, the Wikipedia entry for *The Ministry of Silly Walks* had a chart on silly walk gaits with instructions, done by Jazeen Hollings under a Creative Commons CC BY-SA (attribution / share alike) 3.0 license. At least they had something to get started.

Instead of having 12 independent image files — one for each step in the silly walk gait chart — Phelype and Paulo decided to use a sprite sheet instead. *Sprite sheet* is the name of a big image containing several smaller images or icons. It is a technique employed by designers to reduce the number of requests the browser makes to the server — reducing the number of such requests could make a web page load faster. This technique is also used in animation engines.

They ended up with a silly walk stripe with 12 blocks, each block (a 48-pixel square) containing an image corresponding to a step in the silly walk gait chart. The implementation was straightforward: they simply had to map page numbers to their corresponding blocks, in a circular fashion.

<sup>1</sup> Editor's note: A cousin to the flipbooks idea appeared in William Adams' article "There is no end: Omega and Zapfino", *TUGboat* 24:2, pp.183–199, [tug.org/TUGboat/tb24-2/tb77adams.pdf](http://tug.org/TUGboat/tb24-2/tb77adams.pdf).

The package had to have a name, and the choice was clear. Since the package provided John Cleese's iconic silly walk routine as a page numbering style, they named it `sillypage`, of course.

### 3.1 Version 1.0

Version 1.0 was ready! The package documentation was perhaps far too terse, but that did not stop Phelype and Paulo from sending it to CTAN on January 10, 2022.

This version, however, was not fully compliant with CTAN guidelines. To name a few, they missed a top level folder hierarchy, a proper `README` and an actual example. Thankfully, Petra R. was amazing in helping them shape a new version. Her feedback was fantastic!

Besides working on the issues reported by CTAN, Phelype and Paulo enhanced the documentation a bit, included a new macro to provide the silly walk routine step and added an example. Phelype walked the extra mile and made the package fully compliant with the  $\LaTeX$ 3 build system for build and test purposes. Version 1.1 was ready!

### 3.2 Version 1.1

Phelype and Paulo sent this new version to CTAN on the same day of the previous version, January 10. This time, there were no issues reported from CTAN's side. In one or two days, barring any unforeseen events, this little package would hit  $\TeX$  distributions.

In the meantime, they started working on a new version of our package.

### 3.3 Version 1.2

Version 1.2 had improved documentation to better explain the background and implementation details. On January 11, they were ready to send it to CTAN! For whatever reason, Paulo had to postpone the update to the next day, and it proved to be a wise choice.

On January 12, the two friends got their first issue in the package repository. The culprit was a corrupted image file caused by a blip on the server side. They then took this opportunity to send version 1.2 to CTAN instead of repacking the previous one.

This time, the package was successfully deployed. Little did they know, their lives were about to be changed forever.

As Phelype and Paulo had quite a succession of updates in the span of a few days, version 1.2 was used for the official package announcement on January 13. Of course, they had to go silly on the announcement text as well.

Paulo then announced their package via an innocent tweet and decided to tag John Cleese. Needless to say, the  $\LaTeX$  corner of Twitter made it viral. People and companies joined the fun. Even the official TUG account announced our new package and tagged Cleese as well! DANTE and GUTenberg also spread the news.

Given the engagement and excitement around their package, Paulo was actually hoping for any comment or reply from John Cleese himself, the silly walk ultimate authority, at some point, but unfortunately that did not happen yet. Maybe he is not being annoyed sufficiently — more mentions on Twitter would definitely help!

On that same day, Paulo realised the community comments section of their package page on CTAN got a new entry. More importantly, he realised he could add a comment to our own package! Of course, Paulo wrote a very serious text.

A couple of days later, on January 16, samcarter wrote Paulo about a vector-based silly walk routine she just had created. Vector graphics are not composed of pixels as raster images, so they can be scaled to any size without losing quality. It was definitely a quantum leap from their current approach!

Phelype simplified the code to use a multipage PDF file containing the silly walk steps in vector format instead of the original single raster image. Version 1.3 was then sent to CTAN on January 20.

### 3.4 Version 1.3

Paulo opened an issue on that same day about a future improvement to their package. The silly walk animation was repeated over on a circular fashion, which was not ideal when unique page references had to be used, hence the need of support for a proper silly numeral system.

The next day, on January 21, Ulrike Fischer opened an issue with yet another improvement to their package, this time a code update to make it `hyperref`-friendly. Those two newly created issues were actually a good opportunity to work on a new version as soon as possible!

In the meantime, on January 22, CTAN published the package update announcement for version 1.3, which was released two days earlier. Of course, they had another serious text.

Paulo tweeted about it. The feedback was overwhelmingly positive. Apparently, the idea of having your own life-sized John Cleese-like silhouette cardboard in your living room is appealing to a significant number of  $\TeX$  users!

### 3.5 Version 1.4

Back to the package. Phelype rewrote the entire code using the  $\LaTeX$ 3 programming interface and added the silly numeral system and `hyperref` support. On February 2, version 1.4 was sent to CTAN, their latest and greatest version to date.

Incidentally, since samcarter and Ulrike greatly contributed to the package since the beginning, they were invited to join the team and, of course, the blame list.

Since this is a silly package, Phelype and Paulo added a few Monty Python references next to the author names in the documentation. Phelype's name has an explicit reference to a line that keeps popping up in one episode, Paulo's refers to a sketch about letter dictation woes, samcarter's points to the famous cheese shop sketch, and Ulrike's refers to another sketch where someone wants to learn how to fly an aeroplane.

The next day, on February 3, CTAN published the package update announcement for the latest version. Of course, as was by now a tradition, they had another serious text.

Paulo tweeted about it, making sure to tag John Cleese. Again, the feedback was overwhelmingly positive. People really enjoyed our silly package!

## 4 Silly things

Thanks to Phelype, the package was fully  $\LaTeX$ 3-compliant, so they decided to register the prefix. This registration is of course not compulsory but is encouraged. Paulo promptly contacted the  $\LaTeX$  project team and kindly asked them to add the `silly` prefix to the database under their care. Since this was an unusual and fun prefix, the  $\LaTeX$  team had some fun discussing its potential use in the kernel.

A couple of days later, samcarter wrote Paulo telling that we've made into pop culture. The Wikipedia article on *The Ministry of Silly Walks* has a section named *References in popular culture*. Apparently, someone thought their silly package was worth a mention in this section! Quite the achievement!

Frank Mittelbach is working on the third edition of *The  $\LaTeX$  Companion*, so Paulo decided to write to him. Frank agreed it was a good addition to the book and mentioned their package in one of the appendices!

In issue 45 of *La Lettre GUTenberg*, the editors used the silly page numbering on every verso page! Their description of the package is fantastic: *no one knows what this package is for*. What a jewel this issue is!

On June 24, during the summer DANTE meeting, samcarter presented a lightning talk about the silly package! It was a lot of fun, the attendees really liked it!

## 5 Silly interfaces

Using a silly package like this is quite straightforward. For starters, make sure to include it in your document preamble through `\usepackage{sillypage}`. Then write `\pagenumbering{silly}` somewhere in your document body to use our silly page numbering style.

You can also use the `\silly` macro on a  $\LaTeX$  counter to typeset the corresponding image for the value of that counter. It is worth mentioning that this particular macro is applied to counters and not to integers. The `\sillystep` macro, as the name implies, prints the provided step number from the sequence of steps. This macro works exactly like `\silly`, but on integers instead of counters.

For sillier documents, specially articles, theses and books, you can write

```
\pagenumbering{sillynumeral}
```

somewhere in your document to use a silly numeral system, in which each page will be converted to a unique composition of silly steps. Note that this macro differs from its `sillypage` style, as the former is a proper base 12 numeral system whereas the latter simply walks through a 12-cycle silly routine.

Finally, the `\sillynumeral` macro provides the silly numeral system representation from the provided integer value. Again, this is a proper base 12 numeral system. What a silly yet marvellous interface!

## 6 Silly remarks

To impress your supervisor, the authors highly recommend you to use this package in your thesis, print it and ask them to view those pages in quick succession, so the images appear to animate by simulating motion! It is known to work with thesis committees as well.

That's the story of how a silly package brought a smile to the  $\TeX$  community in these difficult times, warmed their hearts and made people have legitimate fun.

- ◇ Paulo Cereda  
cereda (at) duck dot com
- ◇ Phelype Oleinik  
oleinik (at) duck dot com

## Key–value setting handling in the $\LaTeX$ kernel

Joseph Wright,  $\LaTeX$  Project Team

### 1 Introduction

$\LaTeX 2_\epsilon$  introduced the idea of classes and packages, and along with that the concept of class and package options. These are nowadays very familiar to  $\LaTeX$  authors, with the first optional argument to `\documentclass` and `\usepackage` used in the vast majority of  $\LaTeX$  documents:

```
\documentclass[10pt,final]{article}
\usepackage[T1]{fontenc}
\usepackage[numbers]{natbib}
```

This system is a powerful way of controlling behaviours, but is limited as the options are string literals. This is perhaps best exemplified by the font size options: `10pt`, `11pt` and `12pt` in the standard classes. Rather than being dimensions parsed and interpreted, these options are used to load hard-coded configurations for the three nominal sizes.

For the programmer, creating options is easy, requiring only one command for each option, plus a separate one to process the list of options given by the user:

```
\DeclareOption{foo}{...}
\DeclareOption{bar}{...}
\ProcessOptions\relax
```

### 2 Third-party key–value support

It is natural to want to have a more flexible system, and key–value methods are the obvious way to obtain this. As the kernel hasn't to-date offered this, a number of third-party packages have been developed to support programmers in providing these interfaces.

Fundamentally, all of these packages work in the same way. Keys are defined using an existing key–value implementation, ready to be used when the options are examined. A dedicated command is provided to do the latter, and this examines each option recorded by  $\LaTeX$  and tries to match it as a key or a key–value pair. If that is possible, the key–value process is called, while if the option is not known as a key, an unknown key process is used.

Perhaps the most convenient package for providing key–value options to date is `kvoptions`, written by Heiko Oberdiek. Rather than try to provide a full setup for generic key–value work, this package provides a small set of commands which define the most common types of key–value interface with clear names:

```
\RequirePackage{kvoptions}
\DeclareStringOption[me]{name}
\DeclareBoolOption{draft}
\DeclareComplementaryOption{final}{draft}
\DeclareDefaultOption{\ERROR}
\ProcessKeyvalOptions*
```

As we can see, `kvoptions` supports ‘strings’ (saving the tokens given in the input), boolean (switch) options and inverse booleans. It also provides a way to handle unknown options.

Using `kvoptions` or similar approaches has allowed programmers to provide key–value options for a number of years. But there are some downsides. First, ideally one wouldn't need to load a package to do this. It would also be better if there was one mechanism, not several with slightly different syntaxes. More fundamentally, the  $\LaTeX$  kernel carries out space stripping and expansion of options *before* they are passed to packages to examine. This makes handling some option texts awkward.

There is also the issue of option clashes. The  $\LaTeX$  kernel checks that the option list of a package is *identical* if you try to load it twice:

```
\usepackage[option = a]{mypkg}
\usepackage[option = b]{mypkg}
```

This is an issue even without key–value options, but with them it's much worse: it's not possible to know if there is a true clash or if the settings simply can override each other. So there needs to be a way to let packages themselves handle this.

### 3 The new kernel mechanism

In the 2022-06 release of the  $\LaTeX$  kernel, a new built-in approach is available for processing options using key–value methods. This is based on the `expl3` module `l3keys`, which is nowadays built into the kernel. You don't, though, need to know anything about `expl3` to use the new approach: everything is made available under standard  $\LaTeX 2_\epsilon$  names.

As for the classical approach, we need to do three things: create options (keys), define how to deal with unknown options, and process the options. Unlike `\DeclareOption`, the new command `\DeclareKeys` can create multiple options in one go. Each option (key) is created by setting one or more *properties*: these are given after the key name as `.\langle prop \rangle`. The basic properties are `.store`, `.if` and `.code`. These set up keys, respectively, to store the input, to use it to set a switch or to insert arbitrary code. We can also set the `.default` for a key: the value that is assumed if none is given by the user. We will also be adding `.notif` for the Fall 2022 release: that will be the same as `kvoptions`' `\DeclareComplementaryOption`.

```
\DeclareKeys{
  name .store = \mypkg@name ,
  name .default = me ,
  draft .if = mypkg@draft ,
% final .notif = mypkg@draft ,
  demo .code =
    \protected@edef\mypkg@demo{#1}
}
```

To deal with unknown keys, we can declare a dedicated handler: here we simply issue an error.

```
\DeclareUnknownKeyHandler{%
  \PackageError{mypkg}
    {Unknown option "\CurrentOption"}
  \@ehc
}
```

Finally we need to process the options: the command name here is pretty simple.

```
\ProcessKeyOptions
```

This approach will provide the key–value setup we want, and the kernel will automatically use the new approach to repeated loading: there will be no option clash warnings. We might, though, want more control: that can be obtained using the `.usage` property:

```
\DeclareKeys{
  name .store = \mypkg@name ,
  name .default = me ,
  name .usage = load ,
  draft .if = mypkg@draft ,
  draft .usage = preamble ,
  demo .code =
    \protected@edef\mypkg@demo{#1}
}
```

With this, the kernel will automatically issue an error if an option is used in the wrong place: after first loading for load options, outside of the preamble for preamble options.

#### 4 Options are keys

You might have picked up from the above that ‘options’ and ‘keys’ are used almost interchangeably. That’s because, when processing key–value options, they are simply keys that are created before a call to `\ProcessKeyOptions`. That means that we can use options as keys: all we need is a way to set them. This is available in a command called `\SetKeys`: you might notice the name is similar to the long-standing `\setkeys` from the `keyval` package.

As most of the time we want to set keys *after* loading a package, we need to pass the *family* the keys are in. This is given in an optional argument to `\SetKeys`. (The optional argument applies to all of the other new commands, but most of the time we don’t need to worry about it, as L<sup>A</sup>T<sub>E</sub>X will automatically use the package or class name.) So we might have something like

```
\NewDocumentCommand\mypkgsetup{m}{%
  \SetKeys[mypkg]{#1}%
}
```

We can then use this new setup command to work with exactly the same keys as we have created as options: provided of course we do not try to do that outside their `usage` scope.

#### 5 More flexibility

As the new approach is based on `l3keys`, we can use any key properties that are defined by `l3keys`. That is because of the fact that options are keys and *vice versa*: all we need to do is define the keys in the right place.

- ◇ Joseph Wright  
Northampton, United Kingdom  
joseph dot wright (at)  
morningstar2.co.uk
- ◇ L<sup>A</sup>T<sub>E</sub>X Project Team

## siunitx: Launching version 3

Joseph Wright

### 1 Introduction

Typesetting units is an important task for scientists and engineers. When done in  $\text{\LaTeX}$ , it is natural to use a macro-based approach to carry out the formatting. I entered this area in 2008 with the  $\text{\LaTeX}$  package `siunitx`, which I've looked at in *TUGboat* before (Wright, 2011; Wright, 2018). Last year, I launched version 3 of `siunitx` after development over several years. Here, I will recap why this came about and how I went about launching the latest version.

### 2 Looking back

I started developing `siunitx` in late 2007, when I answered a question about a bug in a previous package, `Slunits` (Heldoorn and Wright, 2007). I quickly decided to write a new package, which combined `Slunits` and `Slstyle` (Els, 2008) with a key–value interface and bringing in some new ideas. The first release of `siunitx` was available at Easter 2008.

This first release worked well, and soon picked up lots of users. However, internally it was essentially a mix of material from the previous packages put together. It also had some poor choices for key names, which I very much wanted to address. I therefore wanted to work on a second implementation, and got on to that in 2010.

Around this time, I started working with `expl3` coding, and quickly decided to move `siunitx` to using `expl3` internally. The re-write meant starting from scratch in some ways, but I got a lot of work done pretty quickly and was quite happy with the results.

### 3 Moving forward again

Version two retains most of the features that version one had, but as well as the good ones, it turns out it retained some issues, particular in the internal API. I needed to look at a system where the different parts of the `siunitx` system communicated with each other using a documented approach. To support that, and to keep things working with existing sources, I needed tests. And there was a big issue with fonts.

The font assumptions in versions 1 and 2 carry all the way through from `Slstyle` (Els, 2008), and which I adjusted only slightly. The approach was:

1. Detect the current font type using  $\text{\LaTeX}$  internal data.
2. Insert everything inside `\text` (an `\mbox`).
3. Apply `\ensuremath` inside the box.
4. Perhaps apply `\text` again (for text mode output).

5. Force the font by using *e.g.* `\mathrm` or `\rmfamily`.

That requires a lot of work, and it's not always easy to get it right. So there is a new approach for version three:

1. Detect current font type using  $\text{\LaTeX}$  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. The downside is that this is quite hard to map in all cases to the older keys: we were going to need a way to deal with this.

#### 3.1 Making it all work

Getting version 3 out needed me to solve three major issues:

1. Testing;
2. Backward compatibility;
3. Handling multi-part and complex values.

Only one of those (the third) is an area for *just me*: the first two are general, and tools from the  $\text{\LaTeX}$  team have helped. Over the past few years, we have developed `l3build` (Wright, 2022) as an automated testing tool. So development of version 3 has been backed by testing for *everything*: each code-level API and all of the key–value interfaces have dedicated tests.

It's not possible to be entirely backward-compatible with the scale of the changes I've made in `siunitx`. Luckily, the  $\text{\LaTeX}$  team have also provided a mechanism to handle this. The user can add

```
\usepackage{siunitx}[=v2]
```

and this will load the last version of `v2`; all I have to do is set things up as a package author. The `siunitx` source thus contains:

```
% Set up a couple of commands in recent-ish
% \LaTeXe{} releases.
\providecommand\DeclareRelease[3]{}
\providecommand\DeclareCurrentRelease[2]{}
%
% Allow rollback to version~$2$: if we need to,
% version~$1$ could eventually be added here too.
\DeclareRelease{2}{2010-05-23}{siunitx-v2.sty}
\DeclareRelease{v2}{2010-05-23}{siunitx-v2.sty}
\DeclareCurrentRelease{}{2021-05-17}
```

The third issue above was dealing with multi-part quantities (*e.g.*  $2\text{ m} \times 3\text{ m}$ ) and complex numbers ( $1 + 2i$ ). I decided to keep the core code faster, and to provide dedicated document commands for these.

Again, this means you might have to update a source to go from version 2 to 3, but I think this is the right call.

#### 4 Conclusions

It's taken a few years, but with the tools available from the L<sup>A</sup>T<sub>E</sub>X team, creating a new version of `siunitx` has worked well. Over a year after the launch, the code is performing well and I've dealt with the minor issues that came up. I'm now looking forward to developing new features that have been outstanding for years.

#### References

- Els, D. “The `Slstyle` package”. 2008. [ctan.org/pkg/sistyle](https://ctan.org/pkg/sistyle).
- Heldoorn, Marcel, and J. Wright. “The `Slunits` package: Consistent application of SI units”. 2007. [ctan.org/pkg/siunits](https://ctan.org/pkg/siunits).
- Wright, Joseph. “`siunitx`: A comprehensive (SI) units package”. *TUGboat* **32**(1), 95–98, 2011. [tug.org/TUGboat/tb32-1/tb100wright-siunitx.pdf](https://tug.org/TUGboat/tb32-1/tb100wright-siunitx.pdf).
- Wright, Joseph. “`siunitx`: Past, present and future”. *TUGboat* **39**(2), 119–121, 2018. [tug.org/TUGboat/tb39-2/tb122wright-siunitx.pdf](https://tug.org/TUGboat/tb39-2/tb122wright-siunitx.pdf).
- Wright, Joseph. “`l3build`: The beginner's guide”. *TUGboat* **43**(1), 40–43, 2022. [tug.org/TUGboat/tb43-1/tb133wright-l3build.html](https://tug.org/TUGboat/tb43-1/tb133wright-l3build.html).

◇ Joseph Wright  
Northampton, United Kingdom  
[joseph dot wright \(at\)  
morningstar2.co.uk](mailto:joseph_dot_wright_at_morningstar2.co.uk)

---

## Case changing: L<sup>A</sup>T<sub>E</sub>X reaches Unicode-land

Joseph Wright, L<sup>A</sup>T<sub>E</sub>X Project Team

### 1 Introduction

The concept of letters having case is familiar to speakers of several languages, most obviously those from Europe using Latin, Greek or Cyrillic scripts. The ability to convert between upper and lower case, *case changing*, is something we might take for granted both for people and for computer systems. However, there are subtleties that a careful implementation needs to take into account.

The Unicode Consortium have defined four different case-related operations that are required to support the range of applications that come up in practical use cases:

- Upper casing
- Lower casing
- Title casing
- Case folding

Here, *title* casing means making the first letter (broadly speaking) upper case, then making the rest of the text lower case. Case folding means removing case, and is needed for programmers: often lower casing is used, but this is not appropriate for true caseless text comparisons.

Unicode have also identified that case changing is not a simple fixed operation: depending on the context around a character, the right outcome can vary, while different languages can have different rules. All of this means that the T<sub>E</sub>X primitives `\lowercase` and `\uppercase` are *not* suitable for case changing with the variety of text we might see today.

### 2 An `expl3` implementation

In 2015, I spoke at the TUG meeting about an `expl3` implementation that sought to provide a full set of Unicode-compliance case changing tools. At that time, only Unicode engines (X<sub>Ǝ</sub>T<sub>E</sub>X and LuaT<sub>E</sub>X) were supported. However, the code could provide all of the requirements of Unicode in an approach that works by expansion: this meant that one could case change text *inside* an `\edef`.

Since that talk, the L<sup>A</sup>T<sub>E</sub>X team have refined ideas about the future of L<sup>A</sup>T<sub>E</sub>X, meaning that the case changer needed to be extended to work with pdfT<sub>E</sub>X. It also needed to be able to carry out something similar to `\protected@edef` by expansion. Both of those ideas have been covered over the past few years.

The `expl3` implementation now also incorporates ideas from David Carlisle's `textcase` package. This is

mainly about being able to exclude content from case changing: math mode material should never be case changed, and it's important to be able to mark individual items as unaffected by case operations. Both of those ideas are relatively easy to do by expansion, as we need to examine each token anyway.

### 3 Bring it to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

Since `\uppercase` and `\lowercase` have long been known to have limitations, I am not the first person to look at supporting the needs of different languages. There have been a number of clever approaches to getting the required mappings from implementations using the T<sub>E</sub>X primitives. However, the new code provides a single consistent interface: it can handle different languages, multiple scripts and so on without needing to load potentially incompatible code.

More importantly for the team, we needed to look again at how active characters are handled in pdfT<sub>E</sub>X. The change, to use engine protection for these active bytes, makes life a lot easier in general but breaks the previous approaches to case changing these characters. The `expl3` code, in contrast, works fine with the protected definitions. So this drove the change.

For users, the long-standing `\MakeUppercase` and `\MakeLowercase` commands stay unchanged: only the implementation has been adjusted. Title-casing has some subtleties that mean it needs a dedicated document command, so now it has one: `\MakeTitlecase`. The command `\NoCaseChange`, originally defined by `textcase`, is also now integrated into the kernel.

For package authors, we have added a command `\AddToNoCaseChangeList` to add commands to the set which are skipped for case changing: things like `\ref` and `\label` are already there. We have also provided `\CaseSwitch`, a command that selects between different outcomes (no change, upper, lower, title case): this is useful if you have something that doesn't expand to text but where you want a different result inside and outside of case changing. Finally, we have added `\DeclareCaseChangeEquivalent` for situations where a package author needs to entirely swap the functionality of a command inside a case changing context.

### 4 The data: pdfT<sub>E</sub>X

One minor wrinkle is the data support, particularly for pdfT<sub>E</sub>X. For the Unicode engines, we can read all of the data automatically. For pdfT<sub>E</sub>X, we don't have `\lccode` and `\uccode` storage for the whole of Unicode, only for the 8-bit range. That means that most of the case changing data has to be stored in macros. To avoid wasting a lot of memory, only code points that are typically typeset with 8-bit engines are included here. That does mean that it's possible a few get missed: if there is something to add, please let us know.

### 5 Looking ahead

For the Spring 2022 release, we have not included support for language variation in the document command interface. But that's on the agenda, and likely to appear in the Fall 2022 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> update. The most likely approach there is as an optional argument to `\MakeUppercase`, *etc.*: watch this space.

- ◇ Joseph Wright  
Northampton, United Kingdom  
joseph dot wright (at)  
morningstar2.co.uk
- ◇ L<sup>A</sup>T<sub>E</sub>X Project Team

## Using spot colors in L<sup>A</sup>T<sub>E</sub>X

Ulrike Fischer

### Abstract

In this paper I recount some practical experiences with spot colors we gained while working on the third edition of *The L<sup>A</sup>T<sub>E</sub>X Companion*. I describe what spot colors are, how to use them for text and (TikZ) graphics, how to mix them properly, and some of the pitfalls we found and how we worked around them.

### 1 Introduction

The *L<sup>A</sup>T<sub>E</sub>X Companion* is printed in two base colors: CMYK black and PANTONE 3005 U, called “blue” in the document. The second is a so-called spot color, a special ink in the printer, and not a mix of CMYK colors.

To prepare the book for print we had to ensure that the CMYK black, the spotcolor and mixes of both are used where needed, and we had to remove or replace all uses of non-black CMYK colors, and all uses of RGB colors.

Using color is quite normal nowadays and so color was found in various places added by a variety of code and packages.

- Various text parts such as headings, side notes and text in examples are colored.
- Page numbers in the header could be colored.
- Crop marks from the `crop` package are colored.
- Various boxes from the `tcolorbox` package looked gray but were actually a mix of RGB colors.
- In one example the use of the `Color` key of `fontspec` led to RGB colors.
- Examples with todo notes and examples with TikZ pictures contained various colors.

Colors are not used only as pure colors. The ease of the `xcolor` syntax means that mixes like `\mouse@body!50!black` are found in many places. So it is not enough to redefine `\mouse@body`; one also has to ensure that the mix uses only the intended colors.

Which colors are used on a page can be checked with various tools. For example Adobe Pro has a tool where one can deselect color plates like black and the Pantone ink and so see if the page also contains other colors.

### 2 What are spot colors?

A spot color is a color described not in CMYK or RGB but in a special color model.<sup>1</sup> Such a color model describes the *ink* or *inks* to use. The amount of ink to use is given with the *tint*, a number between 0 and 1. To also allow devices like a screen or a normal printer to represent the color the spot color model contains also a fallback in a generic color model like CMYK.

The simplest type of a spot color model is a “separation” which describes the use of a single ink. Such a separation has a rather simple setup in a PDF. First one has to add an object with an array which contains the name of the ink, and a function which maps a tint to a CMYK color:

```
6 0 obj % <-- object number
[ /Separation/PANTONE#203005#20U
% ^-- name of the ink, PANTONE 3005 U
/DeviceCMYK % <-- fallback CMYK
  << /FunctionType 2
    /Domain [0 1]
    /C0 [0 0 0 0]
    /C1 [1 0.56 0 0]
    /N 1
  >>
]
endobj
```

Then one has to declare a name for this object. This is done in the page resources in the `/ColorSpace` array with a simple mapping between the chosen name and the object number:

```
/ColorSpace <<
/color1 6 0 R
% ^-- name to reference the Separation object
...
>>
```

After the setup the color model can then be activated with the `cs` and `CS` operators and the value of the tint can be given with the `scn` and `SCN` operators,<sup>2</sup> where the lowercase operators set the fill color, and the uppercase version the stroke color.

name	value/tint
<code>/color1 cs 1.0 scn</code>	%<-- fill color
<code>/color1 CS 1.0 SCN</code>	%<-- stroke color
<code>[(TEXT)]</code>	%<-- following text

<sup>1</sup> The PDF reference calls color models “color spaces”, but I will stick mostly to “color model” here.

<sup>2</sup> PostScript uses a reversed notation, so typically a value is *before* an operator

### 3 Methods provided by the L<sup>A</sup>T<sub>E</sub>X kernel

#### 3.1 Setup of a separation model

A separation color model can be set up with commands included in the L<sup>3</sup> programming layer of L<sup>A</sup>T<sub>E</sub>X. `\color_model_new:nnn` is the main command. It has three arguments, the first is a freely chosen name for the color model, the second describes the type of the model — here “Separation” — and the third is a key/value argument to set the details. As such a color model has to write to the PDF page resources, it is required to load the new PDF management of L<sup>A</sup>T<sub>E</sub>X [2]. This can be done by using the `\DocumentMetadata` command at the begin of the document.

The color model for our Pantone ink can then be declared as in the following listing.

```
% required, loads pdfmanagement:
\DocumentMetadata{
\documentclass{article}
\ExplSyntaxOn
\color_model_new:nnn { pantone3005 }
  { Separation }
  { name = PANTONE~3005~U ,      % ink
    alternative-model = cmyk , % fallback
    alternative-values = {1, 0.56, 0, 0}
  }
\ExplSyntaxOff
```

Colors in this model can then be defined with the `\color_set:nnn` command, which has three arguments: a name for the color, the just-defined model, and a value for the tint.

```
\ExplSyntaxOn
% define color spotA:
\color_set:nnn{spotA}{pantone3005}{1}
% define color spotB (less tint):
\color_set:nnn{spotB}{pantone3005}{0.5}
\ExplSyntaxOff
```

These colors can then be used with the command `\color_select:n`. To ease use in a document it is sensible to define an alias which can be used without having to switch on the `expl3` syntax:

```
\ExplSyntaxOn
% define user command \colorselect:
\cs_set_eq:NN \colorselect \color_select:n
\ExplSyntaxOff

\colorselect{spotA} spot A
% result in the PDF:
%/color1 cs 1.0 scn /color1 CS 1.0 SCN

\colorselect{spotB} spot B
% result in the PDF:
%/color1 cs 0.5 scn /color1 CS 0.5 SCN
```

#### 3.2 Mixing colors

The color commands of the L<sup>3</sup> programming layer support mixing colors in similar ways to the methods provided by the `xcolor` package: You specify integers describing the percentage surrounded by exclamation points between defined color names.

Mixing colors defined in the same color model, e.g., two CMYK colors or two RGB colors or two different tints of a separation model, is straightforward and involves only some math. But when mixing colors of different models one has to decide which is the target color model and then convert all colors into this target model to be able to mix them. There exist various formulas on how to convert RGB into CMYK or Gray and so normally users don’t have problems to mix colors defined in these standard models. Converting a spot color into CMYK is easy too as it can be done with the fallback function, but converting an arbitrary CMYK color into a spot color is not always possible, as it is not clear how to map, e.g., a red to a tint value of a bluish spot color.

The color command of L<sup>A</sup>T<sub>E</sub>X L<sup>3</sup> programming layer uses as the target color model of a mix the color model of the first color<sup>3</sup> and then tries its best to output some color. It will not report an error even if the models are not compatible. It is thus your responsibility to check that the mixes do what you want them to do.

##### 3.2.1 Mixing with white

Mixing a spot color with white normally works fine. It changes the tint and makes the color light, and this is the expected outcome in most cases. You need only pay attention to the order: always start with the spot color.

```
\colorselect{spotA!50!white}
%/color1 cs 0.5 scn /color1 CS 0.5 SCN %good

\colorselect{white!50!spotA}
%/color1 cs 1.0 scn /color1 CS 1.0 SCN %wrong
```

##### 3.2.2 Mixing with black

Mixing with black is more difficult. Black is an ink of its own and when mixing it into the spot color one wants to add some of this ink. With the standard mix, this does not happen. As the following listing shows, our `spotA` doesn’t change at all if black is mixed in, while `spotB` gets a bit darker as the tint value increases, but this is also not what we want: we don’t want more tint but more black ink.

<sup>3</sup> It is possible to force another target model, but this is not discussed here; check the documentation for details.

```
\colorselect{spotA!50!black}
%/color1 cs 1 scn /color1 CS 1 SCN

\colorselect{spotB!50!black}
%/color1 cs 0.75 scn /color1 CS 0.75 SCN
```

The right way to mix in black is to set up another spot color model. This case is not a simple separation model, but a DeviceN color model which supports describing ink mixtures.

In the PDF such a DeviceN model is again a rather simple object. It contains the keyword `/DeviceN`, an array which describes the ink components,<sup>4</sup> and again a function for the CMYK fallback. Similar to the separation model a name must be declared in the `/ColorSpace` resource which can then be used to select the color in the page stream.

```
44 0 obj
[ /DeviceN % the components:
  [ /PANTONE#203005#20U /Black ]
  /DeviceCMYK 45 0 R % fallback info
  ...
]

% name declaration:
/ColorSpace [... /color2 44 0 R ...]
```

A color in this model can then be called in the PDF as before, but now the `scn/SCN` operator expects two values, one for the Pantone component and the other for the black ink:

name	two values
↓	↓
<pre>/color2 cs 0.5 0.5 scn /color2 CS 0.5 0.5 SCN</pre>	

Such a DeviceN color model can also be set up in L<sup>A</sup>T<sub>E</sub>X with the `\color_model_new:nnn` command. The type argument then takes the string `DeviceN` and in the third argument the names of the components are given, here our previously-defined Pantone color, and black as a predefined ink.

```
\color_model_new:nnn { pantone+black }
{ DeviceN }
{
  names = {pantone3005,black} % components
}
```

After the DeviceN model has been set up, colors can be defined in this model. Definitions of “pure” colors which use only one component are useful, as such colors can be used to mix colors of this model on the fly.

<sup>4</sup> This can be also three or more components

```
\color_set:nnn{mix} {pantone+black}{0.5,0.5}
\color_set:nnn{purepantone}{pantone+black}{1,0}
\color_set:nnn{pureblack} {pantone+black}{0,1}

\colorselect{mix}
%/color2 cs 0.5 0.5 scn /color2 CS 0.5 0.5 SCN

\colorselect{purepantone!70!pureblack}
%/color2 cs 0.7 0.3 scn /color2 CS 0.7 0.3 SCN
```

### 3.3 Multi-model colors

While defining and using a color like `pureblack` solves the problem of mixing in black, it is a bit of a problem that a new color name has to be used. Mixtures with black are quite common and one has to change the name in various places. One option to avoid this could be to redefine black to always use the DeviceN model.

Another option is to make use of the capability of the L<sup>A</sup>T<sub>E</sub>X color implementation to define a color in more than one model at once as shown in the next listing. Such a model will behave like a CMYK color if used on its own or when mixed with other CMYK colors, but behave like a DeviceN color when used in this context.

Again, be aware that the order of the colors in the color expression matters and that the main color of the *first* color is used as the target color model!

```
\color_set:nnn {black}
  {cmyk / pantone+black}
  {0,0,0,1 / 0,1}

\colorselect{black}
% cmyk black in the PDF:
% 0 0 0 1 k 0 0 0 1 K

\colorselect{purepantone!50!black}
% Mix as DeviceN:
%/color2 cs 0.5 0.5 scn /color2 CS 0.5 0.5 SCN

\colorselect{black!50!purepantone}
% cmyk mix:
%0.5 0.28 0 0.5 k 0.5 0.28 0 0.5 K
```

### 3.4 Fill and stroke color

Up to now we have always set the fill and stroke color to the same value. This is quite normal for text, but not for graphics, and so the kernel color code allows to select the colors independently. Figure 1 shows an example with the help of the `l3draw` package. The two spot colors are faked by two shades of gray.

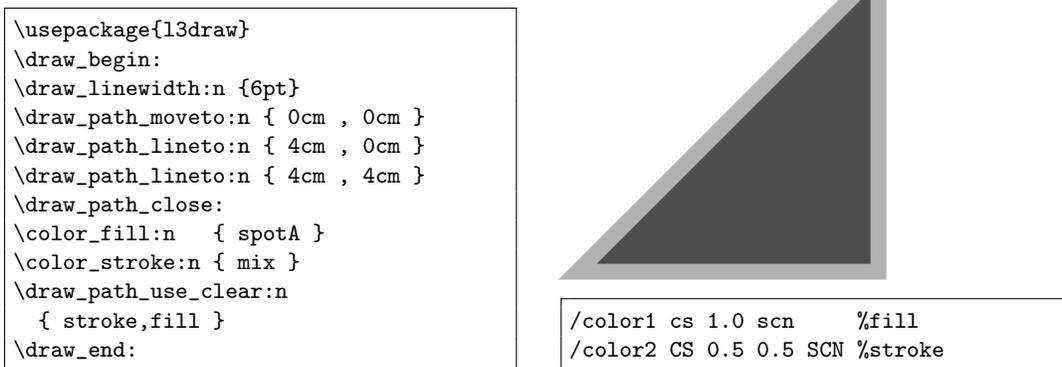


Figure 1: Setting fill and stroke color independently.

### 3.5 Coloring fonts with fontspec

One of the examples in the book demonstrates the use of the `Color` key of the `fontspec` package [3]. Test compilations showed that regardless of how the color is defined, the `fontspec` package inserts an RGB color into the PDF. When using  $X_{\text{L}}\text{A}\text{T}\text{E}\text{X}$  this probably cannot be changed, but with  $\text{L}\text{u}\text{A}\text{L}\text{A}\text{T}\text{E}\text{X}$  a solution was implemented. If the color is defined with the kernel commands, the PDF management is loaded, and a current `luaotfload` is used, the model of the color is honored by `fontspec` and even spot colors can be used without problems.

### 3.6 Summary of spot colors with the kernel methods

- It is easy to set up the models and the colors.
- It should work with all backends.
- The colors work fine for text.
- One has to pay attention when mixing colors of different models.
- It is possible to define colors in more than one model.
- The kernel command can handle fill and stroke color separately.
- The colors can be used with `fontspec` ( $\text{L}\text{u}\text{A}\text{L}\text{A}\text{T}\text{E}\text{X}$  only).
- But: The kernel colors aren't yet supported by  $\text{TikZ}$  [4] (or don't support  $\text{TikZ}$ , depending on one's point of view).

The last point meant that we had to look for an alternative for `tcolorboxes`, `todo` notes, various examples with pictures, etc.

## 4 The alternative: the colorspace package

The `colorspace` package [1] also offers tools to set up spot colors. It supports only  $\text{pdf}\text{L}\text{A}\text{T}\text{E}\text{X}$  and  $\text{L}\text{u}\text{A}\text{L}\text{A}\text{T}\text{E}\text{X}$ .

Unlike the kernel commands it doesn't offer separate commands to use the colors but hooks into the `xcolor` package. This allows documents to use the standard `\color` command, and it also means that  $\text{TikZ}$  is at least in part supported.

The setup of spot colors is quite similar to the kernel commands, but a bit less verbose. The main command for a separation model defines the model and a color with tint directly in one step:

```

% definition of color spotC:
\definespotcolor{spotC}% color name
        {PANTONE 3005 U}% ink
        {1,0.56,0,0}% CMYK fallback

% use with standard \color command:
\color{spotC} Spotcolor

% in the PDF:
/&PANTONE#203005#20U cs
/&PANTONE#203005#20U CS
1 sc 1 SC

```

When mixing colors into a spot color, `colorspace` will give an error for every mix of color models it doesn't know and will not try, like the  $\text{L}\text{A}\text{T}\text{E}\text{X}$  commands, to produce a color nevertheless. This can be sometimes a blessing as it warns you if a faulty mix is somewhere, but also means that any code using such a mix must be adapted; it is not possible to simply accept a slightly imperfect fallback.

A `DeviceN` color model can also be defined with the `colorspace` package. For this two commands are needed. First you set up the model with the command `\definecolorspace` and the keyword `mixed`, and then you can define colors as usual with `\definecolor`. Here too it makes sense to define orthogonal, pure colors which can be used in mixes.

```
\definecolorspace{pantone+black}
  {mixed}
  {spotC,black}
\definecolor{purepantone}{pantone+black}{1,0}
\definecolor{pureblack}{pantone+black}{0,1}

\color{purepantone!50!pureblack}
```

`colorspace` doesn't support multi-model color definitions (as it is built on `xcolor` which doesn't support this either), and so to mix a spot color with black you either have to redefine black or use the `pureblack` where needed.

`colorspace` also doesn't have commands to set stroke and fill color independently, and the method it uses to store the spot colors into the `xcolor` data model is not known to `TikZ`. The support for `TikZ` is implemented through a number of patches, is sketchy and even has a few bugs which we found during the tests. The most problematic one is that it can “lose” the color model during some `\colorlet` operations. For example if a color is copied through the current color, which is represented by a period, or copied with the `named` keyword, the color model is suddenly zero. This results in a broken, unusable PDF. As such operations are very common in `TikZ` we had to implement a few patches to get at least syntactically correct color calls.

```
\colorlet{.}{spot}
\colorlet{newcolor}{.}
0 cs 0 CS 1 sc 1 SC %broken PDF

\colorlet[named]{test}{spot}
0 cs 0 CS 1 sc 1 SC %broken PDF
```

#### 4.1 The fill and stroke color problem

As written above, `colorspace` supports `TikZ` only partially. The core of the problem is that `xcolor` (likewise `color`) doesn't provide interfaces to access and use fill and stroke colors independently. `color` for example stores a color as backend instructions for both colors:

```
% fill and stroke together:
{0 0 1 0 k 0 0 1 0 K}
```

`xcolor` stores more data, but still keeps the backend instructions for fill and stroke together:

```
\xcolor@
{}
% fill and stroke together:
{0 0 1 0 k 0 0 1 0 K}
{cmyk}
{0,0,1,0}
```

This missing support for fill and stroke colors means that `TikZ` and other graphic packages cannot rely only on the interface provided by `xcolor`, but have to implement and use their own backend commands in various places to split out the two parts. The resulting mix of interface commands and low-level commands makes it difficult for `colorspace` (or for the kernel) to fully support spot colors in `TikZ`.

## 5 Special `TikZ` problems

Beside the general problem of missing support for fill and stroke colors, there are a few more specific problems regarding spot colors in `TikZ`.

### 5.1 Shadings

Shadings are not simply drawn with some color, but are special objects defined in the PDF and typically contain instructions about which color model to use.

By default `TikZ` only creates RGB shadings. If you use a shading, you can then see in the PDF lines where the `DeviceRGB` points to RGB color model:

```
/Shading << /Sh <<
  /ShadingType 2 /ColorSpace /DeviceRGB ...
```

Some time ago David Purton also added support for CMYK shadings, which you get if you force `xcolor` to use CMYK everywhere.

```
/Shading << /Sh <<
  /ShadingType 2 /ColorSpace /DeviceCMYK
```

There is no support for shadings using spot colors yet, and there was no time to investigate this use, so the book restricted the use of shadings to grayscale and CMYK shadings which use only the black component.

### 5.2 Patterns

A similar problem was found with patterns. In PDF format, patterns are also special objects and refer to a color model. All patterns, colored and uncolored alike, created by `TikZ` use hard-coded RGB. To change this we added an additional declaration to the `/ColorSpace` resource and patched an internal command to force the use of this declaration and with this, were able to show at least a black and white pattern:

```
\pgfutil@addpdfresource@colorspaces
  { /tlc3pattern [/Pattern /DeviceCMYK] }

\def\pgfsys@setpatternuncolored#1#2#3#4{%
  \pgfsysprotocol@literal{%
    /tlc3pattern cs 0 0 0 1
    /pgfpat#1\space scn}}
```

## 6 Conclusion

With the kernel commands and the `colorspace` package two robust options to use spot colors for text and rules are available. The main work for authors here is to check color expressions which mix colors, and to check and perhaps overwrite default color definitions in packages they use.

The situation for major graphic packages such as `TikZ` (`PSTricks` is probably similar) is not so satisfactory, as they use low-level code to set fill and stroke colors, making it difficult to add support for new color models. Also they hard-code in various places color models like RGB or CMYK. But resolving these problems should be possible as the kernel now provides a more powerful color interface, and the main task is to bring them together.

## References

- [1] J. Bezos. *The Colorspace package*. Provides PDF color spaces. [ctan.org/pkg/colorspace/](https://ctan.org/pkg/colorspace/)
- [2] L<sup>A</sup>T<sub>E</sub>X Project Team. *The Pdfmanagement-testphase package*. [ctan.org/pkg/pdfmanagement-testphase](https://ctan.org/pkg/pdfmanagement-testphase)
- [3] W. Robertson, L<sup>A</sup>T<sub>E</sub>X Project Team. *The Fontspec package*. Advanced font selection in X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X and LuaL<sup>A</sup>T<sub>E</sub>X. [ctan.org/pkg/fontspec](https://ctan.org/pkg/fontspec)
- [4] T. Tantau, C. Feuersänger, et al. *The PGF package*. Create PostScript and PDF graphics in T<sub>E</sub>X. [ctan.org/pkg/pgf](https://ctan.org/pkg/pgf)

◇ Ulrike Fischer  
L<sup>A</sup>T<sub>E</sub>X Project Team  
Bonn  
Germany  
[ulrike.fischer \(at\)  
latex-project.org](mailto:ulrike.fischer@latex-project.org)

---

## The luatruhtable L<sup>A</sup>T<sub>E</sub>X package

Chetan Shirore, Ajit Kumar

### Abstract

This paper describes the development and usage of the *luatruhtable* package in L<sup>A</sup>T<sub>E</sub>X. It is developed to generate truth tables of boolean values in L<sup>A</sup>T<sub>E</sub>X documents. The package provides an easy way of generating truth tables in L<sup>A</sup>T<sub>E</sub>X. There is no need of a special L<sup>A</sup>T<sub>E</sub>X environment for generation of truth tables with the package. It is written in Lua and the T<sub>E</sub>X document is to be compiled with the LuaL<sup>A</sup>T<sub>E</sub>X engine.

### 1 Introduction

The Lua [1] programming language is a scripting language which can be embedded across platforms. With LuaT<sub>E</sub>X [4], it is possible to use Lua in L<sup>A</sup>T<sub>E</sub>X. T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X has scope for programming in themselves. However, with the internals of T<sub>E</sub>X there are several limitations especially for performing calculations on numbers in L<sup>A</sup>T<sub>E</sub>X documents. There are packages like *pgf* [5] and *xparse* [9] in L<sup>A</sup>T<sub>E</sub>X which provide some programming capabilities inside L<sup>A</sup>T<sub>E</sub>X documents. However, such packages are not meant to provide the complete programming structure that in general other programming languages (like Lua) provide.

The generation of truth tables with these packages in L<sup>A</sup>T<sub>E</sub>X gets complicated [7] and probably without using Lua it can't be done in an easier way in L<sup>A</sup>T<sub>E</sub>X. The programming facilities of Lua are effectively used in the *luatruhtable* package. The *xkeyval* package is used in its development, in addition to the *luacode* package [2]. The time for generation of truth tables using the package and compilation of T<sub>E</sub>X document with LuaT<sub>E</sub>X is not an issue.

### 2 Installation and inclusion

The installation of the *luatruhtable* package is similar to any L<sup>A</sup>T<sub>E</sub>X package, where the `.sty` file is placed in the L<sup>A</sup>T<sub>E</sub>X directory of the texmf tree. The package can then be used by including the usual command `\usepackage{luatruhtable}` in the preamble of the L<sup>A</sup>T<sub>E</sub>X document. The document is to be compiled using LuaL<sup>A</sup>T<sub>E</sub>X.

### 3 Core ideas used in the development of the package

Lua [1] is an extensible language that can be embedded in L<sup>A</sup>T<sub>E</sub>X. The T<sub>E</sub>X [8] language has indirect support for scripting languages [6].

The *luatruhtable* function `toBinary(x,y)` is used to produce a sequence of *True* and *False* values

of boolean variables. This function converts the decimal number  $x$  to a binary number by adding  $y$  number of leading zeroes. The result of this is stored in a table in Lua. Here  $y$  corresponds to the number of boolean variables. As  $2^y$  permutations of boolean variables are to be produced, the function `toBinary(x,y)` runs inside a loop where  $x$  takes values from 1 to  $y$ . The splitting of variables and expressions is done using string methods available in Lua. The nested *metamethods* in Lua are used to define several logical operators. The *load* function in Lua is used to evaluate logical expressions.

#### 4 The `\luaTruthTable` command in the `luatruthtable` package

The `\luaTruthTable` command is the main command in the `luatruthtable` package which generates truth tables. It has the following syntax.

```
\luaTruthTable[⟨trtext⟩,⟨fttext⟩]
  {⟨list of boolean / logical variables⟩}
  {⟨list of expressions⟩}
```

The command has two mandatory arguments:

- i) *⟨list of boolean / logical variables⟩*: The list of boolean or logical variables should be separated by commas.
- ii) *⟨list of expressions⟩*: The list of logical expressions that are to be evaluated should also be separated by commas.

And the command has two optional arguments:

- i) *⟨trtext⟩*: the display value for the boolean value *True*. It has the default value `$T$` in the package. It can be any string or number, although assigning the value 0 would not make sense.
- ii) *⟨fttext⟩*: the display value for the boolean value *False*. It has the default value `$F$` in the package. It can be any string or number, although assigning the value 1 would similarly not make sense.

The `\luaTruthTable` command should be used within `\begin{tabular} ... \end{tabular}` environment or any other environment in L<sup>A</sup>T<sub>E</sub>X for tables. The sequence of column heads should be the same as the sequence of *list of boolean / logical variables* and *list of expressions*. Without these correct inputs, `\luaTruthTable` cannot produce the desired output.

#### 5 Operations in the `luatruthtable` package

- a) *not*: The value of *not p* is False when  $p$  is True and it is True when  $p$  is False.

$p$	not $p$
$T$	$F$
$F$	$T$

The command `lognot*` is used in the package to generate a truth table for the *not* operation.

- b) *and*: The value of  $p$  AND  $q$  is True if and only if both  $p$  and  $q$  are True.

$p$	$q$	$p$ and $q$
$F$	$T$	$F$
$T$	$F$	$F$
$T$	$T$	$T$
$F$	$F$	$F$

The command `*logand*` is used in the package to generate a truth table for the *and* operation.

- c) *or*: The value of  $p$  or  $q$  is False if and only if both  $p$  and  $q$  are False.

$p$	$q$	$p$ or $q$
$F$	$T$	$T$
$T$	$F$	$T$
$T$	$T$	$T$
$F$	$F$	$F$

The command `*logor*` is used in the package to generate a truth table for the *or* operation.

- d) *implies*: The value of  $p$  implies  $q$  is False if and only if  $p$  is True and  $q$  is False.

$p$	$q$	$p$ implies $q$
$F$	$T$	$T$
$T$	$F$	$F$
$T$	$T$	$T$
$F$	$F$	$T$

The command `*imp*` is used in the package to generate a truth table for the *implies* operation.

- e) *if and only if*: The value of  $p$  if and only if  $q$  is True if and only if both  $p$  and  $q$  have same truth values.

$p$	$q$	$p$ iff $q$
$F$	$T$	$F$
$T$	$F$	$F$
$T$	$T$	$T$
$F$	$F$	$T$

The command `*iff*` is used in the package to generate a truth table for the *if and only if* operation.

- f) *NAND*: The value of  $p$  NAND  $q$  is 0 if and only if both  $p$  and  $q$  have value 1.

$p$	$q$	$p$ NAND $q$
0	1	1
1	0	1
1	1	0
0	0	1

The command `*lognand*` is used in the package to generate a truth table for the *NAND* operation.

- g) *XOR*: The value of  $p$  XOR  $q$  is 0 if and only if  $p$  and  $q$  have same values.

$p$	$q$	$p$ XOR $q$
0	1	1
1	0	1
1	1	0
0	0	0

The command `*logxor*` is used in the package to generate a truth table for the *XOR* operation.

- h) *NOR*: The value of  $p$  NOR  $q$  is 1 if and only if both  $p$  and  $q$  have value 0.

$p$	$q$	$p$ NOR $q$
0	1	0
1	0	0
1	1	0
0	0	1

The command `*lognor*` is used in the package to generate a truth table for the *NOR* operation.

- i) *XNOR*: The value of  $p$  XNOR  $q$  is 1 if and only if both  $p$  and  $q$  have same values.

$p$	$q$	$p$ XNOR $q$
0	1	0
1	0	0
1	1	1
0	0	1

The command `*logxnor*` is used in the package to generate a truth table for the *XNOR* operation.

Table 1 summarises logical operators in the package.

## 6 Examples and usage

The *luatruthtable* package accepts a finite number of variables. It supports any finite number of variables that one would need in practice. A few examples of usage are given here.

The following example involves three variables,  $p$ ,  $q$ , and  $r$ .

```
\begin{tabular}{|ccc|c|}
\hline
\((p\) & \((q\) & \((r\) & \((p \land q)\) \\
\rightarrow & \(\neg r\) & \\
\hline
\luaTruthTable{p,q,r}{(p*logand*q) *imp*
(lognot*r)} \\
\hline
\end{tabular}
```

The output from the above is shown in Table 2.

Here `lognot*r` is enclosed in parentheses to produce correct results in the generated truth table.

**Table 1:** Operations in the *luatruthtable* package, given boolean variables  $p$  and  $q$ .

Command	Description
<code>lognot*p</code>	Negates the boolean variable $p$ .
<code>p*logand*q</code>	Truth table for the expression $p$ and $q$ .
<code>p*logor*q</code>	Truth table for the expression $p$ or $q$ .
<code>p*imp*q</code>	Truth table for the expression <i>if <math>p</math> then <math>q</math></i> .
<code>p*iff*q</code>	Truth table for the expression <i><math>p</math> if and only if <math>q</math></i> .
<code>p*lognand*q</code>	Truth table for the expression $p$ NAND $q$ .
<code>p*logxor*q</code>	Truth table for the expression $p$ XOR $q$ .
<code>p*lognor*q</code>	Truth table for the expression $p$ NOR $q$ .
<code>p*logxnor*q</code>	Truth table for the expression $p$ XNOR $q$ .

The following is the code generated by the command `\luaTruthTable` in the above code.

```
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$ \\
$$$ & $$$ & $$$ & $$$
```

With the use of optional arguments [*trtext=True*, *fttext=False*] in the previous example, one gets the following output:

$p$	$q$	$r$	$(p \wedge q) \rightarrow \neg r$
False	False	True	True
False	True	False	True
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	False
False	False	False	True

**Table 2:** Example output from the `\luaTruthTable` command.

$p$	$q$	$r$	$(p \wedge q) \rightarrow \neg r$
$F$	$F$	$T$	$T$
$F$	$T$	$F$	$T$
$F$	$T$	$T$	$T$
$T$	$F$	$F$	$T$
$T$	$F$	$T$	$T$
$T$	$T$	$F$	$T$
$T$	$T$	$T$	$F$
$F$	$F$	$F$	$T$

It is possible to give `trtext` and `trfalse` values that are  $\text{\TeX}$  math text. So with the use of optional arguments [`trtext=$T$`, `fltext=$F$`] in the previous example, one gets the following output:

$p$	$q$	$r$	$(p \wedge q) \rightarrow \neg r$
$F$	$F$	$T$	$T$
$F$	$T$	$F$	$T$
$F$	$T$	$T$	$T$
$T$	$F$	$F$	$T$
$T$	$F$	$T$	$T$
$T$	$T$	$F$	$T$
$T$	$T$	$T$	$F$
$F$	$F$	$F$	$T$

Since the `luacode*` environment is used, the backslash is to be escaped in setting `trtext` and `trfalse`. For example: [`trtext=\\(True\\)`, `fltext=\\(False\\)`].

## 7 Known issues, limitations and scope of the package

The associativity and precedence of operators is not yet supported. Thus the package can give appropriate results only when parentheses are used for each of the operations, and gives erroneous results when parentheses are not used. This point is of utmost importance in using the package.

There is no native way of defining a custom operator in Lua [3]. However, some metamethods can be nested in a way to replicate an operator. All operators defined in this package are instances of such nesting. The question may be raised that are there better ways of accomplishing these in Lua. The answer is yes. The alternative ways may be better in one way or another. For example, instead of defining `*logand*` operator and using it in the fashion `p*logand*q`, one could define function `logand` that takes two arguments and use it in a way `logand(p,q)`. But when it comes to embedding in  $\text{\LaTeX}$ , one has to use more and more nested parentheses as the number of statements and operations increases. This

is the exact reason why this approach was not used in the development of the package. Instead of using `implies(logand(p,logor(q,r)),s)` it seems more natural to use `(p*logand*(q*logor*r))*implies*s`.

Also, there is no error handling mechanism used in the package. It relies on the error handling of Lua and  $\text{\TeX}$  itself. The package currently supports the nine listed operations, viz. `not`, `and`, `nand`, `or`, `xor`, `implies`, `iff`, `nor`, `xnor`. Error handling and extending the number of operations may be considered in future versions of the package.

The package, including source code, is released through CTAN ([ctan.org/pkg/luatruthtable](https://ctan.org/pkg/luatruthtable)), and in the usual  $\text{\TeX}$  distributions.

## References

- [1] Lua 5.4 reference manual. <https://www.lua.org/manual/5.4>
- [2] Luacode package page, 2012. <https://ctan.org/pkg/luacode>
- [3] Lua custom operators. <http://lua-users.org/wiki/CustomOperators>
- [4] Luatex package page. <https://ctan.org/pkg/luatex>
- [5] PGF package page. <https://ctan.org/pkg/pgf>
- [6] W.M. Richter.  $\text{\TeX}$  and scripting languages. *TUGboat* 25(1):71–88, 2004. <https://tug.org/TUGboat/tb25-1/richter.pdf>
- [7] StackExchange. Macro for automating truth tables. <https://tex.stackexchange.com/questions/505994>
- [8] Wikipedia.  $\text{\TeX}$ . <https://en.wikipedia.org/wiki/TeX>
- [9] Xparse package page. <https://ctan.org/pkg/xparse>
  - ◇ Chetan Shirore  
Department of Mathematics,  
K.T.H.M. College, Nashik  
422002, Maharashtra, India  
`chetanshirore (at) kthmcollege dot ac dot in`
  - ◇ Ajit Kumar  
Department of Mathematics,  
Institute of Chemical  
Technology, Mumbai  
400019, Maharashtra, India

## L<sup>A</sup>T<sub>E</sub>X classes for doctoral theses in Ukraine: Interesting tips and painful problems

Oleksandr Baranovskyi

### Abstract

In this paper, I introduce `vakthesis`, a bundle of L<sup>A</sup>T<sub>E</sub>X classes for typesetting doctoral theses according to official requirements in Ukraine, discuss the current status of the project and future development plans. Some L<sup>A</sup>T<sub>E</sub>X programming tricks that I have studied are considered.

### 1 Introduction

`vakthesis` is a bundle of L<sup>A</sup>T<sub>E</sub>X classes for typesetting doctoral theses (or dissertations) in Ukraine [7].

The bundle consists of the following main components: `vakthesis` and `vakaref` are traditional classes for a thesis and for a summary (автореферат in Ukrainian) respectively; while `mon2017dev` and `mon2017dev-aref` are modern classes for a thesis and for a summary respectively.

Traditional classes conform to the now “obsolete” official format required by the VAK (Вища атестаційна комісія, i.e., Higher Attestation Commission). Now they are suitable for previously defended theses.

In 2017, the MON (Міністерство освіти і науки, i.e., Ministry of Education and Science) published its own style guide. Modern classes conform to this “new” official format by the MON. These classes are recommended for persons defending their theses now. They are based on the `vakthesis` and `vakaref` classes respectively and cannot work without them.

The `vakthesis` bundle contains a number of example files: the main L<sup>A</sup>T<sub>E</sub>X file of a thesis/summary, introduction, chapter 1, conclusion, bibliography, and B<sup>V</sup>T<sub>E</sub>X files.

So, a thesis author can use these example files as a template for his/her thesis and will receive a thesis/summary with a properly formatted title page of a thesis (and cover of a summary), headings of chapters, sections, subsections, etc., numeration for pages, chapters, sections, subsections, etc., as well as appendices, captions of figures and tables, theorems, lemmata, definitions, etc., list of references, and (in modern classes only) an abstract and list of keywords.

### 2 Tips and problems

In this section, I will discuss some programming tricks related to `vakthesis` development. The first and second cases give tips on how one can define a command with optional arguments of a special form and avoid B<sup>V</sup>T<sub>E</sub>X restrictions, respectively. The remaining three cases are devoted to the problems

with UTF-8 encoding, checking of package loading, and building classes on the base of other classes.

#### 2.1 Command `\speciality` with two optional arguments

In Ukraine, any thesis should be classified according to the so-called List of Specialities.

This is a list of correspondences between speciality code and speciality name and field of science. For example, 01.01.01 is a code for speciality математичний аналіз (Mathematical Analysis) and field of science фізико-математичні науки (Physical and Mathematical Sciences). In this case, a PhD student will be awarded the academic degree of кандидат фізико-математичних наук (Candidate of Physical and Mathematical Sciences).

In the `vakthesis` bundle, there is a particular command `\speciality`. In the standard situation it requires one mandatory argument:

```
\speciality{01.01.01}
```

For a given speciality code, this command takes a speciality name and a field of science from a special plain-text database. Then this information is typeset on the title page of the thesis.

The `\speciality` command has two optional arguments if, for some reason, the user needs to provide a speciality name or a field of science:

```
\speciality[математичний аналіз]{01.01.01}
[фізико-математичних наук]
```

For example, some specialities in the list have an ambiguous form such as теорія та методика навчання (з галузей знань). It means Theory and Methodology of Learning (on some field of knowledge).

In this case the thesis author should choose an appropriate field of knowledge and write, for example, теорія та методика навчання математики (Theory and Methodology of Learning of Mathematics) on the title page.

On the other hand, for some specialities, a degree can be awarded on various fields of science. For example, for the speciality with code 01.04.07, a PhD student can obtain a degree either of кандидат фізико-математичних наук (Candidate of Physical and Mathematical Sciences) or of кандидат технічних наук (Candidate of Technical Sciences) according to the specific nature of the research.

These cases cannot be processed by an algorithm, so optional arguments of the `\speciality` command can be used:

```
\speciality[теорія та методика навчання
математики]{13.00.02}
```

and

`\speciality{01.04.07}[технічних наук]` respectively. Of course both optional arguments may be used simultaneously as in the above-mentioned example for speciality code 01.01.01.

Hence the command `\speciality` cannot have the following syntax:

```
\speciality[{something}][{something}]{{something}}
```

and cannot be defined by the standard L<sup>A</sup>T<sub>E</sub>X command `\newcommand`.

To define the command with two optional arguments around the mandatory argument we can use commands `\def` and `\@ifnextchar`. Below I will give a simplified pseudocode to demonstrate the main idea.

```
\def\speciality{%
  \@ifnextchar[\a@cmd\aa@cmd
}
\def\a@cmd[#1]{%
  % Process speciality name #1
  \@speciality
}
\def\@speciality#1{%
  % Process speciality code #1
  \@ifnextchar[\a@cmd\b\a@cmd@bb
}
\def\a@cmd@b[#1]{%
  % Process field of science #1
}
\def\a@cmd@bb{%
  % Find field of science in the DB
}
\def\aa@cmd#1{%
  % Process speciality code #1
  \@ifnextchar[\aa@cmd\b\aa@cmd@bb
}
\def\aa@cmd@b[#1]{%
  % Process field of science #1
  % Find speciality name in the DB
}
\def\aa@cmd@bb{%
  % Find speciality name in the DB
  % Find field of science in the DB
}
```

This is a very simple idea; and this idea just works. But this code is difficult to maintain as well as to extend to three or more optional arguments.

This is a reason why I reject this idea in a modern version of the classes. Now I use the `xkeyval` package [1] to provide a simple key–value interface:

```
\speciality[
  specialityname=Теорія і методика професійної
    освіти,
  degreefield=педагогічні,
  % specialityfile={filename}.csv
]{13.00.04}
```

A useful overview of how the L<sup>A</sup>T<sub>E</sub>X key–value system works is given in [8].

## 2.2 Environment `bibset` to make two reference lists in a thesis

It is known that BIB<sub>T</sub>E<sub>X</sub> can generate only one reference list in a document, i.e., only one command `\bibliography` can be processed in standard situations. Nevertheless, sometimes there is a need to have more than one reference list. In particular, in Ukrainian theses, the following two lists may exist: the list of referenced sources and the list of author’s publications.

There are many solutions for multiple bibliographies [5, 6] but they are not suitable for me.

Essentially, the `multibbl` and `multibib` packages provide a special bibliography “tag”. So, special “tagged” `\cite` and `\bibliography` commands are available for the user.

Similarly, with the `splitbib` package, the user needs to categorize citations in the document.

The `bibtopic` package separates different bibliographies on different `.bib` files and uses special commands instead of the standard `\bibliography`.

The `chapterbib` and `bibunits` packages separate bibliographies per chapter or per other logical units.

In the `vakthesis` bundle, the `bibset` environment is provided that is used in the following way. Let `xampl-thesis.tex` be a thesis file containing the commands for the list of referenced sources and the list of author’s publications.

```
\begin{bibset}{Список використаних джерел}
  \bibliographystyle{{bibliography style 1}}
  \bibliography{{referenced sources}}
\end{bibset}
```

```
\begin{bibset}[a]{Список публікацій автора}
  \bibliographystyle{{bibliography style 2}}
  \bibliography{{author’s publications}}
\end{bibset}
```

Standard `\cite` and `\bibliography` commands are used in the text. There is no need to tag or categorize them. Argument of any `\bibliography` command may be any list of `.bib` files.

The `bibset` environment redefines commands `\bibliographystyle` and `\bibliography` such that they write commands `\bibstyle` and `\bibdata` to the `.aux` file if they appear in the first environment `bibset` and do not write otherwise.

Then during the first run BIB<sub>T</sub>E<sub>X</sub> sees only one copy of `\bibstyle` and `\bibdata`, corresponding to the first environment `bibset`. At this point the file `xampl-thesis.bbl` generated by BIB<sub>T</sub>E<sub>X</sub> should

be renamed to `xampl-thesis1.bbl` (manually or programmatically).

During the second run the first and second `bibset` environments are interchanged, i.e., commands `\bibstyle` and `\bibdata` are written to the `.aux` file if they appear in the second `bibset` environment and are not written otherwise.

So now `BIBTEX` sees commands `\bibstyle` and `\bibdata` corresponding to the second `bibset` environment. At this point the file `xampl-thesis.bbl` should be renamed to `xampl-thesis2.bbl`.

At the last step `LATEX` includes the generated files `xampl-thesis1.bbl` and `xampl-thesis2.bbl` at the corresponding places. Hence the author has two reference lists in the thesis.

In earlier versions of the official style guides by VAK there was not any mention of the two reference lists. I just realised this for myself. However, the modern official style guide by MON contains an explicit recommendation to prepare up to three reference lists in a thesis. This solution works in modern classes too.

### 2.3 The `casus` package and UTF-8

In the summary, authors write the name of the institution where they studied and worked on their thesis. So a command `\institution` is provided by classes whose argument is the name of this institution. This name (in the nominative case) appears on the cover page, for example, Національний педагогічний університет імені М. П. Драгоманова.

At the same time, there is a sentence on the reverse side of the cover page, where the author states that this work is performed at this institution. Here the name of institution is in the locative case, for example, у Національному педагогічному університеті імені М. П. Драгоманова.

In the Ukrainian language, a noun (as well as some other parts of speech) can change its form to express its syntactic function in the sentence. There exist seven cases of a noun: nominative, genitive, dative, accusative, instrumental, locative, and vocative.

For example, `університет` is a university in Ukrainian. This is the form of nominative case. If I want to write that I study at a university, I would use the form of locative case: `я навчаюся в університеті`. That is, the ending is changing and there is also a preposition.

I think it is redundant to ask a thesis author to provide different forms of the institution name if the `\institution` command already gives a nominative form of the name. The `vakthesis` or `vakaref` class can

“compute” genitive, dative or other form. This is exactly what the auxiliary package `casus` does.

Briefly, the algorithm is the following. Suppose the institution name is Національний педагогічний університет імені М. П. Драгоманова. Here there is a special word `університет` from the list of “known words”. All words before the known word are adjectives. The `casus` package has rules to decline adjectives as well as rules to decline nouns. All words after the known word should not be changed.

The algorithm splits a given sentence into words, finds a known word (`університет`, `інститут`, `академія`, `коледж`, `міністерство`, `бібліотека`, `кафедра`, etc.), then declines adjectives and the known word (as a noun), and then stops, i.e., do not touch the words after the known word. In Ukrainian, any name of institution has this form. So this algorithm works.

Unfortunately, one day an author reported to me a problem he encountered in his thesis. He just re-encoded all `vakthesis` files and his thesis files from Windows-1251 encoding to UTF-8 encoding.

After this operation he received some mysterious error messages such as

```
Missing number, treated as zero.
```

or

```
Undefined control sequence.
```

Skipping non-essential details, the main problem is in the `casus` package.

To decline a word (in particular, an adjective `педагогічний`) the algorithm runs through the word until it finds an ending from a given list of endings. This is the ending `-ий` for this word. This ending is skipped and the other ending that corresponds to a given case is added. So the words `педагогічного`, `педагогічному` and so on are received.

This simple idea should not depend on a file encoding. However, my implementation of the algorithm does not work if a file has an UTF-8 encoding. My conjecture is that the algorithm implementation fails for characters encoded by two or more bytes.

Maybe the better solution would be to use a known stable package for string manipulation instead of my quick and dirty solution implemented in the `casus` package.

### 2.4 Incorrect checking if `hyperref` is loaded

Traditionally, theses are printed on paper and stored at physical libraries. However, `LATEX` can generate an electronic document with hyperlinks. In particular, the `hyperref` package can be used to this end. Some authors of theses want to use this possibility.

Generally speaking, it is complicated to use `vakthesis` classes and the `hyperref` package simultaneously. This can cause errors that are hard to diagnose.

In particular, `vakthesis` classes redefine some internal commands such as `\@spart`, `\@schapter`, and `\@ssect`. The number of arguments is even changed. The `hyperref` package makes its modifications carefully but cannot predict that these commands have more arguments now. As a result, sectioning commands do not work as expected.

Hence, to carefully interact with the `hyperref` package, `vakthesis` classes should check if `hyperref` is loaded, for example, by means of the internal command `\@ifpackageloaded`:

```
\@ifpackageloaded{hyperref}
  {\true branch}
  {\false branch}
```

In the *⟨true branch⟩*, classes should define versions of commands and environments that are aware of the `hyperref` package.

In particular, this approach would allow solving the above-mentioned problem with sectioning commands.

But the internal command `\@ifpackageloaded` can be used in a document preamble only. I cannot use this command, for example, inside of the `bibset` environment to check if `hyperref` is loaded. So I check if some internal command of `hyperref` is defined:

```
\@ifundefined{hyper@warn}
  {\true branch}
  {\false branch}
```

It is easy; and it works. But, after a few years, I received a bug report from a `vakthesis` user. If he uses the `bibset` environment with `hyperref` loaded, then any `\cite` command is not correctly hyperlinked.

The reason is that recent versions of `hyperref` do not define `\hyper@warn` anymore. The command `\Hy@WarningNoLine` is defined instead. Hence the *⟨true branch⟩* in that code is not executed at all.

Obviously, the quick patch is to restore the definition in the document preamble:

```
\@ifundefined{hyper@warn}
  {\let\hyper@warn\Hy@WarningNoLine}
  \relax
```

In the new version of `vakthesis`, the `hyperref` check is also fixed.

However, checking if a package is loaded inside of a command/environment is a bad thing anyway. To make more robust software, I should prepare two versions of a command/environment (with and without `hyperref`) and then choose the corresponding version.

## 2.5 Overwriting and overloading

I started to develop the `vakthesis` bundle in approximately 2003. I was a PhD student at that time. My experience with `TeX` and `LATEX` was very limited. Sure, I used `LATEX` for preparing my papers, slides, etc.; but I did not try `LATEX` programming. I should say also that Internet access was unstable and expensive at that time.

As a result, I did not find any suitable templates or `LATEX` classes for thesis typesetting compatible with Ukrainian requirements. I thought the situation required development of a `LATEX` class that I needed myself.

At some point, I decided to start from the standard `report` class and modify it according to my needs. It seemed that overwriting of an existing class is a better approach. There exists a ready-to-use class that almost complies with my requirements. I just need to patch some parts of the code that do not agree with the requirements.

However, in fact, it is not easy to maintain such a new class. First of all, I should look at `report` class development and update my code. At that time, I thought that changes of the standard `LATEX` classes are rare.

Moreover, careless overwriting may cause problems. For example, in the `report` class, the command `\part` typesets part headings on separate pages. But the `vakaref` class uses this command to typeset structural parts of summary; and they are typeset as usual headings. So I removed any `\newpage` commands and stopped.

One day a user encountered a problem when a structural part heading occurs on the last line of the page. As a result, a page break appears between the heading and the following text. This is unwanted behavior, of course, and fixing of the command is required.

This is one of the reasons why modern classes `mon2017dev` and `mon2017dev-aref` use another approach. They are designed as a “level” above the traditional classes `vakthesis` and `vakaref` respectively. They load these base classes and then redefine some commands and environments.

Unfortunately, overloading may cause problems too. Because the `casus` package (see Section 2.3) works with Cyrillic letters directly it should be loaded after the `inputenc` package. So, in the `vakaref` class, there is a corresponding line:

```
\AtBeginDocument{\usepackage{casus}}
```

Since `mon2017dev-aref` is a level above the `vakaref` class, the following line

```
\LoadClass{vakaref}
```

is in the `mon2017dev-aref`.

For the `mon2017dev-aref` class, some modifications in `casus` are needed. So this class loads a modified version of this package:

```
\AtBeginDocument{\usepackage{casus2017dev}}
```

Of course `casus2017dev` cannot work without `casus`. But we have the following chain. Firstly, the `mon2017dev-aref` class loads the `vakaref` class, then `vakaref` adds `casus` to the `\AtBeginDocument` hook. Later the `mon2017dev-aref` class adds `casus2017dev` to the `\AtBeginDocument` hook. As a result, in a proper place, `casus` is loaded, and then `casus2017dev` is loaded.

This colossus with feet of clay did its excellent work... till one day when it fell. A user is in a slight panic: declension does not work in his summary, and the cover page of the summary is full of nonsense! Oh, and he should submit his thesis and summary today! More interesting, I do not see this problem on my system.

The reason is in a new release of  $\LaTeX$ . In version 2020-10-01, a general hook management system was provided [3]. This affects standard hooks defined by the command `\AtBeginDocument` too.

I am not sure if I understand correctly what exactly happens. I suppose that, since `casus` and `casus2017dev` are loaded in the different classes, we have that they are added to hooks with different labels. As a result, either code is executed in changed order or the second `\AtBeginDocument` just executes code instead of adding to the hook. The visible result is that `casus2017dev` is loaded before `casus`. It cannot work in this situation.

It is easy to fix this problem. It is enough to add the line

```
\RequirePackage{casus}
```

to the `casus2017dev` package.

Clearly, such many-level overloading may be a real headache for users as well as for maintainers.

Taking into account the above-mentioned problems, I do not know now what is the best way to develop a new class: copy an existing class and rewrite some parts of it, or load a base class and redefine the commands/environments.

### 3 Current status

#### 3.1 Two separate modules

Now the `vakthesis` bundle consists of two separate, almost independent, modules: traditional `vakthesis` classes and modern `mon2017dev` classes.

I started development of `mon2017dev` as a separate module to keep the `vakthesis` module stable and harmless for users. In some aspects, modern

requirements by `MON` differ essentially from traditional requirements by `VAK`. Sometimes they are unclear and ambiguous. So `mon2017dev` classes are intended for clarifying development objectives and obtaining feedback from users.

However this separation causes some problems today. In particular, separate installation of two modules is more complicated for users. Documentation is also divided between two modules.

For me as the maintainer, development and maintenance of two modules also requires additional effort. For example, problems similar to the case with `casus/casus2017dev` (see Section 2.5) are mostly caused by this separation.

#### 3.2 Alternatives

Some alternative solutions for typesetting a thesis in Ukraine exist.

First of all, I would like to mention a long-standing `dissert` class by Andrew Martovlos. This class is based on the `report` class and `size14` class option and published in 2002.

Unfortunately, the website where `dissert` was initially posted does not exist anymore. Also, there exist a number of derivatives of `dissert` by now. Probably they are even mutually incompatible. One of them with the name `dissert_new` is published at the `Linux.org.ua` forum [4].

I have a copy of an `Opus` class by Andrii Semenov. Sergei Sharapov sent me this copy in 2009 and said that this class is used in Bogolyubov Institute for Theoretical Physics of the National Academy of Sciences of Ukraine. However I do not know its current status.

Another class is recently published at the above-mentioned forum too [2]. This is `ukrainethesis` class by Kostiantyn Hermash. He used it to prepare his PhD thesis.

So, users that are unsatisfied with `vakthesis`  $\LaTeX$  classes may choose another solution. Nevertheless, one considerable difference between `vakthesis` and other classes is that I actively maintain `vakthesis` and answer user questions about it.

#### 3.3 Users of `vakthesis`

Despite the fact that some unsolved problems exist and these problems can be critical for some users, there exists some interest in `vakthesis`.

Many people use these classes to typeset their doctoral theses belonging to various fields: biology, computer science, mathematics, physics, etc. Also some students adapt the classes for their student qualifying works although `vakthesis` is not intended for this task.

A friendly community of `vakthesis` users is gathered at the `Linux.org.ua` forum: [linux.org.ua/](https://linux.org.ua/). Any user can ask questions here and receive support from the community.

Recently the Doctor Barbarus Services company offers commercial support for all users that need it: [sites.google.com/view/drbarbarus/](https://sites.google.com/view/drbarbarus/).

#### 4 Future plans

I hope the following changes will resolve some complicated problems mentioned in the above sections as well as make the `vakthesis` bundle more convenient for users and developers.

##### 4.1 Combine `vakthesis` and `mon2017dev`

The modern classes `mon2017dev` are quite stable software already. So they can be combined with the base `vakthesis` classes. However, this is not just mechanical work if we want to keep compatibility.

##### 4.2 Make `vakthesis` fully UTF-8 compatible

Full rewriting of the `casus` package is the main problem in this direction. This is an important issue because the UTF-8 encoding is a standard in the modern world.

##### 4.3 Provide more documentation

In particular, the installation process should be described with more details for non-experienced users.

More useful and user-friendly example files are also needed. Some information related to real persons should be removed from the example files. For examples with traditional `vakthesis` classes, I used some parts of my thesis. The modern classes `mon2017dev` are illustrated by examples from the MON style guide that contain real information too.

##### 4.4 Upload to CTAN

I did not upload the `vakthesis` bundle to CTAN earlier because I believed that it is not yet sufficiently stable software.

However I hope that above-mentioned changes will lead `vakthesis` to be more mature and usable.

Moreover having the software available at CTAN under a free license will also make it available in the main  $\text{\TeX}$  distributions. So it will become easily installable.

##### 4.5 Use version control system

I did not use any version control system earlier. But now I understand its importance for development and maintenance.

Later I will upload code to GitLab, GitHub, or my own server (to be decided).

This may help other developers continue this project if I will not be able to maintain it for some reasons. I am in Ukraine now; so such reasons can occur any day.

## 5 Acknowledgments

I am grateful to Barbara Beeton and Karl Berry for their careful reading the manuscript and improving the presentation and English wording.

This paper was prepared in Ukraine during the russian invasion. I sincerely appreciate the Armed Forces of Ukraine's heroic efforts to defend my country.

## References

- [1] H. Adriaens, U. Kern. `xkeyval`—new developments and mechanisms in key processing. *TUGboat* 25(2):194–199, 2004. [tug.org/TUGboat/tb25-2/tb81adriaens.pdf](https://tug.org/TUGboat/tb25-2/tb81adriaens.pdf)
- [2] kostiantyn.hermash. Message at the `Linux.org.ua` forum. [linux.org.ua/index.php?topic=689.msg204167#msg204167](https://linux.org.ua/index.php?topic=689.msg204167#msg204167)
- [3]  $\text{\LaTeX}$  Project Team.  $\text{\LaTeX}2_{\epsilon}$  news 32, Oct. 2020. [www.latex-project.org/news/latex2e-news/ltnews32.pdf](https://www.latex-project.org/news/latex2e-news/ltnews32.pdf)
- [4] sampleplasma. Message at the `Linux.org.ua` forum. [linux.org.ua/index.php?topic=689.msg92080#msg92080](https://linux.org.ua/index.php?topic=689.msg92080#msg92080)
- [5]  $\text{\TeX}$  FAQ contributors. Multiple bibliographies? [texfaq.org/FAQ-multbib](https://texfaq.org/FAQ-multbib)
- [6]  $\text{\TeX}$  FAQ contributors. Separate bibliographies per chapter? [texfaq.org/FAQ-chapbib](https://texfaq.org/FAQ-chapbib)
- [7] Current homepage of the `vakthesis` project. [www.imath.kiev.ua/~baranovskyi/tex/vakthesis/](https://www.imath.kiev.ua/~baranovskyi/tex/vakthesis/)
- [8] J. Wright, C. Feuersänger. Implementing key–value input: An introduction. *TUGboat* 30(1):110–122, 2009. [tug.org/TUGboat/tb30-1/tb94wright-keyval.pdf](https://tug.org/TUGboat/tb30-1/tb94wright-keyval.pdf)

◇ Oleksandr Baranovskyi  
 Doctor Barbarus Services & Institute of  
 Mathematics of the National Academy of  
 Sciences of Ukraine  
[ombaranovskyi \(at\) gmail dot com](mailto:ombaranovskyi@gmail.com)  
<https://sites.google.com/view/drbarbarus/>

## XLingPaper's use of T<sub>E</sub>X technologies

H. Andrew Black, Hugh J. Paterson III

### Abstract

We discuss the use of T<sub>E</sub>X technologies by XLingPaper, an authoring tool for producing academically oriented publications with features required for linguistic publishing. We present the T<sub>E</sub>X modules used and the rationale for the history of XLingPaper development.

### 1 Introduction

Within the publishing industry, there are several notable products for producing complex documents in beautiful formats. T<sub>E</sub>X [26, 27] is one of the well-known publishing technologies used to meet these needs. Since 2000, XML-based technologies such as XSL-FO<sup>1</sup> or the T<sub>E</sub>XML<sup>2</sup> project [31] have been used to integrate content and compose complex documents such as textbooks and maintenance manuals. Requirements for composing these large, inter-linked documents birthed the development of tools such as XMLmind,<sup>3</sup> the <oXygen/> XML Editor,<sup>4</sup> and Xpublisher.<sup>5</sup> These tools can be used to compose content within predefined XML structures.

XLingPaper, as discussed in [7, 8, 9], seeks to provide a constrained environment in which authors of complex works dealing with language descriptions and linguistic analyses can focus on content structure independently from the styling requirements of documents. In this way the underlying design principle of XLingPaper maximizes the SGML design practice of separating content from presentation. With XLingPaper, authors can keep content structure independent from page layout information and thereby provide maximal transferability between publishing styles. The software does this while providing authors a clear structured interface for authoring content.

XLingPaper is designed to reduce friction in the process of writing, composing, and publishing linguistic papers, grammars, and books by removing common time-sinks related to inconsistent formatting (especially citations, references, and numbered

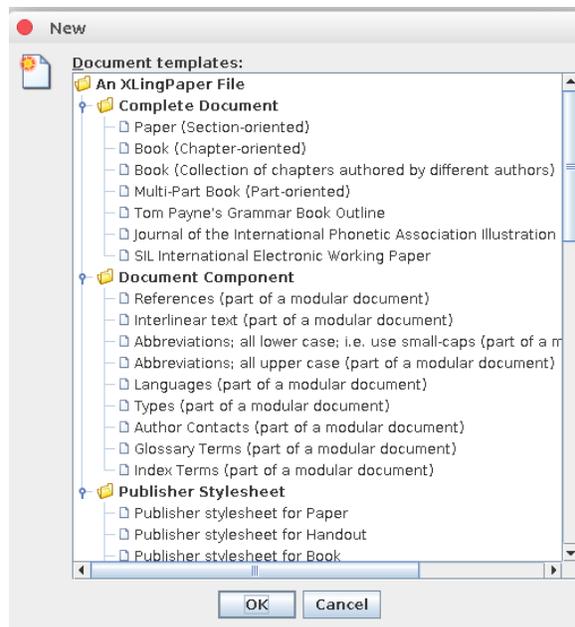


Figure 1: XLingPaper's predefined document types via DTD.

elements like examples). A full list of benefits to all parties in the publishing workflow is available [9].

The XLingPaper software has a growing number of users who have successfully typeset complex documents which, among others, include:

- master theses [28, 37, 56],
- doctoral dissertations [18, 42],
- textbooks [33],
- linguistic grammars [12],
- books [1, 43],
- journal articles [11, 38], and
- bilingual software documentation [2, 3].

### 2 What is XLingPaper?

XLingPaper<sup>6</sup> is a plug-in to the XMLmind XML Editor. XLingPaper benefits from the XMLmind XML Editor's Java-based implementation which allows it to be used on MacOSX, Windows, and Linux. Via a DTD, XLingPaper defines several document classes (articles, books, chapters, etc., as illustrated in Figure 1), in each case providing document layout sections (paragraphs, examples, endnotes, etc.). Figure 2 illustrates the main screen of the user-interface of XMLmind XML Editor. By using this interface, formatting errors are reduced because users are constrained on where in the document flow they can introduce block and line level document elements.

Editor's note: This paper was presented at TUG2021. The slides that accompanied the presentation can be found at [tug.org/tug2021/assets/pdf/Andrew-Black-slides.pdf](http://tug.org/tug2021/assets/pdf/Andrew-Black-slides.pdf), and the video at [youtube.com/watch?v=jBCJC1HI73Y](https://youtube.com/watch?v=jBCJC1HI73Y).

<sup>1</sup> [w3.org/TR/xsl11](http://w3.org/TR/xsl11)

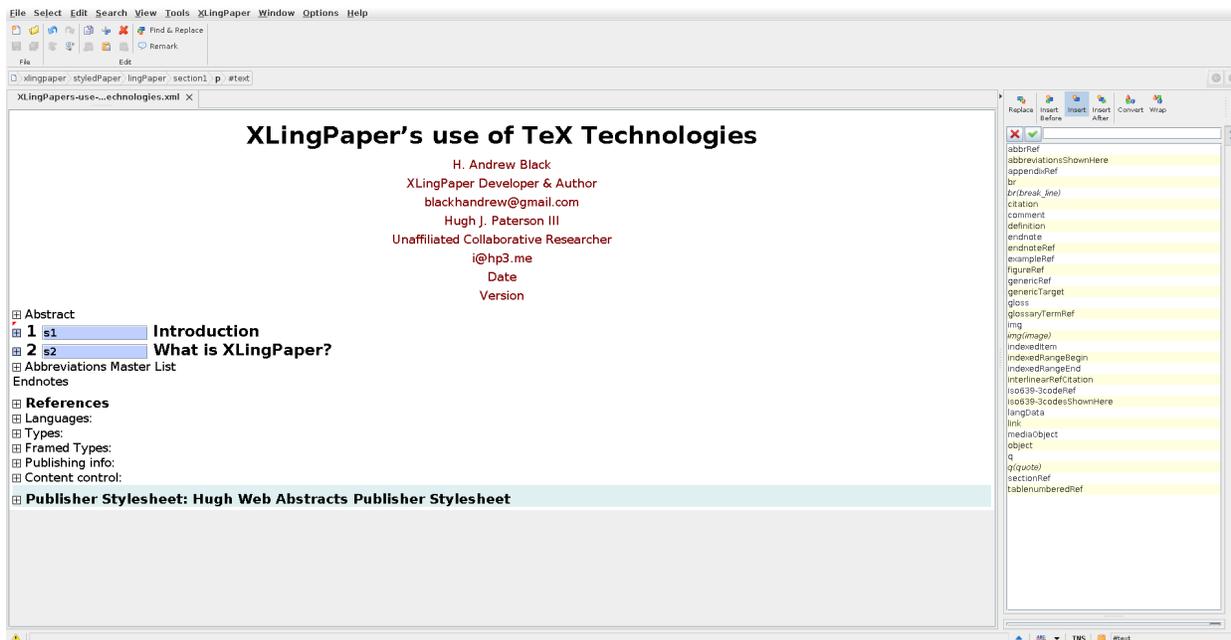
<sup>2</sup> [getfo.org/texml](http://getfo.org/texml)

<sup>3</sup> [xmlmind.com/xmleditor](http://xmlmind.com/xmleditor)

<sup>4</sup> [oxygenxml.com/xml\\_author.html](http://oxygenxml.com/xml_author.html)

<sup>5</sup> [xpublisher.com/products](http://xpublisher.com/products)

<sup>6</sup> [software.sil.org/xlingpaper](http://software.sil.org/xlingpaper)



**Figure 2:** XLingPaper’s user interface. Left side: document content editing. Right side: block and line level units available for use at the cursor location.

That is, first, authors cannot input page layout instructions directly into the document and second, the introduction of layout sections within the document flow is constrained via the DTD.

For many, the PDF format is the quintessential file format for final distribution of publishing outputs. XLingPaper supports PDF production; however, as illustrated in Figure 3, XLingPaper can produce documents in (currently) five formats, all from the same source document:

- PDF (version 1.5),
- Web pages (HTML 4),
- Microsoft Word (.doc),
- Open Office Writer Document (.odt), and
- ePUB.

XLingPaper automatically numbers tables, examples, figures, and sections. It keeps track of internal references to these entities along with citation references, abbreviations, and gloss abbreviations. This keeps numbering and reference links dependable and automated. XLingPaper also automatically generates indexes, a table or list of abbreviations used, and a section for references cited (using a custom references implementation).

Unlike most editing programs which are based on either the WYSIWYG paradigm or are unconstrained text editors such as those used to code or produce Markdown, XLingPaper (via the XMLmind XML Editor) is a structured editor much more like

the block editors we see in tools like MailChimp<sup>7</sup> or WordPress’s Gutenberg editor,<sup>8</sup> albeit without the drag-and-drop features. Rather than visually structuring the document to look the way it is to be formatted, the author “marks up” the items in the document according to their kind. One of the many benefits of using a DTD is that there is a “grammar” of what a well-formed linguistic document looks like. This makes moving, replacing, switching, or reordering sections, chapters, tables, figures, and examples less error prone because it prevents users from inadvertently creating ill-formed documents. The following sections of this paper discuss the  $\text{\TeX}$  technologies used by XLingPaper.

### 3 XLingPaper and $\text{\TeX}$

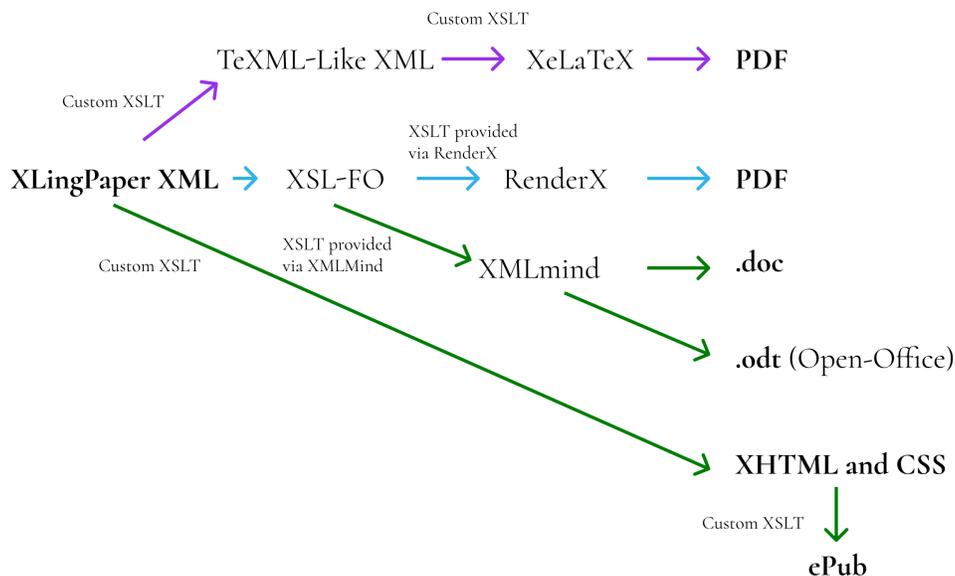
Linguistic publishing has unique requirements when compared to general publishing. The following sections provide more detail on the linguistic publishing context, design requirements and  $\text{\LaTeX}$  packages used by XLingPaper.

#### 3.1 $\text{\TeX}$ and linguistic document production

$\text{\TeX}$  has long been embraced by linguists. Peter [40] writes of a personal communication with Don Knuth where Knuth suggests that linguists were some of the earliest adopters outside of mathematicians. Thiele

<sup>7</sup> mailchimp.com

<sup>8</sup> developer.wordpress.org/block-editor



**Figure 3:** XLPaper’s data processing pipeline to multiple formats.

[51] in an interview given in 2007 states that she was typesetting linguistic journals via  $\text{\TeX}$  in 1983 — a date prior to the formal publication of Knuth’s book on using the  $\text{\TeX}$  typesetting system [26]. Thiele [50] gives an early overview of  $\text{\TeX}$  use in linguistics with mention of significant repositories outside of CTAN. A slightly more recent (2004) update by Peter [40] provides some additional tips and tools for typesetting common information structures in linguistic publishing.

The  $\text{\TeX}$  community has produced many packages which have shaped the visual face of publishing in linguistics, including `tipa`<sup>9</sup> by Rei [44], which provided access to an excellent typeface for phonetic transcriptions, and `pst-asr`<sup>10</sup> by Frampton [19] for autosegmental representations. Some packages used in linguistic publishing are special purpose but are not exclusive to linguistics. For example, Donnelly [17] describes how to use various packages to draw phonetic pitch traces using  $\text{\TeX}$ . Peter [40] and Thiele [50] list and review (through 2004) various packages across several areas of linguistics. Among others, they discuss several packages used to draw syntax trees such as `qtree`<sup>11</sup> and `forest`<sup>12</sup> and specialized packages for presenting examples and interlinear glossed texts such as `expex`.<sup>13</sup> Their re-

views also discuss packages such as `covington`<sup>14</sup> and `gb4e`<sup>15</sup> whose collections of macros serve a variety of page layout functions targeted at publishing in linguistic topics.

The CTAN repository currently lists fifty-four different  $\text{\TeX}$  packages for linguistic typesetting,<sup>16</sup> though some of these packages also include capabilities targeted as multi-lingual or multi-script publications or are specific style sheet implementations for publications at linguistic programs within institutions of higher education (there may be more packages which are not tagged but should be). Several of the packages tagged “linguistic” pre-date Unicode [52] but still see significant use. Sometimes it is the case that secondary packages are developed in an attempt to “fix” publishing outputs in different ways, to bring Unicode features along with the features of the original package. For example, `tipa` is not Unicode compatible, but the packages `unitipa`<sup>17</sup> and `tipauni`<sup>18</sup> seek to address different implications of not publishing with Unicode while giving access to the beautiful typeface of `tipa`. Understanding the long history of publishing and the packages’ interdependencies (including the order of loading packages) constitute barriers of adoption to new  $\text{\TeX}$  users.

We discuss  $\text{\TeX}$  barriers of adoption for two reasons. First, it exemplifies some of the complexities

<sup>9</sup> [ctan.org/pkg/tipa](http://ctan.org/pkg/tipa)

<sup>10</sup> [ctan.org/pkg/pst-asr](http://ctan.org/pkg/pst-asr)

<sup>11</sup> [ctan.org/pkg/qtree](http://ctan.org/pkg/qtree)

<sup>12</sup> [ctan.org/pkg/forest](http://ctan.org/pkg/forest)

<sup>13</sup> [ctan.org/pkg/expex](http://ctan.org/pkg/expex)

<sup>14</sup> [ctan.org/pkg/covington](http://ctan.org/pkg/covington)

<sup>15</sup> [ctan.org/pkg/gb4e](http://ctan.org/pkg/gb4e)

<sup>16</sup> [ctan.org/topic/linguistic](http://ctan.org/topic/linguistic)

<sup>17</sup> [ctan.org/pkg/unitipa](http://ctan.org/pkg/unitipa)

<sup>18</sup> [ctan.org/pkg/tipauni](http://ctan.org/pkg/tipauni)

that XLingPaper seeks to simplify, as it presents authors not just a visual environment for document composition, but also a cohesive output solution. Second, it speaks to the software design process in finding the minimal viable product. That is, *how much (or little) of a software stack is needed to make a usable software product for linguistic publishing?*

The  $\text{T}_{\text{E}}\text{X}$  community is divided on this. While the diagrams in linguistic books and journals since the 1980s exemplify many beautiful, sharp, crisp, illustrations created directly in  $\text{T}_{\text{E}}\text{X}$ , many trainers of  $\text{T}_{\text{E}}\text{X}$  tools,<sup>19</sup> but not all,<sup>20</sup> have steered authors towards a more generic set of packages which do not include specific diagram-creating macros. Rather, they suggest that authors use secondary illustration tools to generate illustrations and then include them as vector PDFs or images.

This second method is the document production path that the XLingPaper philosophy follows. That is, XLingPaper reduces the complexity of the typesetting task for authors by requiring complex visualizations to be produced via graphical tools. We have found tools like Figma<sup>21</sup> and Inkscape<sup>22</sup> very helpful in the graphic production task. The XLingPaper product seeks to: lower barriers of entry, only produce valid documents, and keep the code base to a minimum.

As mentioned in the discussion of `tipa`, linguistic documents were typeset by  $\text{T}_{\text{E}}\text{X}$  before Unicode existed. Unicode was introduced in 1991 and by the early 2000s Unicode along with document and data storage in XML formats were being heralded in academic linguistics as a best practice in order to avoid vendor lock-in, increase interoperability across use cases, and to separate data life-cycles from encoding or software life-cycles [5, 6, 53]. Due to the heavy reliance on Unicode by today’s practitioners of language documentation and linguistic work, XLingPaper specifically uses  $\text{X}_{\text{L}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  and compatible packages to produce PDF outputs. This brings continuity to the text input process for users across their workflows. It also makes importing and using language or phonetically transcribed examples simpler by removing the need to use macros to derive characters.

<sup>19</sup> Among others, see the Linguistics Dissertation guides for the University of Hawai’i at Mānoa [22], the University of Pennsylvania [16], and Language Science Press Guidelines [36].

<sup>20</sup> For counterexamples see [21, 30, 41, 48].

<sup>21</sup> [figma.com](https://figma.com)

<sup>22</sup> [inkscape.org](https://inkscape.org)

### 3.2 Design desiderata for XLingPaper outputs via $\text{T}_{\text{E}}\text{X}$

Three goals have driven the development of XLingPaper:

- separation of content and style,
- software accessibility (license and size), and
- beautiful multi-format outputs.

Deciding how  $\text{T}_{\text{E}}\text{X}$  technologies fit within the project has been a journey. The development of XLingPaper started in 2001 without any use of  $\text{T}_{\text{E}}\text{X}$  technologies. In 2006, XLingPaper added XSL-FO for PDF production. Prior to 2009, XLingPaper used RenderX’s XEP<sup>23</sup> product to produce PDF documents. As far as we know, there are two cross-platform XSL-FO processors written in Java: RenderX’s XEP application and the Apache FOP project.<sup>24</sup> Using a Java implementation reduces the size of the required stack because the XMLmind XML Editor requires Java.

XSL-FO processors can have various degrees of implementation of the XSL-FO standard. RenderX has some limitations which affect page layout but has more complete coverage than the Apache FOP project which lacks certain required table-oriented capabilities.<sup>25</sup> The limitations of RenderX are discussed in Section 5. In 2009 plans were made to add  $\text{X}_{\text{L}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ -based output to XLingPaper because, while there was a free version of RenderX, the output contained a watermark. By implementing the ability to export to PDF via  $\text{X}_{\text{L}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ , watermarks could be avoided. The  $\text{X}_{\text{L}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  method of PDF production is now the default method to produce PDF documents, although the RenderX method is still possible.

Maintaining a separation of content and style in the XLingPaper environment was a key design requirement. When the  $\text{X}_{\text{L}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  method of PDF production was introduced, XLingPaper already had a way to format output per a user-created publisher style sheet — allowing great flexibility due to the separation of style and content. Using  $\text{T}_{\text{E}}\text{X}$  technologies meant the developer (Andrew Black) needed to be able to map from an XLingPaper publisher style sheet to  $\text{X}_{\text{L}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ . It was known that  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  was the ideal  $\text{T}_{\text{E}}\text{X}$  implementation to target. However, pure  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  came with predefined output formatting for front matter, chapters, sections, back matter, etc. Pure  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ , then, would not allow direct control of formatting of all of these per an XLingPaper user-defined publisher style sheet. This required overriding these standard features of  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  with a custom implementation of the  $\text{T}_{\text{E}}\text{X}$  commands needed to control

<sup>23</sup> [renderx.com/tools/xep.html](https://renderx.com/tools/xep.html)

<sup>24</sup> [xmlgraphics.apache.org/fop/index.html](https://xmlgraphics.apache.org/fop/index.html)

<sup>25</sup> [xmlgraphics.apache.org/fop/compliance.html](https://xmlgraphics.apache.org/fop/compliance.html)

formatting. XLingPaper takes a custom approach in implementing flexibility here. Table 1 lists the custom commands implemented.

The programmer of XLingPaper recently discovered `memoir`<sup>26</sup> [54, 55]. As a package, `memoir` accomplishes many of the same tasks and could be considered to replace some of the custom code if it were shown to be easy to implement and that the size of the total XLingPaper code base would be reduced.

The distributability of the software was also seen as a design requirement. Distributability is understood to have two components: license and accessibility, including size.

From the outset, XLingPaper was designed to be costless to the end user. It is licensed under the MIT license, and its code is currently available on Github.<sup>27</sup> The XMLmind XML Editor had a costless Personal Use License that met the distributability goal for the majority of the initial target audience of XLingPaper. The few XLingPaper users who did not meet the terms of that license most likely would be able to afford to purchase (or have their organization purchase) a professional license of the XMLmind XML Editor. The XLingPaper plug-in has always been free.

The software size of XLingPaper is a major design influencer. Many of the expected users of XLingPaper live and work in places around the world where Internet connections are characterized by high costs, low bandwidth capacity, and general unavailability. Therefore, the download required to install XLingPaper needed to be as small as possible. On Windows the current full XLingPaper installer is 146 MB, and the XMLmind XML Editor installer is 116 MB. Both are required. This stands in contrast to the  $\TeX$  Live 2010 installer which has a size of about 1.2 GB when downloaded and 2.38 GB when uncompressed. The size constraint impacts XLingPaper because its distribution must be independent of larger mainstream  $\TeX$  distribution solutions which have a large footprint. This, of course, includes  $\TeX$  Live. Therefore the developer identified which  $\LaTeX$  packages and binaries were needed and created a custom installation package which met the required specifications. In keeping with limiting the installation size, XLingPaper still uses  $\TeX$  Live 2010, although there is now an option to use  $\TeX$  Live 2020, especially for those running on Mac OS X.<sup>28</sup>

<sup>26</sup> [ctan.org/pkg/memoir](https://ctan.org/pkg/memoir)

<sup>27</sup> [github.com/sillsdev/XLingPap](https://github.com/sillsdev/XLingPap)

<sup>28</sup> See ([software.sil.org/xlingpaper/xelatex-package-from-tex-live-2020/](https://software.sil.org/xlingpaper/xelatex-package-from-tex-live-2020/)) for instructions on how to use  $\TeX$  Live 2020 with XLingPaper.

XLingPaper currently uses the following  $\LaTeX$  packages (in alphabetical order):

<code>attachfile2</code>	<code>lineno</code>
<code>booktabs</code>	<code>longtable</code>
<code>calc</code>	<code>lscap</code>
<code>color</code>	<code>mdframed</code>
<code>colortbl</code>	<code>multirow</code>
<code>etoolbox</code>	<code>polyglossia</code>
<code>fancyhdr</code>	<code>setspace</code>
<code>fontspec</code>	<code>tabularx</code>
<code>footmisc</code>	<code>ulem</code>
<code>hyperref</code>	<code>xltxtra</code>

The twenty  $\LaTeX$  packages that are part of the custom XLingPaper distribution are still rather large for someone for whom Internet bandwidth is an expensive and inconsistent commodity.

To reduce bandwidth requirements two assumptions were made which have more or less proven to obtain. The first assumption was that the twenty packages and binaries would not need to change over time; in contrast, the second assumption was that XLingPaper would acquire new features and need bug fixes. These assumptions resulted in an architecture where page layout information expressed in XML is translated via custom  $\TeX$  commands to either  $\TeX$  directly or to commands understood by  $\LaTeX$  packages distributed with XLingPaper. This abstraction layer was then executed when the  $X_{\LaTeX}$  file was processed.

This middle layer has granted XLingPaper flexibility in adding new code and capabilities while keeping the “heavy”  $\LaTeX$  packages stable. The net result is a “heavy” first install package (116 MB), but light-weight upgrade packages (6.21 MB). In the thirteen-year history of development, there have been a few occasions where upgrades have required the download of new “heavy” packages. One such case was when the ability to use framed units was added. These elements depend on the `mdframed`<sup>29</sup> package [14]. The architecture separating stable packages from custom code, however, has generally worked out well and kept update sizes low.

### 3.3 PDF production

We know of two existing pathways for converting XML content into PDFs. The first is via XSL-FO, and the second is via  $\TeX$ ML which converts XML content to  $\TeX$ -formatted documents for further processing to PDF.

Given certain limitations in both XSL-FO and  $\TeX$ ML, XLingPaper uses a custom (or third) method. When an author instructs XLingPaper to produce

<sup>29</sup> [ctan.org/pkg/mdframed](https://ctan.org/pkg/mdframed)

**Table 1:** Custom commands used by XLingPaper

Command for	Purpose
Table of contents	Store and retrieve page numbers; format the contents.
Lists	Numbered and bulleted lists with control over indents, etc.
Examples	Example number and example content, where the content can be a line, a list of lines, a set of words, a list of a set of words, interlinear, a list of interlinears, etc.
Indexes	Handle keeping track of XLingPaper’s indexing capability, including page numbers.
Interlinears	Handle lines in an interlinear text or example, including dealing with an ISO 639-3 code in an interlinear example.
Block quotes	Handle special cases needed for block quotes.
Table headers	Attempt to calculate a column’s width via its contents.

PDF output via  $X_{\text{L}}^{\text{L}}\text{A}^{\text{T}}\text{E}^{\text{X}}$ , XLingPaper produces a  $\text{T}_{\text{E}}\text{X}$ ML-like XML file. This is then converted into a  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ -formatted document via a set of XSLT transforms and processed via  $X_{\text{L}}^{\text{L}}\text{A}^{\text{T}}\text{E}^{\text{X}}$  to produce the PDF. Figure 3 contains a diagram of the data handling process.

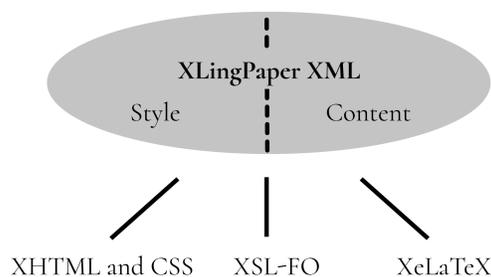
### 3.4 $\text{T}_{\text{E}}\text{X}$ ML

$\text{T}_{\text{E}}\text{X}$ ML was discovered in the process of planning for the transition of the default PDF renderer from RenderX’s XEP to  $X_{\text{L}}^{\text{L}}\text{A}^{\text{T}}\text{E}^{\text{X}}$ . Initial analysis conducted in 2009 understood  $\text{T}_{\text{E}}\text{X}$ ML to have two infelicities for use-cases required in linguistic publishing with XLingPaper:

1.  $\text{T}_{\text{E}}\text{X}$ ML has Python as a dependency and the XLingPaper developer did not want to require XLingPaper users to install a version of Python specifically for  $\text{T}_{\text{E}}\text{X}$ ML. The use of Python raised two concerning issues. First, potential conflicts with other installed versions of Python; and second, an increase in the required download size due to the inclusion of Python.
2. XLingPaper users require a high degree of control of white space. The fine grain control of whitespace was not immediately clear how to accomplish with  $\text{T}_{\text{E}}\text{X}$ ML.

### 3.5 Control characters

Even with the use of Unicode in the text of documents, there are some features of typesetting with  $\text{T}_{\text{E}}\text{X}$ -based implementations which require the use of control characters. Additionally, XML also has control characters. In  $\text{T}_{\text{E}}\text{X}$  these include [, ], <, and >. When transforming data between XML and  $\text{T}_{\text{E}}\text{X}$ ,  $\text{T}_{\text{E}}\text{X}$  control characters and commands need to be escaped to ensure proper data processing. This has been implemented via Java since Java was already present in the dependency stack due to the XMLmind XML Editor requiring it. Additionally,



**Figure 4:** XLingPaper combines style and content information contained in its custom XML and then exports it into three different formats for further processing.

some small methods have been written in Java to provide additional access to features via the graphical user interface. Among other things, these include adding rows and/or columns to tables, automatically converting glosses to abbreviation references, and importing references from various XML formats.

### 3.6 Ling- $\text{T}_{\text{E}}\text{X}$

One might ask, “Why not add more linguistic-related  $\text{T}_{\text{E}}\text{X}$  packages to the available stack, or use those instead of creating custom code?” The answer has two simple parts: First, in 2009 the linguistic capabilities of  $\text{T}_{\text{E}}\text{X}$  packages were different than they are today. Second, XLingPaper is more than a  $\text{T}_{\text{E}}\text{X}$  document producer. For example, some authors [2], [3] use XLingPaper to manage multilingual content on websites.

Besides  $\text{T}_{\text{E}}\text{X}$ , XLingPaper also produces XSL-FO and XHTML/CSS outputs. When new features are

considered for inclusion, they must be considered for all output formats.

After excluding  $\text{\TeX}$ ML as a viable option, and still seeking to create  $\text{\XeLaTeX}$ -based output, a solution was needed to determine which set of minimal  $\text{\TeX}$  packages would be needed. The Ling- $\text{\TeX}$  group,<sup>30</sup> which also ran the Ling- $\text{\TeX}$  mailing list from 1995–2018, was discovered.<sup>31</sup> Ling- $\text{\TeX}$  seemed to be the locus of activity in linguistic typesetting via  $\text{\TeX}$  even though other web pages discussing linguistics and  $\text{\TeX}$  also existed, e.g., Essex<sup>32</sup> and UPenn.<sup>33</sup> Today, now that the mailing list is no longer in operation, many of the mailing list participants can be found interacting on the  $\text{\TeX}$  stackexchange.<sup>34</sup>

State-of-the-art for  $\text{\TeX}$ -based linguistic publishing in 2009, as recommended by the Ling- $\text{\TeX}$  website, suggested using `covington` and `ling-mac`—the list of macros discussed by Thiele in [50]. These macros were used to solve similar use cases, among others, to those already implemented by X $\text{\LingPaper}$ . Their approaches and outputs, however, had more limitations than what X $\text{\LingPaper}$  already offered. X $\text{\LingPaper}$  had the following capabilities for typesetting interlinears:

- no limits on the number of lines within an interlinear grouping;
- no limits on the number of free translation and literal translation lines;
- the ability to include a source reference within the interlinear; and especially
- the ability to tag interlinear items with an ISO 639-3 code for the language used in the interlinear.

At the time the best solution given the state of the  $\text{\TeX}$  packages available was custom  $\text{\TeX}$  scripts, although now similar features may be possible via other packages. For example in 2019 Pellard [39] discussed the limiting approaches in various  $\text{\TeX}$  packages related to interlinear glosses and his solution `typgloss`.<sup>35</sup> X $\text{\LingPaper}$ 's examples can be seen in Figures 6–7 which contain output illustrating some of the special capabilities X $\text{\LingPaper}$  offers.

<sup>30</sup> [web.archive.org/web/20150702123633/http://heim.ifi.uio.no/~dag/ling-tex](http://web.archive.org/web/20150702123633/http://heim.ifi.uio.no/~dag/ling-tex)

<sup>31</sup> [ling-tex.ifi.uio.narkive.com](http://ling-tex.ifi.uio.narkive.com)

<sup>32</sup> [essex.ac.uk/linguistics/external/clmt/latex4ling](http://essex.ac.uk/linguistics/external/clmt/latex4ling)

<sup>33</sup> [ling.upenn.edu/advice/latex.html](http://ling.upenn.edu/advice/latex.html)

<sup>34</sup> [tex.stackexchange.com](http://tex.stackexchange.com)

<sup>35</sup> [github.com/tpellard/typgloss](https://github.com/tpellard/typgloss)

## 4 Typesetting tasks X $\text{\LingPaper}$ users often encounter

Linguistic documents have several formatting needs that other kinds of documents do not. This section discusses some of them.

### 4.1 Numbered example layouts

Linguistic documents usually have many numbered examples. The prose often refers to examples which are typographically nearby or to previous examples. X $\text{\LingPaper}$  automatically keeps track of the example identifiers. This is especially important in linguistic publishing because authors, and publishing style sheets, often make use of different kinds of examples, including sub-examples, and table-like design layouts which can contain lists of words along with their glosses (as shown in Figure 5) and interlinear clauses (as shown in Figure 6). Some cases even have headings in portions of the example.

- (9)
- |    |           |           |              |
|----|-----------|-----------|--------------|
| a. | ko-sis-o  | [– – –]   | move forward |
| b. | kɔ-kij-a  | [– – –]   | act          |
| c. | ko-bund-o | [– – –]   | break        |
| d. | kɔ-bʊt-a  | [– – –]   | become long  |
| e. | ko-bɛŋ-o  | [– – –]   | wink         |
| f. | kɔ-kɛk-a  | [– – –]   | decorate     |
| g. | ko-sok-o  | [– – –]   | cackle       |
| h. | kɔ-mvɔɖ-a | [– – –]   | suck         |
| i. | kɔ-bab-a  | [ – – ↘ ] | carry        |

Figure 5: List of words as seen in [42]

### 4.2 Interlinear glossed texts

There is a long tradition within linguistics and language study of presenting phrases containing different languages (but the same content) as interlinear texts. Di Biase-Dyson et al. [15] trace the practice back as far as the 1652 publication of Kircher [25]. More recent publications display significant variation in page layout related to interlinear glossed texts and interlinear examples. Variation exists in three dimensions:

- content elements,
- data-structure of the encoded elements, and
- page layout (visual display of the elements).

A full demonstration of the variation in content and its positioning across common style sheets in linguistics is beyond the scope of this paper. Significant variations include the presence or absence of the following elements:

- index elements such as example numbers or sub-numbers (as shown in Figure 6),

Una frase cuantificadora puede acompañar al sustantivo (véanse [Los Cuantificadores](#) y [Los Números Cardinales](#)). Cuando se presenta esta frase, siempre va delante del núcleo de la frase nominal, como en los ejemplos en (2).

- (2) a. [tcf- Náa majñuu nákhü iduu iya'  
 Zila] náā māhjūù" nákù idūū ijā?  
 LOC entre TOT.cuatro ojo.3SG agua  
 'De entre los cuatro manantiales'<sup>5</sup> [Smajiin:6]
- b. [tpl- Gí'doo witsu rakhóó mikhúdí  
 Tlac] EST.tener.3SG cinco nariz.3SG (EST).picud@  
 'Tiene cinco esquinas picudas' [FC:5.1]

El cuantificador puede presentarse en construcciones donde no hay sustantivo expreso, como se explica en [Los Cuantificadores](#). Un ejemplo se incluye aquí.

**Figure 6:** Interlinear example from [32]. Note the example numbers on the left followed by example groups (a) and (b). Each interlinear then also has a language indicator in square brackets. Customization allows for as many rows per group as is required. Finally, on the right the hyperlinked citation to the reference for the source text is indicated.

- headings to the interlinear,
- speaker indicator,
- language indicator,
- citation indicator pointing to the larger text from which the example element is taken (see Figure 6 for an example), and
- limits on the number of rows in the original, gloss, translation, and free translation tiers.

Existing T<sub>E</sub>X packages approach these content requirements in different ways. As far as we can tell, the following commonly used packages for interlinear glossing all have limitations to some degree. The `expex` package does not offer a content solution for the language code or the citation. The package `langsci-gb4e`,<sup>36</sup> a fork of `gb4e`, supports the *Leipzig Glossing Rules*,<sup>37</sup> a commonly adopted set of linguistic typesetting conventions. While the Leipzig Glossing Rules do call for the language name or identifier to appear on the right hand side of the interlinear glossed text, it does not have a place for the citation. The package `linguex` does not have either language or citation content places built in. With these considerations, it was clear in 2009 that X<sub>L</sub>ingPaper offered more to authors than any single package in the T<sub>E</sub>X ecosystem. In order to implement existing X<sub>L</sub>ingPaper features, it meant creating custom T<sub>E</sub>X scripts to implement interlinear texts.

There are also some reasons related to data structure for considering X<sub>L</sub>ingPaper over alternatives. Interlinear glossed texts are often stored in one of a few formats: ELAN files,<sup>38</sup> FLE<sub>X</sub> Text files,<sup>39</sup> Standard Format files,<sup>40</sup> L<sup>A</sup>T<sub>E</sub>X files,<sup>41</sup> custom project-specific XML files, or relational databases such as MySQL, PostgreSQL, or FileMakerPro. Moving content from analysis and markup tools to typesetting tools is an ever-present need for linguists. Several tools such as ELAN and FLE<sub>X</sub> have well-established workflows for data transfer [45]. FLE<sub>X</sub> is often considered the tool of choice for many field linguists, language documenters, and lexicographers. For many linguists entering the field, it is the tool of choice over older tools like Toolbox (which uses Standard Format files) due to built-in collaborative features and grammar parsers [4]. Interlinear text in FLE<sub>X</sub> can be exported via XML and the data used within X<sub>L</sub>ingPaper documents. This presents FLE<sub>X</sub> users the opportunity to typeset their texts rather easily. Enabled by XML's modular document referencing features, X<sub>L</sub>ingPaper documents can reference components. Using the XML document referencing strategy with XML-encoded FLE<sub>X</sub> texts allows authors to reflow typesetting outputs easily if they make content changes in their FLE<sub>X</sub> environment.

<sup>36</sup> [ctan.org/pkg/langsci](http://ctan.org/pkg/langsci)

<sup>37</sup> [eva.mpg.de/lingua/pdf/Glossing-Rules.pdf](http://eva.mpg.de/lingua/pdf/Glossing-Rules.pdf)

<sup>38</sup> [archive.mpi.nl/tla/elan](http://archive.mpi.nl/tla/elan)

<sup>39</sup> [software.sil.org/fieldworks](http://software.sil.org/fieldworks)

<sup>40</sup> [software.sil.org/toolbox](http://software.sil.org/toolbox)

<sup>41</sup> For examples see [46] and [49].

XLingPaper does not have a direct ELAN import process. However, we have had reports of linguists using the FLEx-XLingPaper publication pathway to typeset ELAN texts in L<sup>A</sup>T<sub>E</sub>X documents. One user reports capturing the X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X document prior to rendering and then copying the relevant T<sub>E</sub>X sections to their primary document and adding any required packages required by XLingPaper to the header of their primary T<sub>E</sub>X document.

Finally, there is the matter of page layout. The main types of variation in page layout we have seen include the grouping of lines into sets or subsets (see Figure 6 for example), the labeling of sets and subsets, wrapping of interlinear glosses across lines (recall that these may themselves include three or more lines), and the alignment of the various elements of content within the interlinear glosses. We have seen word and morpheme aligned interlinears. XLingPaper automatically wraps interlinears which makes the author’s job much easier. Figure 7 demonstrates the wrapping of interlinear glossed texts. It does so by formatting each aligned word in an `hbox` and then having X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X put them together in a hanging indent paragraph. This is based on the work of Kew & McConnel 1990 [24]. Similar examples can be seen in [34] and [35], among others.

### 4.3 Gloss abbreviations

Linguists use two types of abbreviations. First, they might use abbreviations for names, titles, or commonly used words. This is much like standard publishing. The second way that linguists use abbreviations is to indicate the grammatical meaning of pieces of words (morphemes). This second usage is often referred to as ‘glossing’ with the abbreviations referred to as ‘glosses’. One common set of glosses is the *Leipzig Glosses*. Leipzig Glosses, however, are not universally used for several reasons, including:

- some authors have established their own tradition within their works which they started prior to the release of the Leipzig Glosses,<sup>42</sup>
- the typeset examples are quoted from a database which does not use Leipzig Glosses,
- they are not comprehensive, and
- they are not theoretically sufficient for some linguists.

XLingPaper supports both types. XLingPaper approaches this by providing built-in access to Leipzig Glosses, but also allowing the author to fine-tune a set of abbreviations and their definitions. When

<sup>42</sup> For examples of the variation and scope of coverage consider the works of Greville Corbett, William Croft, Denis Creissels, and Martin Haspelmath.

producing the output, XLingPaper creates hyperlinks between the abbreviation and its definition. This allows readers to quickly find the meaning of glosses and for the automatic generation of a table or list of abbreviations used.

### 4.4 Bibliographies

For better or worse XLingPaper has rolled its own bibliography solution. Import options are provided for MODS<sup>43</sup> and EndNote XML formats. This enables users to import from tools like EndNote,<sup>44</sup> Zotero,<sup>45</sup> and JabRef.<sup>46</sup> XLingPaper uses custom T<sub>E</sub>X scripts to output T<sub>E</sub>X code for final rendering. It does not rely on BIB<sub>T</sub>E<sub>X</sub> or BIB<sub>L</sub>A<sub>T</sub>E<sub>X</sub>. Figure 8 shows an example of a bibliography created with XLingPaper.

### 5 Outputs L<sup>A</sup>T<sub>E</sub>X allows that others do not

While XLingPaper has a large array of linguistically-oriented formatting capabilities across all output formats, there are some that only the X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X output can produce. This is, of course, due to the formatting power of T<sub>E</sub>X and X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X.

#### 5.1 Automatically wrapping interlinears

One of the most popular features of XLingPaper is its ability to automatically wrap long interlinear examples and lines in interlinear texts. As seen in Figure 7, wrapping occurs for the glossed text tiers and free translation tiers.

For the RenderX output, the interlinear examples do not wrap; they run off to the right, which can mean completely off the page. To fix this, the XLingPaper user must break the interlinear into smaller units by hand.<sup>47</sup>

#### 5.2 Font rendering

X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X renders fonts extremely well. We show three cases where XSL-FO (via RenderX) and/or XHTML outputs have text rendering issues while X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X does not.

First, when a line of text contains material rendered in different fonts on the same line, the two fonts may not line up evenly in the vertical direction. See Figure 9. This mismatch is due to the two fonts having different ascender and descender values. In order to overcome this when using XSL-FO, one has to add custom commands to deal with the font that differs from the primary font.

<sup>43</sup> [loc.gov/standards/mods/](http://loc.gov/standards/mods/)

<sup>44</sup> [endnote.com](http://endnote.com)

<sup>45</sup> [zotero.org](http://zotero.org)

<sup>46</sup> [jabref.org](http://jabref.org)

<sup>47</sup> We note that current CSS technologies enable wrapping for XHTML output, but XLingPaper does not employ it at the moment.

## DN-1:10

आऊर	जानू	बाजा	मन	धरला,	हून मन	के	बाजाला	आऊर
aur	džanu	badža	mən	dʰərɻla	hun mən	ke	badžala	aur
CONJ	PRT	N	PRT	V	PPRON	CASE	V	CONJ
and	focus	drum	=PL	take hold-3P.PTC	they	GOL	play drum-3P.PTC	and
जानू	गूलाए	बूलाला	हून	सड़क	सड़क	आऊर	फेर	जानू
džanu	gulae	bulala	hun	səɾək	səɾək	aur	pʰer	džanu
PRT	ADJ	IT	DEM	N	N	CONJ	ADV	PRT
focus	everywhere-EMP	walk around-CAUS-3P.PTC	that	road	road	and	again	focus
हाई	ईस्कूल	ने	ईलू					
hai	iskul	ne	ilu					
N		POSTP	V					
high school	=LOC	come-1P.PTC						

And they took hold of drums and they played the drums and made us walk all over the place, on the roads, and later we came to the high school.

Figure 7: Wrapped interlinear text as seen in [57].

- Chao, Yuen Ren. 1930. ə sistim əv "toun-letaz" [A system of "tone-letters"]. *Le Maître Phonétique (Troisième Série du Le Maître Phonétique)* 30. 24–27.
- 赵元任 [Chao, Yuen-Ren]. 1980. 一套标调的字母 (英文). *方言* 1980(2). 81–83.
- Chelliah, Shobhana Lakshmi, Willem Joseph de Reuse. 2011. *Handbook of descriptive linguistic fieldwork*. Dordrecht, Netherlands; New York: Springer. doi:10.1007/978-90-481-9026-3
- Chen, Yiya & Carlos Gussenhoven. 2015. Shanghai Chinese. *Journal of the International Phonetic Association* 45(3). 321–337. doi:10.1017/S0025100315000043
- Cheung, Kwan-hin [張群顯]. 2016. Chao Tone Letters: Original theory Versus Current Practice. In 錢志安, 郭必之 and 鄒嘉彥, *Commemorative Essays for Professor Yuen Ren Chao: Father of Modern Chinese Linguistics* 現代漢語語言學之父——趙元任先生紀念論文集, 65–76. 臺北市 [Taipei City]: 文鶴出版有限公司 [Crane Publishing Company].

Figure 8: An XLingPaper bibliography demonstrating mixed Latin and Chinese scripts.

Second, the RenderX way of producing PDF cannot handle stacked diacritics, but the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X way does it very well. See Figure 10.

Third, X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X can even handle special features requiring Graphite<sup>48</sup> processing. Graphite is a multi-part technology which includes a rendering engine and a rule-based grammar which compiles against a TrueType font and effectively extends the font allowing for additional glyph selection and context shaping [13, 23, 47]. Figure 11 illustrates the special font handling needed for the Awami Nastaliq font. Of the four output renderings, the only one which renders correctly is the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X implementing Graphite.<sup>49</sup>

<sup>48</sup> [graphite.sil.org](http://graphite.sil.org)

<sup>49</sup> One must use XLingPaper's X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X package from T<sub>E</sub>X Live 2020 ([software.sil.org/xlingpaper/xelatex-package-from-tex-live-2020/](http://software.sil.org/xlingpaper/xelatex-package-from-tex-live-2020/)) for this particular

### 5.3 Hyphenation for non-English languages

Since we use the polyglossia package, one can write an XLingPaper document in any of the sixty-one non-English languages listed in the polyglossia documentation and indicate the language code for this language in a document-wide attribute. XLingPaper passes this information to X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X which will hyphenate according to that language's hyphenation rules.

### 5.4 Author contact information

XLingPaper allows one to define sets of contact information for authors containing things like name, address, affiliation, email address, phone number, etc. With the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output, these author contact information boxes will wrap if there are more of them than will fit on one line on the page. The lines containing these boxes will also be justified. The RenderX output neither automatically wraps these boxes nor justifies them.

### 5.5 Vertical fill

For title page material, only the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output allows using vertical fill between items on a particular page of output. This can be useful for automatically inserting whatever vertical space is needed between, say, the last author's name and some publishing information that needs to appear at the bottom of the page. RenderX requires using overt, fixed vertical spacing values. For the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output, then, one does not need to manually adjust this vertical space

font. The Graphite included in the 2010 version of X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X is not capable of rendering Awami Nastaliq well.

- (16) a. Farsi: bozorgan "leaders" (16) a. Farsi: bozorgan "leaders" (16) a. Farsi: bozorgan "leaders"  
 b. Gilaki: bozorgan "leaders" b. Gilaki: bozorgan "leaders" b. Gilaki: bozorgan "leaders"

**Figure 9:** Ascender/descender font differences: The RenderX output is on the left; XHTML output is in the middle; the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output is on the right. Fonts used are Times New Roman 12pt and Charis SIL 14pt.

- (1) a. Duu gúḁ, māḅ mlā-gə. (1) a. Duu gúḁ, māḅ mlā-gə.  
 house DEM.3C3 1S.CONTR make.PFV-3C3 house DEM.3C3 1S.CONTR make.PFV-3C3  
 ‘That house, it’s I who built it.’ ‘That house, it’s I who built it.’

**Figure 10:** Stacked diacritics on the third word from the left: The RenderX output is on the left; the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output is on the right.

RenderX: دی لآآآآآ لآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ

XHTML: دی لآآآآآ لآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ

X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X:  
 (not using Graphite) دی لآآآآآ لآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ

X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X:  
 (using Graphite) دی لآآآآآ لآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ آآآآآآ

**Figure 11:** Awami Nastaliq rendering. Only the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output using Graphite is correct. Since Graphite requires a rendering engine and only the Firefox browser has included it, support for Graphite rendering in XHTML is limited to Firefox.

The example text is in the Saraiki language (spoken in Pakistan) and is a section from the Universal Declaration of Human Rights. We are grateful to Sharon Correll for this example.

for each document. One must do so for the RenderX output, however. This is a non-issue for the XHTML output because there are no page breaks as there are in PDF output.

## 5.6 Line numbering

When submitting an article for review, some publishers want the PDF to have continuous line numbers throughout the document. Only the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output does this.

## 6 Features other outputs have that the L<sup>A</sup>T<sub>E</sub>X output does not

X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X does not allow for custom table cell padding and spacing. Having said that, the developer cannot remember any XLingPaper user ever asking for a way to do this for the X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X output. It just looks fine.

Setting the background color is not available for section titles.

Section 11.17.1.1 “Known limitations of using X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X” in the XLingPaper user documentation lists other known problems [10].<sup>50</sup>

## 7 Conclusion

While the XLingPaper approach to composing documents via DTD-controlled user interface limitations has great value in and of itself, the fact that it can produce great looking output via X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X makes the learning curve rewarding. We feel that being able to produce PDF via X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X has made XLingPaper a fantastic tool for linguists.

Additionally, XLingPaper serves as a model for other developers who seek a modular approach to creating custom publishing solutions. That is, one does not need to deploy the whole T<sub>E</sub>X Live system to create great looking outputs. Specific packages can be combined and redistributed to fit market needs.

## A Hyphenation supported languages

Language name	Two letter code	Three letter code
Albanian	sq	sqi
Amharic	am	amh
Arabic	ar	ara
Asturian		ast
Basque	eu	eus
Bengali	bn	ben
Bulgarian	bg	bul
Catalan	ca	cat
Coptic		cop
Croatian	hr	hrv

<sup>50</sup> [software.sil.org/downloads/r/xlingpaper/resources/documentation/xxe7/UserDocXMLmind.htm#sXeTeXLimitations](https://software.sil.org/downloads/r/xlingpaper/resources/documentation/xxe7/UserDocXMLmind.htm#sXeTeXLimitations)

Czech	cs	ces
Danish	da	dan
Dutch	nl	nld
English	en	eng
Esperanto	eo	epo
Estonian	et	est
Farsi	fa	fas
Finnish	fi	fin
French	fr	fra
Galician	gl	glg
German	de	deu
Greek	el	ell
Hebrew	he	heb
Hindi	hi	hin
Hungarian	hu	hun
Icelandic	is	isl
Indonesian	id	ind
Interlingua	ia	ina
Irish	ga	gle
Italian	it	ita
Lao	lo	lao
Latin	la	lat
Latvian	lv	lav
Lithuanian	lt	lit
Lower Sorbian		dsb
Malay	ms	msa
Malayalam	ml	mal
Marathi	mr	mar
Nynorsk	nn	nno
Occitan	oc	oci
Polish	pl	pol
Portuges	pt	por
Romanian	ro	ron
Russian	ru	rus
Sanskrit	sa	san
Scottish	gd	gla
Serbian	sr	srp
Slovak	sk	slk
Slovenian	sl	slv
Spanish	es	spa
Swedish	sv	swe
Syriac		syr
Tamil	ta	tam
Telugu	te	tel
Thai	th	tha
Turkish	tr	tur
Turkmen	tk	tuk
Ukrainian	uk	ukr
Urdu	ur	urd
Upper Sorbian		hsb
Vietnamese	vi	vie
Welsh	cy	cym

## References

- [1] Bartholomew, Doris A, and Louise C Schoenhals. 2019. *Bilingual Dictionaries for Indigenous Languages*. Edited by Thomas L Willett. 2nd edn. Tlalpan, Ciudad de México, México: Instituto Lingüístico de Verano, A.C. [SIL International in Mexico]. [sil.org/resources/archives/80401](https://sil.org/resources/archives/80401).
- [2] Beadle, Jennie, and Matthew Lee. 2020a. *Paratext 9 Manual — in English*. SIL International. [lingtran.net](https://lingtran.net).
- [3] Beadle, Jennie, and Matthew Lee. 2020b. *Paratext 9 Manual — in French*. SIL International. [outilingua.net](https://outilingua.net).
- [4] Beier, Christine, and Lev Michael. 2022. *Managing Lexicography Data: A Practical, Principled Approach Using FLEx (FieldWorks Language Explorer)*. In *The Open Handbook of Linguistic Data Management*, edited by Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, and Lauren B. Collister, 301–14. Open Handbooks In Linguistics. Cambridge, Massachusetts: The MIT Press. [doi.org/10.7551/mitpress/12200.003.0029](https://doi.org/10.7551/mitpress/12200.003.0029).
- [5] Bird, Steven, and Gary Simons. 2002. Seven Dimensions of Portability for Language Documentation and Description. In ISCA SALT MIL SIG: “Speech and Language Technology for Minority Languages”, 23–30. Las Palmas, Canary Islands, Spain: ELRA. [lrec-conf.org/proceedings/lrec2002/pdf/ws15.pdf#page=29](https://lrec-conf.org/proceedings/lrec2002/pdf/ws15.pdf#page=29).
- [6] Bird, Steven, and Gary Simons. 2003. Seven Dimensions of Portability for Language Documentation and Description. *Language* 79 (3):557–82. [jstor.org/stable/4489465](https://www.jstor.org/stable/4489465).
- [7] Black, Cheryl A., and H. Andrew Black. 2012. *Grammars for the People, by the People, Made Easier Using PAWS and XLingPaper*. In *Electronic Grammaticography*, edited by Sebastian Nordhoff, 103–28. LD&C Special Publication 4. Honolulu, Hawai‘i: University of Hawai‘i Press. [hdl.handle.net/10125/4532](https://hdl.handle.net/10125/4532).
- [8] Black, H. Andrew. 2009. Writing Linguistic Papers in the Third Wave. *SIL Forum for Language Fieldwork* 2009 (004): 11 pages. [sil.org/resources/publications/entry/7790](https://sil.org/resources/publications/entry/7790).
- [9] Black, H. Andrew. 2017. *Why Learn to Use XLingPaper*. Dallas, Texas: SIL International. [software.sil.org/downloads/r/xlingpaper/resources/documentation/WhyUseXLingPaper.pdf](https://software.sil.org/downloads/r/xlingpaper/resources/documentation/WhyUseXLingPaper.pdf).
- [10] Black, H. Andrew. 2020. *XLingPaper User Documentation* [Software Documentation]. Dallas, Texas: SIL International. [software.sil.org/downloads/r/xlingpaper/resources/documentation/xxe7/UserDocXMLmind.htm](https://software.sil.org/downloads/r/xlingpaper/resources/documentation/xxe7/UserDocXMLmind.htm).
- [11] Brownie, John. 2013. Adverbs in the Mussau-Emira Verb Phrase. *Language & Linguistics in Melanesia* 31 (1):1–11. [langlxmlanesia.com/issues](https://langlxmlanesia.com/issues).
- [12] Buck, Marjorie J. 2018. *Gramática del amuzgo Xochistlahuaca, Guerrero*. (Serie de gramáticas de lenguas indígenas de México №16.) Tlalpan, Ciudad de México, México: Instituto Lingüístico de Verano, A.C. [SIL International in Mexico]. [sil.org/resources/archives/75518](https://sil.org/resources/archives/75518).
- [13] Correll, Sharon. 2000. *Graphite: An Extensible Rendering Engine for Complex Writing Systems*. Paper presented at the 17th International Unicode Conference, San Jose, California. [rabbits.continuation.org/w/images/7/73/Graphite\\_paper.pdf](https://rabbits.continuation.org/w/images/7/73/Graphite_paper.pdf).
- [14] Daniel, Marco, and Elke Schubert. 2013. *The mdframed Package: Auto-split Frame environment* version 1.9b.
- [15] Di Biase-Dyson, Camilla, Frank Kammerzell, and Daniel A. Werning. 2009. Glossing Ancient Egyptian. Suggestions for Adapting the Leipzig Glossing Rules. *Lingua Aegyptia. Journal of Egyptian Language Studies* 17: 343–66. [wwwuser.gwdg.de/~lingaeg/lingaeg17.htm](http://wwwuser.gwdg.de/~lingaeg/lingaeg17.htm).
- [16] Dimitriadis, Alexis. 2016. *TeX/LaTeX Information*. Web page. [ling.upenn.edu/advice/latex.html](https://ling.upenn.edu/advice/latex.html).
- [17] Donnelly, Kevin. 2013. Representing Linguistic Pitch in X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X. *TUGboat* 34 (2): 223–27. [tug.org/TUGboat/tb34-2/tb107donnelly.pdf](https://tug.org/TUGboat/tb34-2/tb107donnelly.pdf).
- [18] Ebarb, Kristopher J. 2014. *Tone and variation in Idakho and other Luhya varieties*. University of Indiana Ph.D. dissertation. [pqdtopen.proquest.com/doc/1625743679.html?FMT=ABS](https://pqdtopen.proquest.com/doc/1625743679.html?FMT=ABS).
- [19] Frampton, John. 2006. *Pst-Asr: Tex Macros for Typesetting Autosegmental Representations*. Version: 1.1. CTAN. [ctan.org/tex-archive/graphics/pstricks/contrib/pst-asr/pst-asr-doc.pdf](https://ctan.org/tex-archive/graphics/pstricks/contrib/pst-asr/pst-asr-doc.pdf).
- [20] Frampton, John. 2012. *ExpPex for Linguists: Example Formatting, Glosses, and Reference*. Version: 4.1. [mathserver.neu.edu/~ling/tex/expex/base/doc/expex-doc.pdf](https://mathserver.neu.edu/~ling/tex/expex/base/doc/expex-doc.pdf).

- [21] Freitag, Constantin and Antonio Machicao y Priemer. 2019. *L<sup>A</sup>T<sub>E</sub>X-Einführung Für Linguisten*. Berlin, Germany: Humboldt-Universität zu Berlin. doi.org/10.13140/RG.2.2.29299.27682.
- [22] Holton, Gary. 2021. *Writing You Dissertation with L<sup>A</sup>T<sub>E</sub>X*. Typescript. Hawai‘i. Github.com. gmholton.github.io/files/DissertationWriting.pdf.
- [23] Kew, Jonathan. 2007. X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X Live. *TUGboat* 29 (1): 146–50. tug.org/TUGboat/tb29-1/tb91kew.pdf.
- [24] Kew, Jonathan and Stephen McConnell. 1990. *Formatting Interlinear Text*. Occasional Publications in Academic Computing, Number 17. Dallas, Texas: Summer Institute of Linguistics.
- [25] Kircher, Athanasius. 1652. *Ædipus Ægyptiacus, hoc est Vniuersalis Hieroglyphicæ Veterum Doctrinæ temporum iniuria abolitæ Instauratio*. Opus ex omni Orientalium doctrina & sapientia conditum, nec non viginti diuersarum linguarum autoritate stabilitum, Romæ: Ex Typographia Vitalis Mascaradi.
- [26] Knuth, Donald Ervin. 1984. *The T<sub>E</sub>Xbook. Computers & Typesetting*, vol. A. Reading, Massachusetts: American Mathematical Society; Addison-Wesley.
- [27] Knuth, Donald Ervin. 1986. *T<sub>E</sub>X: The Program. Computers & Typesetting*, vol. B. Reading, Massachusetts: Addison-Wesley.
- [28] Lamicela, Andrew Charles. 2020. *Distinguishing Passive from MP2-marked Middle in Koine Greek*. University of North Dakota M.A. thesis. commons.und.edu/theses/3277.
- [29] Lehmann, Christian. 2004. *Interlinear morphemic glossing*. In *Morphologie: Ein internationales Handbuch zur Flexion und Wortbildung / Morphology: an international handbook on inflection and word-formation*, edited by Geert E Booij, Christian Lehmann, Joachim Mugdan, and Stavros Skopeteas, 2:1834–57. Handbücher zur Sprach- und Kommunikationswissenschaft / Handbooks of Linguistics and communication science 17. Berlin, New York: Walter de Gruyter. doi.org/10.1515/9783110172782.2.20.1834.
- [30] Liter, Adam. 2017. *L<sup>A</sup>T<sub>E</sub>X Workshop (for Linguists)*. adamliter.org/content/LaTeX/latex-workshop-for-linguists.pdf.
- [31] Lovell, Douglas. 1999. T<sub>E</sub>XML: Typesetting XML with T<sub>E</sub>X. *TUGboat* 20 (3): 176–183. tug.org/TUGboat/tb20-3/tb64love.pdf.
- [32] Marlett, Stephen A. (compiler). 2012. La Frase Nominal. In Stephen A. Marlett (ed.) *Los Archivos Lingüísticos Me’phaa*. Instituto Lingüístico de Verano, A.C. [SIL International in Mexico]. mexico.sil.org/publications/i-wpindex/work\_papers\_-\_mephaa\_grammar\_files.
- [33] Marlett, Stephen A. 2019. *Phonology From the Ground Up: The Basics*. Dallas, Texas: SIL International. sil.org/resources/archives/79207.
- [34] Marlett, Stephen A. 2019. Presentation of three short texts in Isthmus Zapotec. SIL-Mexico Electronic Working Papers #25. Ciudad de México: Instituto Lingüístico de Verano, A.C. [SIL International in Mexico]. sil.org/resources/archives/80964.
- [35] Neri Méndez, Emilia and Stephen A. Marlett. 2011 (Nov). Presentación Analítica del Texto “Flor de Calabaza”. In Stephen A. Marlett (ed.) *Los Archivos Lingüísticos Me’phaa* (versión preliminar). Instituto Lingüístico de Verano, A.C. [SIL International in Mexico]. mexico.sil.org/publications/i-wpindex/work\_papers\_-\_mephaa\_grammar\_files.
- [36] Nordhoff, Sebastian, and Stefan Müller. 2020. *Language Science Press Guidelines*. Berlin, Germany: Language Science Press. langsci.github.io/guidelines/latexguidelines/LangSci-guidelines.pdf.
- [37] Paterson III, Hugh J. 2021. *Language Archive Records: Interoperability of Referencing Practices and Metadata Models*. University of North Dakota M.A. thesis. commons.und.edu/theses/3937.
- [38] Paterson III, Hugh J. 2021. *On Rights Management in Anthropological and Linguistic Sound Collections*. *ARSC Journal* 52 (3): 547–563. arsc-audio.org/journal.html.
- [39] Pellard, Thomas. 2019. Automatic formatting of interlinear glosses with L<sup>A</sup>T<sub>E</sub>X. *Cipanglossia* cipanglo.hypotheses.org/1221.
- [40] Peter, Steve. 2004. T<sub>E</sub>X and Linguistics. *TUGboat* 25 (1): 58–62. tug.org/TUGboat/tb25-1/peter.pdf.
- [41] Machicao y Priemer, Antonio, and Constantin Freitag. 2019. *L<sup>A</sup>T<sub>E</sub>X-Einführung für Linguisten*. Presentation at the MGK Workshop — SFB 1412, Berlin. linguistik.hu-berlin.de/de/staff/amp/latex20sfb/07-141-math2-trees-handout.pdf.

- [42] Rasmussen, Kent. 2018. *A Comparative Tone Analysis of Several Bantu D30 Languages (DR Congo)*. University of Texas Arlington Ph.D. dissertation. [hdl.handle.net/10106/27483](https://hdl.handle.net/10106/27483).
- [43] Rastorgueva, V. S., A. A. Kerimova, A. K. Mamedzade, L. A. Pireiko, and D. I. Edel'man. 2012. *The Gilaki Language*. Edited by Ronald M. Lockwood. Acta Universitatis Upsaliensis; Studia Iranica Upsaliensia 19. Uppsala, Sweden: Acta Universitatis Upsaliensis. [urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-182789](https://nbn-resolving.org/urn:nbn:se:uu:diva-182789).
- [44] Rei, Fukui. 1996. TIPA: A System for Processing Phonetic Symbols in L<sup>A</sup>T<sub>E</sub>X. *TUGboat* 17 (2): 102–14. [tug.org/TUGboat/tb17-2/tb51rei.pdf](https://tug.org/TUGboat/tb17-2/tb51rei.pdf).
- [45] Salfner, Sophie, and Tim Gaved. 2014. Working with ELAN and FLE<sub>X</sub> Together: An ELAN-FLE<sub>X</sub>-ELAN Teaching Set. Electronic Manuscript. SOAS, London, England. [web.archive.org/web/20210613121544/https://www.soas.ac.uk/elar/helpsheets/file122785.pdf](https://web.archive.org/web/20210613121544/https://www.soas.ac.uk/elar/helpsheets/file122785.pdf).
- [46] Schenner, Mathias, and Sebastian Nordhoff. 2016. Extracting Interlinear Glossed Text from L<sup>A</sup>T<sub>E</sub>X Documents. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, edited by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, et al., 4044–48. Portorož, Slovenia: European Language Resources Association (ELRA). [aclweb.org/anthology/L16-1638.pdf](https://aclweb.org/anthology/L16-1638.pdf).
- [47] SIL International. 2012. Comparison of OpenType and Graphite shaping speeds in a Nastaliq context. (accessed: 27 May 2022) [scripts.sil.org/cms/scripts/page.php?site\\_id=projects&item\\_id=graphite\\_otcompare](https://scripts.sil.org/cms/scripts/page.php?site_id=projects&item_id=graphite_otcompare).
- [48] Smith, Zac, Todd Snider, and Mia Wiegand. 2016. *L<sup>A</sup>T<sub>E</sub>X and Linguistics — How to Make Your Research Pretty*. Presentation at the Cornell Linguistics Circle, Cornell, New York. [conf.ling.cornell.edu/miawiegand/Latex\\_Slides.pdf](https://conf.ling.cornell.edu/miawiegand/Latex_Slides.pdf).
- [49] So Miyagawa, and Vincent W.J. van Gerven Oei. 2021. Building Web Corpus of Old Nubian with Interlinear Glossing as Digital Cultural Heritage for Modern-Day Nubians. In *The Proceedings of the 11th Conference of Japanese Association for Digital Humanities*, vol. 2021. 144–147. Tokyo: Historiographical Institute, The University of Tokyo. [www.hi.u-tokyo.ac.jp/JADH/2021/Proceedings\\_JADH2021\\_rev0905.pdf](https://www.hi.u-tokyo.ac.jp/JADH/2021/Proceedings_JADH2021_rev0905.pdf).
- [50] Thiele, Christina. 1995. T<sub>E</sub>X and Linguistics. *TUGboat* 16 (1): 42–44. [tug.org/TUGboat/tb16-1/tb46ling.pdf](https://tug.org/TUGboat/tb16-1/tb46ling.pdf).
- [51] Thiele, Christina. 2007. *Christina Thiele Interview by Dave Walden for the T<sub>E</sub>X Users Group*. Transcript. [tug.org/interviews/thiele.html](https://tug.org/interviews/thiele.html).
- [52] Unicode Consortium, ed. 1991. *The Unicode Standard: Worldwide Character Encoding*. Version 1.0. Reading, Massachusetts: Addison-Wesley. [unicode.org/versions/Unicode1.0.0](https://unicode.org/versions/Unicode1.0.0).
- [53] Ward, Monica. 2002. Reusable XML Technologies and the Development of Language Learning Materials. *ReCALL* 14 (2): 285–94. [doi.org/10.1017/S0958344002000629](https://doi.org/10.1017/S0958344002000629).
- [54] Wilson, Peter. 2007. The Memoir Class. *TUGboat* 28 (2): 243–46. [tug.org/TUGboat/tb28-2/tb89wilson.pdf](https://tug.org/TUGboat/tb28-2/tb89wilson.pdf).
- [55] Wilson, Peter. 2021. *The Memoir Class for Configurable Typesetting: User Guide*. version 3.70. Normandy Park, WA: The Herries Press. CTAN. [ctan.org/tex-archive/macros/latex/contrib/memoir/memman.pdf](https://ctan.org/tex-archive/macros/latex/contrib/memoir/memman.pdf).
- [56] Wood, Joyce Kathleen. 2012. *Valence-Increasing Strategies in Urim Syntax*. Graduate Institute of Applied Linguistics M.A. thesis. [diu.edu/documents/theses/Wood\\_Joyce-thesis.pdf](https://diu.edu/documents/theses/Wood_Joyce-thesis.pdf).
- [57] Woods, Frances. In preparation. *Halbi Interlinear Texts: Everyday Village Life*. Electronic Manuscript.
- ◇ H. Andrew Black  
blackhandrew (at) gmail dot com
- ◇ Hugh J. Paterson III  
i (at) hp3 dot me  
<http://hp3.me>

## STEX3 — A L<sup>A</sup>T<sub>E</sub>X-based ecosystem for semantic/active mathematical documents

Dennis Müller, Michael Kohlhasse

*This paper uses STEX3. The semantically annotated XHTML version of this paper is available at: [tinyurl.com/tug22stex](http://tinyurl.com/tug22stex)*

### Abstract

We report on STEX3 — a complete redesign and reimplementation (using L<sup>A</sup>T<sub>E</sub>X3) from the ground up of the STEX ecosystem for semantic markup of mathematical documents. Specifically, we present:

1. The STEX package that allows declaring semantic macros and provides a module system for organizing and importing semantic macros using logical identifiers. Semantic macros allow for annotating arbitrary L<sup>A</sup>T<sub>E</sub>X fragments, particularly symbolic notations and formulae, with their functional structure and formal semantics while keeping their presentation/layout intact. The module system induces a theory graph-structure on mathematical concepts, reflecting their dependencies and other semantic relations.
2. The RuSTEX system, an implementation of the core T<sub>E</sub>X engine in Rust. It supports converting arbitrary L<sup>A</sup>T<sub>E</sub>X documents to XHTML. For STEX3 documents, these are enriched with semantic annotations based on the OMDOC ontology.
3. An MMT integration: The RuSTEX-generated XHTML can be imported and served by the MMT system (meta-meta-theory or meta-meta-tool, depending on desired emphasis) for semantically informed knowledge management services, e.g. linking symbols in formulae to their definitions, or “guided tour” mini-courses for any (semantically annotated) mathematical concept/object.

Generally, STEX3 documents can be made not only *interactive* (by adding semantic services), but also “*active*” in that they actively adapt to reader preferences and pre-knowledge (if known).

### 1 Introduction

In mathematics (and adjacent disciplines), L<sup>A</sup>T<sub>E</sub>X is the de facto standard for typesetting *static* documents of all kinds. While L<sup>A</sup>T<sub>E</sub>X has thus established itself as the perfect tool for that job, since the advent of the internet a lot of functionalities have been developed and are commonly used (primarily via HTML) that allow for a more *active* interaction with documents than static formats allow for.

At the same time, computer scientists and mathematicians have developed techniques for representing the *formal semantics* of mathematical definitions, theorems, proofs and other statements in a computer-actionable manner. While the *strongest* of these techniques require significant expertise and effort to represent even relatively simple mathematical settings in their full formality, these are largely only required for the strongest forms of computer services (such as automated theorem proving); in contrast, relatively simple semantic annotations already allow for a plurality of useful services that can be integrated (primarily) in active documents.

To that end, we developed the STEX [5, 11] package and related systems, and its recent redesign and reimplementation in the form of STEX3.

STEX is a standard L<sup>A</sup>T<sub>E</sub>X package that provides a mechanism for declaring semantic macros (representing distinct mathematical concepts), which can be used to annotate arbitrary document fragments with their semantics to an arbitrary degree of formality (we speak of *flexiformality* [4]). These semantic macros are collected in modules which can be imported anywhere (analogously to L<sup>A</sup>T<sub>E</sub>X packages), and are in turn collected in math archives [2] which can be developed communally. The main difference between modules and L<sup>A</sup>T<sub>E</sub>X packages is that the objects of modules are (mathematical) concepts, objects, and structures, not abbreviations and layout primitives. As a consequence, modules usually contain the corresponding definitia that specify the concepts, objects, structures, and possibly theorems that state their properties and relations to others, and proofs that justify these all in a neat self-contained package of reusable components. The overall effect of this is that documents and archives can be developed modularly in an “object-oriented” fashion.

Many such archives are available on [gl.mathhub.info](https://github.com), in particular SMGloM, a multilingual mathematical glossary [10], currently containing  $\geq 2250$  concepts in English (93%), German (71%) and Chinese (11%).

In addition to being standard L<sup>A</sup>T<sub>E</sub>X documents, when converted to HTML the semantic information obtained from semantic macros (and other annotations) can be preserved in the form of HTML attributes. For those purposes, we implemented the RuSTEX system, a plain T<sub>E</sub>X engine converting arbitrary L<sup>A</sup>T<sub>E</sub>X documents to XHTML.

The resulting XHTML documents can be imported and served by the MMT system [8, 9], which can interpret the semantic annotations and offer corresponding semantics-aware services, effectively transforming the (originally) statically typeset L<sup>A</sup>T<sub>E</sub>X

document into an active HTML document. Our collection of such active documents generated from sTeX can be browsed on `mmt.beta.vollki.kwarc.info/sTeX`, including 3000+ pages of semantically annotated course notes and slides for various university lectures.

Notably, this paper itself uses sTeX. The semantically enriched version of it is linked above. Additionally, the source files are available on Overleaf at `www.overleaf.com/read/rvjbsnfshvhg` for demonstration purposes.

## 2 The sTeX package

For a detailed description of sTeX we refer to the documentation [6].

### 2.1 Modules and symbols

A module is opened via

```
\begin{smodule}{<name>}
```

Within a module, we can declare a new *symbol* with a corresponding semantic macro using `\symdecl`; for example, a symbol named `natural-number` with semantic macro `\Nat` would be declared with:<sup>1</sup>

```
\symdecl{Nat}[name=natural-number]
```

We can now reference our new symbol using e.g. `\symname`, where `\symname{Nat}` now yields the (annotated!) text “natural number”. Additionally, we can provide a new notation for the symbol using `\notation`, as in `\notation{Nat}{\mathbb N}`, allowing us to now use the semantic macro in math mode to print  $\mathbb{N}$ , or in text mode to annotate arbitrary text via `\Nat{<text>}`.

Semantic macros can also take arguments and be provided with additional semantic information, e.g. “types”. While the latter are ignored by L<sup>A</sup>T<sub>E</sub>X, the MMT system can use these for additional services, e.g. type checking (see below). Furthermore, the `\symdef` macro combines the (usually used in conjunction) functionalities of `\symdecl` and `\notation`. For example,

```
\symdef{plus}[
  name=addition,
  args=2, op=+,
  type=\funspace{Nat, Nat}{Nat}
]{#1 + #2}
```

declares `\plus` to be a binary function of type  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , and immediately provides it with an appropriate notation, after which `\plus ab` yields “ $a + b$ ”. The `op=+` in the above declaration allows us to refer to addition *itself* (rather than its application to arguments) via `\plus!`, yielding just  $+$ .

<sup>1</sup> See the source files of this paper for direct demonstrations of the examples here.

Analogously, we can introduce variables using `\vardef` (unlike symbols that have object-oriented scope, variables are local to the current T<sub>E</sub>X group).

### 2.2 Statements

Complex statements can be semantically marked-up using appropriate environments. For example, the following slightly simplified syntax allows us to declare commutativity as a predicate on binary operations and semantically annotate its definiens directly:

```
\symdecl{commutative}[args=1]
\begin{sdefinition}[for=commutative]
  \vardef{setA}{\comp{A}}
  \vardef{varop}[op=\circ, args=2]
    {#1 \circ #2}
```

```
A binary operation
$\fun{\varop!}{\setA, \setA}\setA$ is
called \definame{commutative}, iff
\definiens{
  \forall{
    \arg[2]{
      $\eq{\varop{a}{b}, \varop{b}{a}}$
    }
    \comp{for all}
  }
  \arg[1]{
    $\inset{a, b}\setA$
  }
}
\end{sdefinition}
```

yielding:

**Definition 2.1.** A binary operation  $\circ : A \times A \rightarrow A$  is called **commutative**, iff  $a \circ b = b \circ a$  for all  $a, b \in A$ .

(See the source files and/or documentation for details on the syntax.)

Similarly, we can mark up e.g. *theorems*, like

```
\begin{sassertion}[type=theorem,
  name=commutativity-of-addition]
\conclusion{
  \commutative{
    \arg{\plus{\comp{Addition}}} is
  }
  \comp{commutative}
}
\end{sassertion}
```

yielding

**Theorem 2.1.** *Addition is commutative.*

... and allowing us to now refer to commutativity of addition like any other symbol (e.g. via `\symname`).

The naming convention of prefixing environment names with `s-` (as in e.g. `sdefinition`) is to allow for functionality with respect to semantic optional

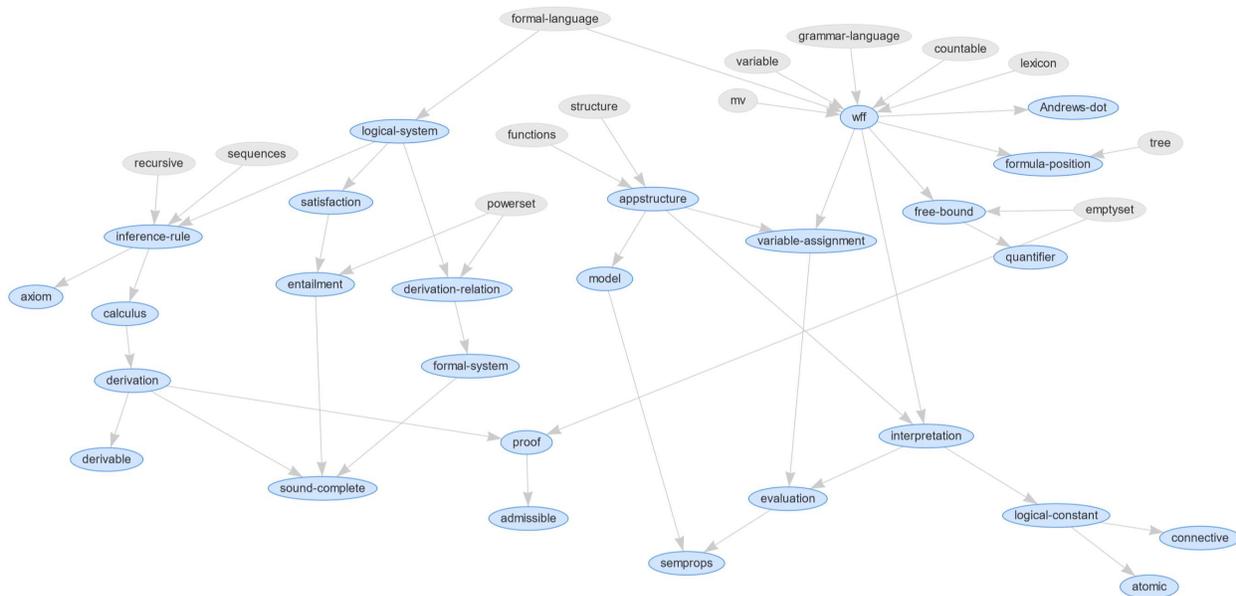


Figure 1: An example of a theory graph in the area of formal logic

arguments (e.g. `type=`, `for=`), while staying compatible with already existing environments. In fact, all the typesetting and semantic highlighting done by  $\text{\TeX}$  can be fully customized — in the case of the `definition` environment, for example by deferring typesetting to a standard `definition` environment defined via the `amsthm` package (as in this paper).

### 2.3 Importing modules

The semantic macros `\eq` and `\forall` used in our definition above represent *equality* and *universal quantification* (i.e. “for all”). These are imported from existing  $\text{\TeX}$  modules, namely `mod?Equal` in the math archive `sTeX/MathBase/Relations` and `mod/syntax?UniversalQuantifier` in the archive `sTeX/Logic/General`. If we only want to *use* the semantic macros in these modules, we can use the syntax `\usemodule[archive]{module}`. If, however, we are currently *in* a module, the contents of which *depend* on the symbols we want to import, we can use `\importmodule[archive]{module}` instead, which additionally *exports* the contents of the thus-imported module whenever we import the current one. For example, this paper might never explicitly import the `Equal` module, but could still use its contents if it imports others that in turn (transitively) import `Equal`.

This import mechanism naturally induces a theory graph, with modules as nodes and the import-relation as edges (see Figure 1).  $\text{\TeX}$  and MMT support more complicated edges as well, that represent less trivial and thus more interesting *theory*

*morphisms* between modules that knowledge can be translated along (see e.g. [9] for details).<sup>2</sup>

To let  $\text{\TeX}$  know where the required archives can be found, users can (among other ways) set a corresponding macro `\mathhub`, or set an environment variable `MATHHUB` once and for all. As a result, references to archives (and thus modules) are independent of the local filesystem. Notably, the number of modules imported in a given document can grow large very quickly — to allow for submission procedures (e.g. with *TUGboat* or *arxiv.org*) without needing to submit possibly hundreds of files, package options allow for storing and retrieving all semantic macros imported from external modules in/from a dedicated `\jobname.sms` file during compilation, which can be distributed alongside the document.

### 3 The $\text{RusTeX}$ system

There are multiple existing applications to convert  $\text{\TeX}$  documents to HTML, including but not limited to `TeX4ht` [1] and `\LaTeXML` [7]. Unfortunately, all of these have turned out to be deficient for our purposes, primarily due to their lacking support for either commonly used packages and macros or for introducing the required XML attributes for semantic annotations. We therefore decided to add to the existing set of such conversion tools.

<sup>2</sup> The full theory graph for (exemplary) the `SMGloM` can be navigated actively on `mtt.beta.vollki.kwarc.info/graphs/tgview.html?type=stexgraph&graphdata=smg1om`.

RuS $\TeX$ <sup>3</sup> is an implementation of a plain  $\TeX$  engine using the programming language Rust, outputting XHTML. It implements merely the (vast majority of) primitives of  $\TeX$ , e $\TeX$  and pdf $\TeX$ , and uses a locally installed L $\TeX$ E $\X$  distribution (by processing the available `latex.ltx` file) to handle L $\TeX$ E $\X$  documents. While this means that RuS $\TeX$  behaves virtually identically to pdf $\LaTeX$  (except for the output format), this comes at the cost of a priori no special treatment of standard L $\TeX$ E $\X$ -macros (although RuS $\TeX$  allows for adding special treatment of arbitrary macros on top). Instead, everything is expanded to primitive  $\TeX$  *whatsits*, which are exported to (primarily) `<div>`-nodes, styled via CSS-classes depending on the *whatsit*.

Notably, however, with s $\TeX$ 3 we opted for a mechanism analogous to the `pgf`-package: The relevant functionality is reduced to a mere handful of primitive macros for (HTML) annotations that a configuration file for a *backend* of choice (e.g. pdf $\LaTeX$  or RuS $\TeX$ ) can provide. This means that s $\TeX$  can be easily made compatible with alternative conversion tools, provided they support the basic functionality required.

#### 4 MMT integration and applications

MMT [8, 9] is a software system and API for a wide range of generic knowledge management services, providing algorithms for e.g. library management, parsing, (parametric) bi-directional type checking and reconstruction, term simplification, and various other computations on formal knowledge. The system uses a variant of the OMDOC [3] ontology, a representation format for semantically enriched mathematical documents.

The XHTML generated by RuS $\TeX$  can be imported by the MMT system directly, extracting the semantic annotations and converting them to the corresponding OMDOC elements. As a result, the full suite of MMT services are available for s $\TeX$  documents.

Services thus enabled in active documents currently include:

1. *Disambiguation*: Every symbol is assigned a globally unique MMT URI (i.e. identifier) that unambiguously determines the semantics of the symbol, regardless of e.g. notations used. In this paper (in the PDF), this MMT URI is shown when hovering over a symbol reference.

In the HTML version, hovering over a symbol reference shows (if available) the corresponding definition or theorem statement of the symbol,

allowing for quick reminders of the meaning of terms and notations.

2. *Type checking*: For fully formally annotated document fragments, we can make use of MMT’s type checking and inference mechanism: For example, in the definition and theorem in subsection 2.2, the MMT system can infer from our usage of `\definiens` that `commutative` is a unary predicate on binary operations, and uses that information to *type check* the theorem; that is, the system checks that the content of the `\conclusion`-macro is an actual proposition (which it determines by inferring the type of `commutative`), which recursively entails checking that `\plus` is indeed a binary operation (in this case on `\Nat`), and would warn us of a likely mistake otherwise.
3. *Guided tours*: Theory graphs provide us with the full semantic dependencies of a module. This allows us to generate small mini-courses that contain all prerequisite knowledge leading up to some intended concept in inverse dependency order: starting with the basics and ending with the concept to be explained. Clicking on a symbol reference in the XHTML opens a window linking to these guided tours.

Other features we are actively working on include:

1. *Notation selection*: Since s $\TeX$  allows for providing arbitrarily many notations for symbols, besides authors choosing the notation they prefer, in the HTML the document can in principle replace the notations used based on *readers’* preferences, making the resulting document more accessible for readers from different backgrounds with differing conventions.
2. *User-adapted guided tours*: In the context of classes at our university, we are working on modelling students’ knowledge as probabilistic “*user models*” that allow us to generate guided tours specifically adapted to a user’s prior knowledge. For example, by omitting already known concepts, selecting the most adequate examples, choosing their most familiar programming language for code snippets, etc.
3. *Flexible knowledge exploration/recommendation*: If we have a theory graph and a user model as above (possibly of a whole cohort of readers), we can use this information to recommend “useful knowledge items nearby” that might be interesting to the reader. These could be additional examples that help deepen understanding, theorems that give additional properties or relations, or even self-test problems. MMT can use the

<sup>3</sup> [github.com/slatex/RusTeX](https://github.com/slatex/RusTeX)

theory graph topology and user model information to determine what items are “nearby” the part of the theory graph that is (estimated to be) known to the reader.

To make these services as accessible to users as possible, we are actively developing a dedicated IDE in the form of a plugin for the VS Code editor using the *Language Server Protocol*.<sup>4</sup> The IDE integrates the MMT system (which in turn integrates  $\text{R}_{\text{U}}\text{S}_{\text{T}}\text{E}_{\text{X}}$ ) and can preview the active XHTML document generated from  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ . Additionally, it allows for searching both local and remote (on [gl.mathhub.info](http://gl.mathhub.info))  $\text{S}_{\text{T}}\text{E}_{\text{X}}$  content and downloading remotely available math archives directly.

## 5 Conclusion

The  $\text{S}_{\text{T}}\text{E}_{\text{X}}$  package now allows us to cover the complete spectrum from purely informal to fully formally annotated knowledge, directly in standard  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  documents. Via  $\text{R}_{\text{U}}\text{S}_{\text{T}}\text{E}_{\text{X}}$  and MMT, this makes formal knowledge management services available for  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  documents, allowing us to generate active documents that integrate semantically informed services for readers. The IDE bundles the whole toolchain required and makes it conveniently accessible to authors.

It is clear that semantic annotations constitute a considerable additional effort — in our experience up to 25–30% of the overall document development effort. Whether this investment can be amortized by the services that become available by it depends on the document or archive and on the context. We envision  $\text{S}_{\text{T}}\text{E}_{\text{X}}$  as an alternative to  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  primarily for documents with a high

- *impact*, i.e. which have many more readers than authors, or
- *inherent complexity*, which need semantic services to help readers understand them.

Some of the effort can surely be mitigated by advanced IDEs such as the one we are developing.

The main problem is that semantic annotations need semantic targets — i.e. annotated  $\text{S}_{\text{T}}\text{E}_{\text{X}}$  documents they can point to. This makes the first  $\text{S}_{\text{T}}\text{E}_{\text{X}}$  documents in a new domain very tedious to annotate, since we have to create archives for the “dependency cone”. We aim to alleviate this by providing a community portal for flexiformal mathematics: [MathHub.info](http://MathHub.info), where math archives can be hosted, discussed, and maintained so that — over time — we can ensure that the “cost” of annotating a document is proportional to the size of the document and not to the size of the domain.

<sup>4</sup> [github.com/slatex/sTeX-IDE](https://github.com/slatex/sTeX-IDE)

## References

- [1] E. Gurari, M. Hoftich, et al. *T<sub>E</sub>X4ht*. [tug.org/tex4ht/](http://tug.org/tex4ht/).
- [2] F. Horozal et al. Combining Source, Content, Presentation, Narration, and Relational Representation. *Intelligent Computer Mathematics*. Ed. by J. Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 212–227. [kwarc.info/frabe/Research/HIJKR\\_dimensions\\_11.pdf](http://kwarc.info/frabe/Research/HIJKR_dimensions_11.pdf).
- [3] M. Kohlhase. *OMDoc — An Open Markup Format for Mathematical Documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [4] M. Kohlhase. The Flexiformalist Manifesto. *14th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*. Ed. by A. Voronkov et al. Timisoara, Romania: IEEE Press, 2013, pp. 30–36. [kwarc.info/kohlhase/papers/synasc13.pdf](http://kwarc.info/kohlhase/papers/synasc13.pdf).
- [5] M. Kohlhase. Using  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  as a Semantic Markup Format. *Mathematics in Computer Science* 2:2, 2008, pp. 279–304. [kwarc.info/kohlhase/papers/mcs08-stex.pdf](http://kwarc.info/kohlhase/papers/mcs08-stex.pdf).
- [6] M. Kohlhase and D. Müller. *The sTeX3 Package Collection*. [github.com/slatex/sTeX/blob/main/doc/stex-doc.pdf](https://github.com/slatex/sTeX/blob/main/doc/stex-doc.pdf).
- [7] B. Miller. *LaTeXML: A L<sup>A</sup>T<sub>E</sub>X to XML Converter*. [mst.nist.gov/LaTeXML/](http://mst.nist.gov/LaTeXML/).
- [8] *MMT — Language and System for the Uniform Representation of Knowledge*. [uniformal.github.io/](https://uniformal.github.io/).
- [9] F. Rabe and M. Kohlhase. A Scalable Module System. *Information & Computation* 0:230, 2013, pp. 1–54. [kwarc.info/frabe/Research/mmt.pdf](http://kwarc.info/frabe/Research/mmt.pdf).
- [10] *SMGloM: A Semantic, Multilingual Glossary for Mathematics*. [gl.mathhub.info/smgloM/](http://gl.mathhub.info/smgloM/).
- [11] *sTeX: A Semantic Extension of TeX/LaTeX*. [github.com/sLaTeX/sTeX](https://github.com/sLaTeX/sTeX).

◇ Dennis Müller  
Friedrich-Alexander University,  
Erlangen-Nürnberg, DE  
[dennis.mueller@fau.de](mailto:dennis.mueller@fau.de)  
ORCID 0000-0002-4482-4912

◇ Michael Kohlhase  
Friedrich-Alexander University,  
Erlangen-Nürnberg, DE  
[michael.kohlhase@fau.de](mailto:michael.kohlhase@fau.de)  
ORCID 0000-0002-9859-6337

## Pushing math forward with ConTeXt lmtx

Hans Hagen, Mikael P. Sundqvist

Editor’s note: For enlarged views of many of the small details presented here, please also see the slide presentation, available from [tug.org/tug2022](https://tug.org/tug2022).

### Abstract

We report on some recent work on mathematical typesetting. Our main purpose has been to make both the input and output of math cleaner and more structured. Among the many enhancements, we mention here the introduction of new atom classes that has given better control over many details. We also cover the unboxing of fenced material which, together with improved line-breaking and more flexible multiline display math, has created a coherent way to produce displayed formulas that split over lines.

### 1 Introduction

When creating ConTeXt MkIV, the ConTeXt version that is based on LuaTeX, we only had Cambria available as an OpenType math font with a complete math table, so for the others we started out with runtime virtual math fonts assembled from traditional TeX fonts. Stepwise, more fonts became available. Not all fonts conform to the way Cambria does things. In order to deal with the inconsistencies in these fonts and because the specification was vague — it is better now — and had to be derived from e.g. how Microsoft Word does things, we ended up with a mix of generic runtime fixes to fonts and font-specific corrections done in what we call goodie files. However, given the amount of work involved, it never became complete.

That all changed when we became aware of Lansburgh’s 1964 book [1]. This book predates TeX, and it is, with its more than 400 pages of well-motivated typesetting rules — the majority of them about mathematics — the most comprehensive guide we are aware of. The book, written in Swedish, was originally used as a typesetting guide for the publisher Almqvist & Wiksell, in particular for the highly respected mathematics journal *Acta Mathematica*. For this reason Lansburgh discusses the rules for typesetting mathematics in great detail.

The question became: why can’t we do now what was recommended 50 years ago? The LuaMetaTeX math engine was already at this point partially redone and more configurable, but why not go further? We knew it would take much time to get all done, and it did (basically years of full time), but here we are. In the process we looked over the goodie files (they are now organized tweaks) and all fonts,

by now stable but flawed, were studied in detail. A direct consequence was rewriting the LuaTeX math engine to permit more control. So, in some ways, one has to thank Lansburgh for our work.

In this article we discuss only some of what the end user sees. At a lower level it all boils down to configuring the many (also new) font parameters, selectively fixing properties of glyphs, adding additional properties such as staircase-like kerns for some, setting up lots of pairwise spacing and penalties (we inherit where we can, so that saves some effort), defining rules that influence the inter-atom handling, etc. The LuaMetaTeX engine is completely configurable, meaning that we have more variables that can be set, and one can even change the styling rules, of which many were hard-coded. This is why a project like this takes much time and dedication but is also much fun.

One note has to be made: Don Knuth did a tremendous job on TeX and the math engine, and only by working with the code can one realize how quickly it was all achieved: we’re baffled. It does what was possible within the constraints of hardware and fonts, and it does it well. For instance, when we mention the `\nulldelimiterspace` parameter that we try to avoid, it doesn’t mean that it was not there for a reason: there is a subtle interplay between fonts, where characters have italic corrections as the means for spacing and attachments of sub/superscripts, and a zero-ordered spacing that then cooperates nicely with the few relatively unknown spacing parameters. As with everything TeX: it all makes sense when you see it in perspective and there are excellent tricks to be found in there. That said: we took advantage of today’s faster processors, plenty of memory, fonts that collect all shapes into one with more properties per font and shape, and in the end “time”, as we were under no pressure to finish this soon.

To date, enough has been done to fill a whole issue of *TUGboat*. Maybe we will wrap up some more in articles in due time. After all, we also have an additional wishlist to fulfill.

### 2 Math microtypography

By math microtypography we mean the fine-tuning of small details in mathematical formulas. Let us give an example. When you type `a_{0}b` in math mode you get  $a_0b$ . Have you ever noticed that there is a small space automatically inserted between the 0 and the  $b$ ? If the space is not there, as in  $a_0b$ , it is no longer clear if the 0 belongs to the  $a$  or to the  $b$ .

There are occasions when this space is unwanted. For example, we usually expect a symmetric space around relations (as in  $a = b$ ) and binary symbols

(as in  $a + b$ ). If, however, there is a subscript (or a superscript) just to the left of such a symbol, the surrounding space becomes uneven because of the inserted extra space.

$$a_0 b_0 = c_0 + d_0$$

The space is specified by the `\scriptspace` parameter. Don Knuth set it to 0.5pt in plain TeX (likely a choice that looked good with his 10pt bodyfont size). The `\scriptspace` parameter, and its particular value, has survived several decades, formats, body font sizes and engines. In ConTeXt lmtx we have introduced several options for the different math atom classes. One of these class options is `\nopostslackclassoptioncode`, and if it is set for a class then any inserted `\scriptspace` will be removed. Looking at the example above we see that the unwanted extra space is present before `=` and `+`. And indeed, both the relation class and the binary class do have this option set. Thus, when we typeset the formula above in ConTeXt lmtx we get the following.

$$a_0 b_0 = c_0 + d_0$$

The space between the  $a_0$  and the  $b$  is in fact no longer a `\scriptspace`, but we instead rely on the font parameter `SpaceAfterScript`.

The situation with `\nulldelimiterspace` is a bit similar. It is traditionally used as a kind of side bearing in fences and fractions. Its value was in plain TeX set to 1.2pt, and that has also stayed. In the formula  $\frac{1}{2}a$  the space is inserted between the  $\frac{1}{2}$  and the  $a$  and without it the formula would look bad:  $\frac{1}{2}a$ .

The `\nulldelimiterspace` is, however, also inserted *before* the fraction  $\frac{1}{2}$ , making the space before the formula slightly (1.2pt) larger than the space after it. This means that the margin will not be perfectly aligned if the fraction is located at the beginning or at the end of a line.

$$\frac{a}{b} + c = \frac{d}{e} f$$

In addition to the extra space at the left margin, the spaces around the `+` and the `=` above have become asymmetrical due to the inserted 1.2pt space.

In ConTeXt lmtx we use new atom classes to control the spacing around fractions. One of the new atom classes is the fraction class. Thus, we set the `\nulldelimiterspace` value to 0pt.

$$\frac{a}{b} + c = \frac{d}{e} f$$

Observe that no space is inserted to the left of the first fraction.

In the examples above we have used one of the many ConTeXt helpers (`\showmakeup[mathglue]`) to visualize the inserted spaces. For instance, to the right of the fraction we see `frabin`, which means that the classes that meet are fraction and binary; the space between them is set up to be a `\medmuskip`. We also used `\showglyphs` to draw the bounding boxes in orange (grayscaled in print). These and other helpers have been indispensable for our work.

### 3 A more general spacing model

In traditional TeX the spaces between atoms have traditionally been set to one of the following muskips.

```
\thickmuskip 5mu plus 5mu
\medmuskip   4mu plus 2mu minus 4mu
\thinmuskip  3mu
\zeromuskip  0mu
```

For example, between an ordinary and a binary atom, TeX inserts a `\medmuskip`. It has not been possible to set up the space between a single pair of atoms without altering the spaces between others.

In ConTeXt lmtx the inter-atom spaces are no longer hard-coded to `\thickmuskip`, `\medmuskip` and `\thinmuskip`. Users are free to define new muskips and to use them between any atom pair. After a lot of testing, we decided to alter the old muskips just a little, and added two new ones.

```
\thickmuskip 5mu plus 3mu minus 1mu
\medmuskip   4mu plus 2mu minus 2mu
\thinmuskip  3mu
\tinymuskip  2mu minus 1mu
\pettymuskip 1mu minus 0.5mu
\zeromuskip  0mu
```

We use the `\tinymuskip` for example between the radical and ordinary atoms, and between ordinary and fraction atoms. Traditionally, there is no space inserted in the first case.

$$a\sqrt{bc} + d\frac{e}{f}g$$

This is how it looks in ConTeXt lmtx:

$$a\sqrt{bc} + d\frac{e}{f}g$$

Note that there is a space between the  $\sqrt{b}$  and the  $c$ .

The `\pettymuskip` is mostly used in scriptstyle, in sub- and superscripts, where TeX traditionally inserts no space. We don't know why, but it might be that one simply wants the formulas to take less

space. It might also be that the smallest available non-zero muskip, the `\tinymskip`, was too big.

$$\sum_{k=0}^{j+n} a_k = e^{a+b-c}$$

With the `\pettymuskip` added, it looks like below, and you can judge for yourself whether it looks better or not.

$$\sum_{k=0}^{j+n} a_k = e^{a+b-c}$$

The observant reader has now realized that the spacing between atoms not only can be set to values other than the traditional four, but also they can also be different in different math styles. Indeed, when we do the setups we have access to the following keywords.

```
\alldisplaystyles
\alltextstyles
\allscriptstyles
\allscriptscriptstyles
\allmathstyles
\allsplitstyles
\alluncrampedstyles
\allcrampedstyles
```

Let us show examples with one of the new classes, the exponential class. This is a very small class, with currently only one member, the exponential  $e$ , accessed via `\ee`. This class is set up to inherit the inter-atom spaces from the ordinary atom class.

```
\setnewconstant \mathexponentialcode
\mathclassvalue exponential
```

```
\copymathspacing \mathexponentialcode
\mathordinarycode
```

Thus, if we type

```
\dm{rs \ee^{-rs} \ee^{st} tu} tu}
```

in math mode, we get

$$rse^{-rse^{st}tu}tu$$

Lansburgh suggests that a small space should be inserted between exponentials and other symbols, in particular if it carries exponents. We obtain that with the code below (and similar for the ordinary exponential combination).

```
\setmathspacing
\mathexponentialcode \mathordinarycode
\allsplitstyles \tinymskip
\setmathspacing
\mathexponentialcode \mathordinarycode
\allscriptstyles \pettymuskip
```

This results in some extra space around the  $e$ .

$$rse^{-rse^{st}tu}tu$$

#### 4 New atom classes

The classes defined in the LuaMetaTeX engine are ordinary, operator, binary, relation, open, close, punctuation, variable, active, inner, under, over, radical, fraction, middle, accent, fenced, ghost, vcenter.

The classes defined so far in ConTeXt lmtx are imaginary, differential, exponential, ellipsis, function, digit, explicit, division, factorial, wrapped, construct, mathpunctuation, dimension, unspaced, begin, end, all and unary.

You probably recognize many of the engine classes from classical TeX. We felt that it would be good to convert some standard constructions, like fractions and radicals, to their own classes. Once we decided to open up for more classes, we rapidly found a use for several new ones, some with just a few members, and some of a more technical nature. Let us give some comments.

Fractions and radicals are now their own classes, and since fenced material inherits their class structure from the content, there is currently no use of the inner class in ConTeXt lmtx.

The middle class, introduced in  $\varepsilon$ -TeX, was more like a technical hack built on top of the open class. In ConTeXt lmtx it is a true atom class.

The imaginary, differential, exponential, ellipsis and factorial classes have only a few members each. The differential class is perhaps the most interesting among them. The macro `\dd` yields a differential  $d$  with an adapted spacing; that is, the code `\int_{a}^{b} f(x) \dd x` in math mode gives  $\int_a^b f(x) dx$ . With

```
\setupmathematics[differentiald=upright]
```

the `\dd` gives an upright  $d$  instead,  $\int_a^b f(x) dx$ .

The factorial class consists of only one character, the exclamation mark. It is merely there to automatically get a small space between the exclamation mark and an ordinary symbol. Thus, if we type `\binom{n}{k} = \frac{n!}{(n-k)!k!}` in display math mode it comes out as follows.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

Observe the extra space between the  $)!$  and the  $k$ .

#### 5 Tweaking fonts

Take a look at the following formula, set with TeX Gyre Bonum Math:

$$\mathcal{I}[f] = \int_0^\pi [f'(t)]^2 dt$$

In the so-called “goodie files” we have collected fixes for the various math fonts. Let us look at the same formula, with the fixes in the goodie file applied.

$$q[f] = \int_0^\pi [f'(t)]^2 dt$$

We fix most issues in so-called tweaks, but some are also done with the help of font parameters, some of which are our own. We used

- the dimension tweak to scale the whole fraktur lowercase alphabet.
- the same tweak to modify the bounding box and italic correction of lower case italic f so that it does not clash with other letters.
- the font parameter `DisplayOperatorMinHeight` to increase the size of the integral sign, and `NoLimitSubFactor` to move the lower limit closer to the integral sign.
- the `fixprimes` tweak to move the prime down (see further discussion below).

The details of these fixes, as well as others, can be found in the goodie file `bonum-math.lfg`.

In order to solve the persistent issues with primes (fonts differ widely in that) the engine now supports primes natively. This means that every atom can have a super- and subscript, a super- and subpre-script as well as a prime attached. Optionally, scripts can be shifted to behave like an index. The fact that we need to deal with all four corners of a nucleus also means that we need to make sure that the glyphs behave well at both ends. That gave us some extra work. There are additional parameters to control the relative positioning of primes and superscripts.

Some Unicode math fonts, including Bonum, have several sizes of the integral sign, and we can use `\startintegral` and `\stopintegral` to make them grow as delimiters. This means we can write

```
\startintegral[bottom={a},top={b}]
\frac{1 + \frac{f_1(x)}{f_2(x)}\{f_2(x)\}}
{1 + \frac{f_3(x)}{f_4(x)}\{f_4(x)\}} \dd x
\stopintegral
```

in math mode, to get

$$\int_a^b \frac{1 + \frac{f_1(x)}{f_2(x)}}{1 + \frac{f_3(x)}{f_4(x)}} dx$$

Technically, the integral sign works as the left part of a paired delimiter. Thus, we see an example of a paired delimiter where the sub- and superscript are placed on the left delimiter. It is also possible to set the size of the integral sign manually. You can play with `\int[size=50pt]` if you need specific sizes.

## 6 Math macrotypography

ConTeXt has in the past had good support for type-setting displayed equations, and there has been rather

complete support for different types of alignments, numbering of equations, and so on [2].

Multiline formulas have historically been set in TeX via the `\halign` primitive. As a consequence these formulas have in fact been an array of math mode cells.

In ConTeXt there has for some time existed partial support for displayed formulas typeset as paragraphs, but they were not configured for real usage. When we opened up the set of atom classes, and introduced the unboxing of subformulas, it was also a good time to set this up and extend the functionality.

We build the formulas as one long formula, and do the layout mainly with the `split` and `align` keys. The user can also insert manual formatting with `\breakhere`, `\skiphere` and `\alignhere`.

The default value for the `split` key is `text`, and that means that formulas can split over lines, but not over pages. If we also want them to split over pages, we set `split` to `page`. The only difference between these two settings is the setup of penalties, and it is possible for the user to define their own. For formulas that fit on a line it does not matter.

$$\|P(\lambda) - P(\lambda_0)\|_{L^2(\Gamma) \rightarrow H^{5/2}(\Omega)} \leq C|\lambda - \lambda_0|$$

Longer formulas automatically split over lines.

```
\startformula
\iint K(x,y) f(x) g(y) \dd x \dd y
\leq \phi(p^{-1})
\left[\int x^{p-2} f(x)^p \dd x\right]^{1/p}
\left[\int g(y)^q \dd y\right]^{1/q}
\stopformula
```

$$\iint K(x,y) f(x) g(y) dx dy \leq \phi(p^{-1}) \left[ \int x^{p-2} f(x)^p dx \right]^{1/p} \left[ \int g(y)^q dy \right]^{1/q}$$

The splitting of the formula can be prohibited by adding `split=no` as an argument to `\startformula`. We get a formula that is set in a box (here we clip the formula so as not to mess up the formatting of this article).

$$\iint K(x,y) f(x) g(y) dx dy \leq \phi(p^{-1}) \left[ \int x^{p-2} f(x)^p dx \right]$$

If we instead add `align=slanted`, and also insert a `\breakhere` just before the `\leq`, we get

$$\iint K(x,y) f(x) g(y) dx dy \leq \phi(p^{-1}) \left[ \int x^{p-2} f(x)^p dx \right]^{1/p} \left[ \int g(y)^q dy \right]^{1/q}$$

The `align=slanted` flushes the first line left, the last line right, and midaligns the other lines. This

key can be given any of the values `middle` (default), `flushleft`, `flushright` and `slanted`.

With the default values of `split` and `align` we can easily add an align point with `\alignhere`.

```
\startformula
\frac{1}{2}(p^2 \abs{x} + \abs{x} p^2 )
\alignhere
= \abs{x} p \abs{x}^{-1} p \abs{x}
- \frac{1}{2} \abs{x}
( \laplace \abs{x}^{-1} ) \abs{x}
\breakhere
= \abs{x} p \abs{x}^{-1} p \abs{x}
- \frac{1}{2} \abs{x}^4 \pi \delta(0)
\abs{x}
\breakhere
= \abs{x} p \abs{x}^{-1} p \abs{x}
\stopformula
```

Observe the `\breakhere` where we want new lines.

$$\begin{aligned} \frac{1}{2}(p^2|x| + |x|p^2) &= |x|p|x|^{-1}p|x| - \frac{1}{2}|x|(\Delta|x|^{-1})|x| \\ &= |x|p|x|^{-1}p|x| - \frac{1}{2}|x|4\pi\delta(0)|x| \\ &= |x|p|x|^{-1}p|x| \end{aligned}$$

The careful reader also notes that there is a space after the close atoms, something that was suggested in [1]. Compare the final term on the right-hand side with  $|x|p|x|^{-1}p|x|$ , with no such space inserted.

We end with a slightly more advanced chain formula.

$$\begin{aligned} \mathbf{A} P'(iy_1, \dots, iy_n, i\eta_k) \\ \mathbf{B} &= (ir)^{k-1} \left[ P'_k \left( z_1, \dots, z_n, \frac{\eta_k}{r} \right) \right. \\ \mathbf{B} \quad \mathbf{S} \mathbf{3} &+ \left. \frac{1}{ir} P'_{k-1} \left( z_1, \dots, z_n, \frac{\eta_k}{r} \right) + \dots \right] \\ \mathbf{B} &= (ir)^{k-1} P'_k \left( z_1, \dots, z_n, \frac{\eta_k}{r} \right) + O(r^{k-2}) \end{aligned}$$

Here we have marked the align point with an A. Its position might at first glance be a bit surprising. To the formula we have added `textdistance=2em`. This is the space that is automatically added at each `\breakhere`, the extra horizontal shift you see at the B, compared to the A. At one row we have in addition added `\skiphere[3]`, that adds an extra space of 6em (the configurable unit is by default 2em). This is shown as S 3. This is how we typed the formula:

```
\startformula[textdistance=2em]
\alignhere
P'(iy_1, \ldots, iy_n, i\eta_k)
\breakhere
= (ir)^{k-1} \left[
P_k' \left( z_1, \ldots, z_n,
\frac{\eta_k}{r} \right)
\right]
\breakhere
```

```
\skiphere[3]
+ \frac{1}{ir} P_{k-1}'
\left( z_1, \ldots, z_n,
\frac{\eta_k}{r} \right)
+ \ldots
\right]
\breakhere
= (ir)^{k-1}
P_k' \left( z_1, \ldots, z_n,
\frac{\eta_k}{r} \right)
+ O(r^{k-2})
\stopformula
```

We emphasize that the formula above is broken inside the `\left[` and `\right]` fences. This is thanks to the possibility to unpack and repack subformulas.

One of the things we're currently experimenting with is carrying over kerns at the corners of nested subformulas. For instance, when a fenced formula, fraction, radical or any composed atom is prepared, it happens in a nested call to the mlist-to-hlist converter. The content is sort of abstract and wrapped in an atom of some class (say fenced) that determines spacing. In that case, anchoring a superscript cannot be related to the shape of, for instance, the right fence, which can have some extreme inward bending shape (as in Cambria). Dealing with that is not entirely trivial, but we managed to get it working. Of course, we then need to add shape-related kerning information to the goodie files because it is not part of the OpenType math concept. It is all about look and feel here.

### References

[1] W.N. Lansburgh. *Almqvist & Wiksells sättningsregler*. Almqvist & Wiksell, 1964.  
 [2] A. Mahajan. My way: Using `\startalign` and friends, 2006. [dl.contextgarden.net/myway/mathalign.pdf](https://dl.contextgarden.net/myway/mathalign.pdf)

- ◇ Hans Hagen  
<https://pragma-ade.nl>
- ◇ Mikael P. Sundqvist  
Department of Mathematics  
Lund University  
Box 118  
221 00 Lund  
Sweden  
mickep (at) gmail dot com



## The Treasure Chest

These are the new packages posted to CTAN ([ctan.org](http://ctan.org)) from April–August 2022. Descriptions are 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](http://ctan.org/pkg/pkgname).

A few entries which the editors subjectively believe to be especially notable are starred (\*); of course, this is not intended to slight the other contributions.

We hope this column helps people access the vast amount of material available through CTAN and the distributions. See also [ctan.org/topic](http://ctan.org/topic). Comments are welcome, as always.

◇ Karl Berry  
<https://tug.org/TUGboat/Chest>

---

### biblio

\***biber-ms** in **biblio**  
 Multi-script Biber.

---

### fonts

**simpleicons** in **fonts**  
 Simple Icons font support.

**srbtikz** in **fonts**  
 Stix2 support for Serbian and Macedonian italics.

**symbats3** in **fonts**  
 Support for the Symbats3 OpenType font.

**yfonts-otf** in **fonts**  
 OpenType versions of Yannis Haralambous's Old German fonts.

---

### graphics

**customdice** in **graphics/pgf/contrib**  
 Draw customizable dice.

**fancyqr** in **graphics/pgf/contrib**  
 Create fancy QR codes with TikZ.

**figput** in **fonts**  
 Create interactive or static figures in L<sup>A</sup>T<sub>E</sub>X.

**tikz-ext** in **graphics/pgf/contrib**  
 A collection of libraries for PGF/TikZ.

**tikzfill** in **graphics/pgf/contrib**  
 TikZ libraries for filling with images and patterns.

**tikzpingus** in **graphics/pgf/contrib**  
 Penguins.

---

### macros/generic

**expex-acro** in **macros/generic**  
 Wrapper for **expex** with support for glossing abbreviations.

**lt3luabridge** in **macros/generic**  
 Execute Lua code in any T<sub>E</sub>X engine that exposes the shell.

---

### macros/latex/contrib

**asternote** in **macros/latex/contrib**  
 Annotation symbols enclosed in square brackets and marked with an asterisk.

\***biblatex-ms** in **macros/latex/contrib**  
 Multi-script BIBL<sup>A</sup>T<sub>E</sub>X; requires **biber-ms**.

**chinesechess** in **macros/latex/contrib**  
 Typeset Chinese chess with l3draw.

**circledtext** in **macros/latex/contrib**  
 Create circled text.

**cprotectinside** in **macros/latex/contrib**  
 Use **cprotect**, arbitrarily nested.

**csassignments** in **macros/latex/contrib**  
 Support for computer science assignments.

**dvisirule** in **macros/latex/contrib**  
 Superimpose the covered hline and vline in a L<sup>A</sup>T<sub>E</sub>X **tabular** or **colortbl** environment.

**familytree** in **macros/latex/contrib**  
 Draw family trees.

**fixdif** in **macros/latex/contrib**  
 Typesetting differential operators.

**flexipage** in **macros/latex/contrib**  
 Flexible page geometry with marginalia.

**hereapplies** in **macros/latex/contrib**  
 Cross-linking applications of concepts.

**hideanswer** in **macros/latex/contrib**  
 Toggle printing of answers.

**hvextern** in **macros/latex/contrib**  
 Writing and reading of external source code, and inserting the output.

**inlinelabel** in **macros/latex/contrib**  
 Assign equation numbers to inline equations.

**jpneduenumerate** in **macros/latex/contrib**  
 Enumerative expressions in Japanese education.

**jpnedumathsymbols** in **macros/latex/contrib**  
 Mathematical equation representation in Japanese education.

**kfupm-math-exam** in **macros/latex/contrib**  
 Produce homework, quiz and exam papers.

**langnames** in **macros/latex/contrib**  
 Name languages and their genetic affiliations consistently.

**macros/latex/contrib/langnames**

`lt3rawobjects` in `macros/latex/contrib`  
 Declare and allocate L<sup>A</sup>T<sub>E</sub>X3 objects like C structures.

`magicwatermark` in `macros/latex/contrib`  
 Watermarks, based on `everypage` and `TikZ`.

`mathsemantics` in `macros/latex/contrib`  
 Semantic math commands in L<sup>A</sup>T<sub>E</sub>X.

`multifootnote` in `macros/latex/contrib`  
 Multiple numbers for the same footnote.

`multiple-choice` in `macros/latex/contrib`  
 Multiple-choice questions.

`ndsu-thesis-2022` in `macros/latex/contrib`  
 North Dakota State University support, update for 2022.

`postnotes` in `macros/latex/contrib`  
 Endnotes for L<sup>A</sup>T<sub>E</sub>X.

`precattl` in `macros/latex/contrib`  
 Write code containing tokens with unusual catcodes.

`prettytok` in `macros/latex/contrib`  
 Pretty-print token list.

`proflabo` in `macros/latex/contrib`  
 Draw laboratory equipment.

`rescansync` in `macros/latex/contrib`  
 Execute saved code to typeset text while preserving SyncT<sub>E</sub>X information.

`saveenv` in `macros/latex/contrib`  
 Save environment content verbatim.

`scripture` in `macros/latex/contrib`  
 Typesetting Bible quotations.

`sidenotesplus` in `macros/latex/contrib`  
 Place material in margins.

`simples-matrices` in `macros/latex/contrib`  
 Define matrices by given list of values.

`thermodynamics` in `macros/latex/contrib`  
 Macros for multicomponent thermodynamics documents.

`tkzexample` in `macros/latex/contrib`  
 Package for documentation of `tkz-*` packages.

`wrapstuff` in `macros/latex/contrib`  
 Wrapping text around stuff, using new L<sup>A</sup>T<sub>E</sub>X hooks.

---

#### **m/l/c/beamer-contrib/themes**

`beamerthemeamurmaple` in `m/l/c/b-c/themes`  
 A new modern beamer theme.

`beamertheme-tcolorbox` in `m/l/c/b-c/themes`  
 Inner beamer theme that reproduces standard beamer blocks.

`m/l/c/b-c/themes/beamertheme-tcolorbox`

---

#### **macros/latex/required**

`latex-lab` in `macros/latex/required`  
 Development pre-release. See L<sup>A</sup>T<sub>E</sub>X news installment in this issue.

---

#### **macros/luatex/latex**

`ligtype` in `macros/luatex/latex`  
 Suppress inappropriate ligatures, for German by default.

`luamathalign` in `macros/luatex/latex`  
 Flexible alignments in `amsmath` environments.

`luaquotes` in `macros/luatex/latex`  
 Smart setting of quotation marks.

`showhyphenation` in `macros/luatex/latex`  
 Show hyphenation points.

`showkerning` in `macros/luatex/latex`  
 Show kerns.

`spacekern` in `macros/luatex/latex`  
 Kerning between words and against whitespace.

---

#### **macros/plain**

`transparent-io` in `macros/plain/contrib`  
 Show for approval the filenames used in `\input`, `\openin`, or `\openout`. See article in *TUGboat* 43:1.

---

#### **macros/unicodetex/latex**

`swungdash` in `macros/unicodetex/latex`  
 Swung dash (U+2053), made by transforming the tilde.

`unisc` in `macros/unicodetex/latex`  
 Unicode small caps with Lua/X<sub>Q</sub>L<sup>A</sup>T<sub>E</sub>X.

---

#### **macros/xetex/latex**

`exam-zh` in `macros/xetex/latex`  
 L<sup>A</sup>T<sub>E</sub>X template for Chinese exams.

`hfutthesis` in `macros/xetex/latex`  
 L<sup>A</sup>T<sub>E</sub>X thesis template for Hefei University of Technology.

`xduts` in `macros/xetex/latex`  
 Xidian University T<sub>E</sub>X suite.

---

#### **support**

`texlive-dummy-fedora` in `support`  
 Dummy T<sub>E</sub>X Live RPM for use with Fedora and similar distributions.

---

**TUG 2022 abstracts**

Editor's note: Links to videos and other information posted at [tug.org/tug2022](http://tug.org/tug2022).

— \* —

**Looking outside the cockpit: An in-depth look at airport signage**

*Oliver Austin*

If you take a quick glance at an airport and its signage, you'll see many different situations where text is used to enhance and streamline processes for both pilot and ground crew alike. Thus, this exploration will take a closer look at such variations along the taxiway and apron at major airports, also discussing how the onset of autonomous aircraft can factor into it.

**The residual concepts of production vs. the emergent cultures of distribution in publishing**

*David Blakesley*

Who wins? The base or the superstructure? I'm not a Marxist per se, but I've lived this struggle for some time as a writer and publisher. In this keynote presentation, I describe my efforts to change or adapt the democratized tools of production to produce new forms of writing, which ultimately led to an ongoing battle with the dominant cultures of production in the world of publishing. I'll narrate two case studies. One focuses on the writing and production of an innovative, if not disruptive, textbook in the ultra-conservative textbook industry. The second tells the ongoing story of an interloping publishing company (Parlor Press) that reveals the central challenge of *distribution* for both writers and publishers, from typesetting (print) to transformation (digital).  $\LaTeX$  developers and users, take note! The return of the nonbreaking space and soft return is nigh!

**Fonts and formats of constitutions**

*Sarai Castañeda*

Through the different constitutions from different countries we'll look at, France, Canada, the United States, Mexico, and Argentina it is clear that the fonts range from cursive to typewriter-like. The fonts and format of each country's constitution are based on the time period it was written and other countries' influence. The countries have developed different iterations in order for the constitution to best represent their country's values.

**Comparing  $\TeX$  engines and formats**

*Max Chernoff*

Initially,  $\TeX$  was a single engine and a single format. However, over the past 40 years, the number of en-

gines and formats has significantly grown, meaning that there are multiple ways of implementing similar solutions depending on the  $\TeX$  variant used. In this talk, I'll introduce and compare each engine and format, focusing on both history and practical tips.

**Revamping a youth chess workbook using  $\LaTeX$  packages**

*Jennifer Claudio*

Playing chess can range from a casual pastime to a highly competitive event. Several local organizations offer chess as enrichment programs in K–12 schools, often having their own workbooks to supplement their instruction. One drawback is that these workbooks are often created using screen captures of online sources, resulting in low-quality outputs when used for print. This exploration tours a few packages used for typesetting diagrams for chess problems and puzzles and presents comparisons of one enrichment program's original workbook to equivalent pages produced using  $\LaTeX$ .

**Access and accessibility**

*Jonathan Fine*

The Chafee Amendment ([www.loc.gov/nls/about/organization/laws-regulations/copyright-law-amendment-1996-p1-104-197](http://www.loc.gov/nls/about/organization/laws-regulations/copyright-law-amendment-1996-p1-104-197)) to US copyright law “allows authorized entities to reproduce or distribute copies or phonorecords of previously published literary or musical works in accessible formats exclusively for use by print-disabled persons.”

This wonderful legal exemption to copyright nicely illustrates the relation between access (here to print works) and accessibility (here production of phonorecords, i.e., audiobooks). Here's another illustration.

Jonathan Godfrey, a blind Senior Lecturer in Statistics in New Zealand wrote to the Blind Math list “I used to use  $\TeX$ 4ht as my main tool for getting HTML from  $\LaTeX$  source. This was and probably still is, an excellent tool. How much traction does it get though? Not much. Why? I don't know, but my current theory is that tools that aren't right under people's noses or automatically applied in the background just don't get as much traction.” ([nfbnet.org/pipermail/blindmath\\_nfbnet.org/2021-January/009641.html](http://nfbnet.org/pipermail/blindmath_nfbnet.org/2021-January/009641.html))

Jonathan Godfrey also wrote to the BlindMath list “Something has to change in the very way people use  $\LaTeX$  if we are ever to get truly accessible pdf documents. I've laboured the point that we need access to information much more than we need access to a specific file format, and I'll keep doing so. [...] I do think a fundamental shift in thinking about how we get access to information is required across most

STEM disciplines. ([nfbnet.org/pipermail/blindmath\\_nfbnet.org/2021-March/009778.html](https://nfbnet.org/pipermail/blindmath_nfbnet.org/2021-March/009778.html))

This talk looks at the experience of visually impaired STEM students and professionals, from both the point of view of easy access to suitable inputs and tools and also the generation of accessible outputs, as pioneered and enabled by the Chafee Amendment.

### The UK T<sub>E</sub>X Users Group — a personal history

*Jonathan Fine*

UK TUG was established in the early 1990s. I’ve been a member of UK TUG almost from its start through to its dissolution earlier this year. Much has changed both in the T<sub>E</sub>X community and in the wider world over that time.

UK TUG was a significant part of the T<sub>E</sub>X community. Besides myself (Jonathan Fine), former members of UK TUG include Peter Abbott, Kaveh Bazargan, David Carlisle, Paulo Cereda, Malcolm Clark, David Crossland, Robin Fairbairns, Alan Jeffrey, Sebastian Rahtz, Arthur Rosendahl, Chris Rowley, Philip Taylor and Joseph Wright.

This list includes two past Presidents of TUG, the current Vice President and a past Secretary. Ten people on the list served on the TUG Board, for a total of over 30 years.

Five are or were members of the L<sup>A</sup>T<sub>E</sub>X3 project. One was the founder and for 8 years editor of T<sub>E</sub>X Live, and another the Technical coordinator of the  $\mathcal{N}\mathcal{T}\mathcal{S}$  project. One is a Lead Program Manager for Google Fonts.

This talk provides a personal history from `\begin{uktug}` to `\end{uktug}`, with a short ‘`\aftergroup`’ appendix.

### New in stock — a walk through recent L<sup>A</sup>T<sub>E</sub>X improvements (that you may have missed)

*Ulrike Fischer*

In this talk I present a selection of improvements we made in the recent L<sup>A</sup>T<sub>E</sub>X releases. The changes are not discussed in depth; the goal is to give some interesting examples and make you curious enough to explore the documentation and learn more. (See the L<sup>A</sup>T<sub>E</sub>X news installment in this issue, and previously, for details: [latex-project.org/news](https://latex-project.org/news).)

### Boxes and glue: T<sub>E</sub>X algorithms reimplemented

*Patrick Gundlach*

T<sub>E</sub>X (and therefore L<sup>A</sup>T<sub>E</sub>X) have enjoyed great popularity over the years as an extremely flexible, versatile, and robust text typesetting system. The flexibility comes not least from the ability to modify the behavior of T<sub>E</sub>X through programming and from Knuth’s

foresight in recognizing the individual elements on the page as small, rectangular building blocks that can be combined into larger units and also manipulated (box).

The development of LuaT<sub>E</sub>X made modern applications possible for the first time in the long history of T<sub>E</sub>X via some extensions:

- The number of characters in fonts is no longer limited to 256. This eliminates crutches like output encoding.
- Through the integration of HarfBuzz a solid “shaper” is available. This allows OpenType features and complicated writing systems (e.g., Arabic) to be output without any problems.
- The system can be programmed with Lua instead of the built-in macro language.
- Due to the clever PDF support, almost all PDF properties and standards can be supported.

I use these extensions for the program ‘speedata Publisher’, which is mainly made for the fully automatic creation of product catalogs and data sheets from XML.

Despite all the achievements of T<sub>E</sub>X and LuaT<sub>E</sub>X, there are still serious disadvantages:

- T<sub>E</sub>X and LuaT<sub>E</sub>X are anything but modular. Changing single areas is especially difficult, because T<sub>E</sub>X is not designed for that.
- Some things cannot be achieved with LuaT<sub>E</sub>X’s on-board tools. For example, HTTPS requests require an external library. Documents in our catalog area often get their images from image databases that are accessed via HTTPS.
- For other tasks, too, it is better to use an external library than to reinvent the wheel. For example, an XML parser or a library for bidirectional text typesetting.
- Parallelization of tasks: modern processors usually have several processor cores, which lie idle with T<sub>E</sub>X. Several tasks in T<sub>E</sub>X could be executed in parallel. Paragraphs could be wrapped with different parameters and then the best one selected. Loading font files and preparing them for subsetting in PDF does not have to be done sequentially. T<sub>E</sub>X does not provide such facilities.
- Distributing LuaT<sub>E</sub>X binaries across platforms is difficult due to external dependencies. For single applications you don’t want to ship or require a whole T<sub>E</sub>X Live installation.

The restrictions mentioned have troubled me considerably. Regarding the output quality of T<sub>E</sub>X, there are hardly comparable alternatives — especially

not in the open source realm. Therefore, there seemed no alternative left but to re-implement  $\text{\TeX}$  in a “modern” programming language. Some years ago there was already such an attempt ( $\mathcal{N}\mathcal{T}\mathcal{S}$ ), but it failed. After long pondering, respectively to meet my requirements for a text typesetting system for catalogs and datasheets, I came to the conclusion that I “only” take over the algorithms and the logic of  $\text{\TeX}$ , but not the input language.

#### *Boxes and glue*

“Boxes and glue” is a library written in the Go programming language. The name is based on the model of  $\text{\TeX}$  with the stretchable spaces between the rectangular units. The development of boxes and glue is quite advanced and includes among other things:

- $\text{\TeX}$ ’s smallest units (node) with ways to nest them inside each other (vbox, hbox).
- $\text{\TeX}$ ’s paragraph breaking algorithm.
- The pattern-based hyphenation.
- The inclusion of TrueType and OpenType fonts and PNG, JPEG, and PDF images.
- Text shaping with HarfBuzz.

Besides these basic parts, there is yet another library that builds on `boxesandglue`. It offers:

- Reading XML files
- Interpretation of HTML and CSS
- grouping of font files into families with easy font selection
- Handling of colors of all kinds (RGB, CMYK, spot colors)
- Tagged PDF

The application programming interface (API) is not yet fixed. The development of boxes and glue is being carried out in parallel with the further development of the speedata Publisher ([github.com/speedata/xts](https://github.com/speedata/xts)) and the requirements here largely determine the programming interface of `boxesandglue`. Since it is a library, there is no fixed input language as with  $\text{\TeX}$ . In this respect also, `boxesandglue` is also yet suitable for and (end) user.

#### *References*

- $\mathcal{N}\mathcal{T}\mathcal{S}$ : [en.wikipedia.org/wiki/New\\_Typesetting\\_System](https://en.wikipedia.org/wiki/New_Typesetting_System)
- Boxes and glue: [github.com/speedata/boxesandglue](https://github.com/speedata/boxesandglue)
- speedata Publisher: [github.com/speedata/publisher](https://github.com/speedata/publisher)
- XTS XML: [github.com/speedata/xts](https://github.com/speedata/xts)

### **Using $\text{\LaTeX}$ deployed in AWS as a PDF report generation tool for a cancer clinical trial search engine**

*Hubert Hickman, Matthew Mariano, Haibin Wu, Hong Dat Cheung*

Matching cancer patients with clinical trials is a complex process. One of the outputs of that process is the production of a PDF report containing relevant information about a set of trials. In this paper we present strategies, challenges, and conclusions regarding our use of  $\text{\LaTeX}$  deployed in AWS to generate PDF reports.

### **Bridging the gap between $\text{\LaTeX}$ /PDFs and the modern web**

*Nicolas Jimenez*

In this talk we explore the history of  $\text{\LaTeX}$  and PDFs in scientific communication, the roles these tools play, and how those roles may evolve over time. We discuss the rise of Markdown for web publishing, its limitations, and opportunities. We also touch on some recent developments by Mathpix to facilitate document interoperability and accessibility for researchers and the broader STEM community.

### **Right to left beamer documents in $\text{\XeTeX}$**

*Vafa Khalighi*

I will discuss the recent changes to the  `bidi`  package allowing users to produce right to left beamer documents describing the challenges and what needs to be done. I will also discuss other recent changes of the  `bidi`  package.

### **Bidirectional multi-columns and paragraph footnotes in $\text{\TeX}$**

*Vafa Khalighi*

Appendix D (Dirty Tricks) of *The  $\text{\TeX}$ book* describes algorithms for multi-column typesetting and paragraph footnotes, among much more. The described algorithms are used in various  $\text{\TeX}$  packages such as `footmisc`, `fnpara`, `manyfoot`, and many others.

When the package `multicol` is used, things get more complicated. Another level of complication arises when you want to mix these with both right-to-left and left-to-right typesetting.

The  `bidi`  package provides both right-to-left and left-to-right multi-columns and paragraph footnotes. This talk will describe my own experience learning about how other packages implement multi-columns and paragraph footnotes, and also the approach I took in the  `bidi`  package for these features.

### **Typesetting mathematics in Persian**

*Vafa Khalighi*

I will discuss how mathematics is typeset in Persian and what is required. I will also talk about how

the X<sub>Y</sub>Persian package implements these features and show some examples. I will then discuss recent changes to the xepersian package allowing users to change between English and Persian digits mid-math mode.

### Observations and analysis of Vietnamese text

*Tia Luc*

Having Vietnamese as my first language and English as my dominant language has inspired exploration of the history and applications of the former. Considering how Vietnamese and English both use the Latin alphabet, this presentation will explore the similarities and differences between the two using a collection of instances in which Vietnamese text is displayed in our world.

### Accessible tables using ‘Tagged PDF’

*Ross Moore*

Some basic requirements for accessibility of tabular material are:

- each cell, whether header or content, must have an attribute providing a unique ID for that cell;
- each data cell must specify the corresponding row and column headers that most directly provide the meaning of the information contained within the cell. This is done via a `Headers` attribute using the unique IDs for the header cells.

Header cells themselves may have other row or column headers; e.g., as a common header for a block of rows or columns.

Tagged PDF has the tagging and mechanisms to provide such attributes. When the PDF is translated into HTML (using the ngPDF online converter, say) this information is recorded in the web pages, to be available to Assistive Technologies. In this talk we show several examples of tables specified using various packages, as in *The L<sup>A</sup>T<sub>E</sub>X Companion*, both in PDF and HTML web pages.

A novel coding idea that allows this to be achieved was presented. This involves two aspects:

- turning the ‘&’ character into an active token while the tabular material is being processed;
- use of ‘look-ahead’ to see what kind of material is coming at or before the start of each tabular cell.

The example documents shown can be found on the author’s website, at [science.mq.edu.au/~ross/TaggedPDF/TUG2022/](http://science.mq.edu.au/~ross/TaggedPDF/TUG2022/). This HTML page was itself created using the same methods, from a PDF file which is available at [science.mq.edu.au/~ross/TaggedPDF/TUG2022/TableSite.pdf](http://science.mq.edu.au/~ross/TaggedPDF/TUG2022/TableSite.pdf).

Some rationale concerning how header cells are determined in real-world documents is explained in

one of the examples: [science.mq.edu.au/~ross/TaggedPDF/TUG2022/FishTables-only.pdf](http://science.mq.edu.au/~ross/TaggedPDF/TUG2022/FishTables-only.pdf)

### Machine translation of mathematical text

*Aditya Ohri, Tanya Schmah*

We present a machine translation system, the Poly-Math Translator, for L<sup>A</sup>T<sub>E</sub>X documents containing mathematical text. The system combines a L<sup>A</sup>T<sub>E</sub>X parser, tokenization of math and labels, a deep learning Transformer model trained on mathematical and other text, and the Google Translate API with a custom glossary. Ablation testing shows that math tokenization and the Transformer model each significantly improve translation quality, while Google Translate is used as a backup when the Transformer does not have confidence in its translation. For L<sup>A</sup>T<sub>E</sub>X parsing, we have used the pandoc document converter, while our latest development version instead uses the TexSoup package. We will describe the system, show examples, and discuss future directions.

### Musical composition typesetting

*Christopher Park, Emily Park*

We will explain the typesetting of a musical composition using the L<sup>A</sup>T<sub>E</sub>X markup.

### T<sub>E</sub>X Live 2022 status update

*Norbert Preining*

This talk reports on changes within the T<sub>E</sub>X Live project and distribution over the last year, as well as looking at further development directions and challenges we are facing.

### Bricks and pieces

*samcarter*

Real world bricks and jigsaw puzzles are a fun pastime for many people. The tikzbricks and jigsaw packages bring them to the L<sup>A</sup>T<sub>E</sub>X world. This short talk will give an overview of both packages and show examples how they can be used.

Detailed descriptions of usage and options can be found in the respective package documentations, linked from the CTAN package pages: [ctan.org/pkg/jigsaw](http://ctan.org/pkg/jigsaw) and [ctan.org/pkg/tikzbricks](http://ctan.org/pkg/tikzbricks).

### A gentle introduction to Markdown for writers

*Tereza Vrabcová*

T<sub>E</sub>X is great for producing beautiful documents, but not the easiest to read and write. At this workshop, you will learn about Markdown and how you can use it to produce different types of beautiful documents from beautiful source texts that don’t distract you from your writing.

**MAPS 52 (2022.1)**

MAPS is the publication of NTG, the Dutch language  $\text{\TeX}$  user group (<https://www.ntg.nl>).

MAPS REDACTIE, Welcome; pp. 1–2

By means of the MAPS we want to keep you informed of developments, and reward our members for their loyal support to the  $\text{\TeX}$  developers. We also offer space to readers who let others share their experiences with  $\text{\TeX}$ , MetaPost, fonts and such. So don't hesitate to send us articles. Half a page is already very nice, more is appreciated. It does not have to be a 'heavy cost', as readers are very interested in reading how others use  $\text{\TeX}$ . So an article like "This is what I do with  $\text{\TeX}$ , here is how I do it and now you can do it too" is very welcome!

Although the Internet is an important source of information today, paper continues to fulfill a function within the association. After all, that fits with  $\text{\TeX}$ !

HANS HAGEN, Dutch government math rendering; pp. 3–8

Mikael Sundqvist and I have spent quite some time on an upgrade of the math engine in LuaMeta $\text{\TeX}$ , with an example of an educational document published by the Dutch government.

HANS HAGEN, MIKAEL SUNDQVIST, A different approach to math spacing; pp. 9–36

Extending and generalizing math spacing, atoms, classes, and more.

FRANS GODDIJN, Danlan type by Adriaan Goddijn — (and a salacious gnome); pp. 37–40

The author (Frans Goddijn) purchased a drawing by Adriaan Goddijn (no relation), not for the image but to have a close look at the way the artist signed it, namely with his initials in a font he had created himself over the course of twenty years, in which every character of the alphabet is depicted using just seven glyphs, inspired by the Roman Capitalis Quadrata. He named it DANELAN (also called DANLAN) because he worked on it in the country of Denmark and the Dutch province of Friesland.

TACO HOEKWATER, Danlan type by Adriaan Goddijn — (quick font hack); pp. 41–46

When Frans Goddijn first showed me the Danlan font article in September 2019, I immediately thought that it would be fun to play with those letters a bit in  $\text{\TeX}$  and MetaPost.

But then the almost inevitable thing happened that so often happens to me: I got distracted by other things, and forgot about Danlan completely. Until this spring, when Frans reminded me that I had promised an article for the MAPS. This is that promised article: it will show what a few days playing around with a specification and MetaPost, FontForge, and Con $\text{\TeX}$ t got me. I have not created a complete font by any means, but it is just enough of one to show off a little bit and document how the creation process worked out for me.

MIKAEL SUNDQVIST, Finding all intersections of paths in MetaPost; pp. 47–70

In this article we will discuss different ways to implement macros to find all intersections of paths in MetaPost. We will first work out some rather simple ideas, providing partial solutions on the macro level. We then show by an example how the `intersectiontimes` macro works, and describe how it was extended in the engine by Hans Hagen.

HANS HAGEN, Cyrillisch in publieke fonts [Cyrillic in public fonts]; pp. 71–72

A review of Cyrillic support in several publicly-available fonts.

YURI ROBBERS, Tante Lenie weet raad — Uw trouwe steun en toeverlaat voor al uw problemen [Aunt Lenie knows what to do — Your loyal support and rock for all your problems]; pp. 73–75

This time Aunt Lenie helps some NTG members and other deep souls with their  $\text{\TeX}$  problems. This is how she helps Tamara J., designer of board games, to depict beautiful dice in the manual for her new game, using  $\text{\LaTeX}$ , and she helps classical language teacher Jaap T. to mark words in a text for a test he is making in  $\text{\XeLaTeX}$ . Finally, she helps Herman R., a mathematician who got stuck with boldfaced math formulas in section titles in plain  $\text{\TeX}$ .

TACO HOEKWATER, Dice3D OpenType — (quick font hack two); p. 76

The previous article by Yuri Robbers shows simulated 3D dice. That font existed only as a Metafont source file, so for the Con $\text{\TeX}$ t-format MAPS article, I had to quickly create an OpenType version.

HANS HAGEN, The art of MAPS proofreading; pp. 77–81

Con $\text{\TeX}$ t support for the 3D dice font described in the previous article, with enhancements.

FABRICE LARRIBE, MetaFun for generative art; pp. 82–90

This article shows how MetaFun can be used to create generative art, by showing the construction of three projects, step by step, in remarkably colorful images.

JOS WINNINK, Afscheid [Goodbye]; pp. 91–92

I have been a member of the NTG since 1990. Now that I'm retired, my use of  $\text{\TeX}$  has decreased to such an extent that I no longer see myself as an active user. In this article I look back on more than 30 years of  $\text{\TeX}$  and especially  $\text{\LaTeX}$ -related activities.

[Received from Wybo Dekker.]

---

**La Lettre GUTenberg 45, 2022**

*La Lettre GUTenberg* ([gutenberg-asso.fr/-Lettre-GUTenberg-](http://gutenberg-asso.fr/-Lettre-GUTenberg-)) is a free online publication of GUTenberg, the French-language T<sub>E</sub>X user group. Its new document class is now available as the `letgut` package on CTAN and the T<sub>E</sub>X distributions.

Issue #45 was published May 20, 2022.

PATRICK BIDEAULT, Éditorial [Editorial]; pp. 1–3

MAXIME CHUPIN AND PATRICK BIDEAULT, Compte-rendus de conseils d’administration [Minutes of the board’s meetings]; pp. 3–12

MAXIME CHUPIN, Retour sur le choix du nouveau nom de domaine [Report about the new domaine name]; pp. 12–14

The new domain name was chosen by a poll among GUTenberg’s members.

PATRICK BIDEAULT, Les différents travaux de l’association [The various works of the group]; pp. 15–16

CÉLINE CHEVALIER, PATRICK BIDEAULT, DENIS BITOUZÉ, MAXIME CHUPIN, Et maintenant, une bonne *vieille* veille technologique ! [Technology watch]; pp. 17–32

130 new CTAN packages, August 2021–May 2022.

THIERRY LARONDE, KerT<sub>E</sub>X, un projet d’ampleur développé par un contributeur francophone [KerT<sub>E</sub>X, a major project developed by a French-speaking contributor]; p. 33

A brief report about KerT<sub>E</sub>X, a project hosted at [kertex.kergis.com/en](http://kertex.kergis.com/en).

MAXIME CHUPIN, Promouvoir L<sup>A</sup>T<sub>E</sub>X dans l’enseignement au collège et au lycée ? [Promoting L<sup>A</sup>T<sub>E</sub>X in teaching at middle and high school]; pp. 34–41

JACQUES ANDRÉ & PATRICK BIDEAULT, La fonte de ce numéro : Infini [This issue’s font: Infini]; pp. 42–80

The Infini typeface was created by Sandrine Nugue for the National Center for Plastic Arts. And it is precisely because of this public commission that we have decided to use it for this issue of *La Lettre GUTenberg*. Using our usual layout tools with this font seemed an interesting challenge: how were we going to access the various glyphs without benefiting from a package which simplifies this work?

Moreover, deciding to use this typeface meant putting it to the test: for *Lettre*, we usually call on a wide range of characters: with or without serifs, fixed width, dedicated to mathematics, etc.; how was Infini going to fit into our production process? And what solutions were we going to find to respond to any technical problems?

Finally, using this typeface here means letting it unfold on many pages, appreciating the level of gray, the different forms and spaces. This allows you to form an opinion based on a real reading and not merely on the contemplation of an example.

[Editor’s note: The font, shown in its most playful form in the titles of the issue’s articles, is reminiscent of Matthew Carter’s Mantinia font, and of “space-saving” architectural inscriptions; see [tug.org/TUGboat/tb35-1/tb109beet.pdf](http://tug.org/TUGboat/tb35-1/tb109beet.pdf) for an illustration, and we highly recommend visiting *La Lettre* to see how it’s applied there.]

JÉRÉMY JUST, Compte rendu de lecture [Book review]; pp. 81–82

About Christophe Aubry’s book *L<sup>A</sup>T<sub>E</sub>X – Conception de documents élaborés et structurés*, ENI, 2021.

PATRICK BIDEAULT, Compte rendu de lecture [Book review]; pp. 82–84

About Isabel Meirelles’ book *Design de l’information (Design for Information)*, Rockport Publishers, 2013), published in French in 2014 by Parramón.

PATRICK BIDEAULT, En bref [At a glance]; pp. 85–87

Short news items about a new French website about TikZ, the way `xindex` handles the French language, Helmut Schmid, the DeMo Festival and more.

[Received from Patrick Bideault.]

**Die T<sub>E</sub>Xnische Komödie 2/2022**

*Die T<sub>E</sub>Xnische Komödie* is the journal of DANTE e.V., the German-language T<sub>E</sub>X user group ([dante.de](http://dante.de)). Non-technical items are omitted.

STEFAN KOTTWITZ, Bericht über Projektförderung von L<sup>A</sup>T<sub>E</sub>X-Servern [Report on project funding of L<sup>A</sup>T<sub>E</sub>X servers]; pp. 6–10

Report on the project funding of Stefan's L<sup>A</sup>T<sub>E</sub>X servers by DANTE.

ADELHEID BONNETSMÜLLER, Having Fun with L<sup>A</sup>T<sub>E</sub>X: Klein- und großkariert [Having fun with L<sup>A</sup>T<sub>E</sub>X: Small and large grids]; pp. 11–18

Typesetting graph paper with L<sup>A</sup>T<sub>E</sub>X.

RALF MISPELHORN, Quintenzirkel mit Gitarren-Akkorden [Circle of fifths with guitar chords]; pp. 18–21

Typesetting guitar chords.

RALF MISPELHORN, Erstellen eines Songbooks [Create a songbook]; pp. 21–30

Songs collected while learning to play the guitar are compiled into a songbook using a Python script.

HERBERT VOSS, Erstellen, Ausführen und Einbinden der Ausgaben externer Dateien [Creation, execution and integration of the output of external files]; pp. 30–60

A tutorial on the `hvxextern` package by Herbert Voß. (A translation will be published in *TUGboat* 43:3.)

HENNING HRABAN RAMM, ConT<sub>E</sub>Xt kurz notiert [ConT<sub>E</sub>Xt briefly noted]; pp. 61–63

News from the ConT<sub>E</sub>Xt world.

PETER FLYNN, Druck oder Nichtdruck [To print or not to print]; pp. 64–68

Translated from Peter Flynn's *TUGboat* column, Typographer's Inn (*TUGboat* 41:3 (2020), 265–268). Translated by Patrick Bideault and Bernd Raichle.

JÜRGEN FENN, Neue Pakete auf CTAN [New packages on CTAN]; pp. 68–72

List of new packages on CTAN.

KARL-HEINZ OHNEMUS, Beyond the Archive – Von der Gießerei zum Klingspor Type Archive [Beyond the archive – From the foundry to the Klingspor type archive]; pp. 73–75

Review of the catalogue for the exhibition of the same name.

UWE ZIEGENHAGEN, L<sup>A</sup>T<sub>E</sub>X Beginner's Guide, 2. Auflage [L<sup>A</sup>T<sub>E</sub>X Beginner's Guide, 2nd edition]; pp. 75–76

Book review of Stefan Kottwitz's new book.

[Received from Uwe Ziegenhagen.]

[doi.org/10.47397/tb/43-2/tb134komo](https://doi.org/10.47397/tb/43-2/tb134komo)

## TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:

[tug.org/instmem.html](http://tug.org/instmem.html)

Thanks to all for their support!

American Mathematical Society,  
*Providence, Rhode Island, [ams.org](http://ams.org)*

Association for Computing  
Machinery, *New York, New York,*  
[acm.org](http://acm.org)

Aware Software,  
*Newark, Delaware, [awaresw.com](http://awaresw.com)*

Center for Computing Sciences,  
*Bowie, Maryland*

CSTUG, *Praha, Czech Republic,*  
[cstug.cz](http://cstug.cz)

CTAN, [ctan.org](http://ctan.org)

Duke University Press, *Durham,*  
*North Carolina, [dukeupress.edu](http://dukeupress.edu)*

Hindawi Foundation, *London, UK,*  
[hindawi.org](http://hindawi.org)

Institute for Defense Analyses,  
Center for Communications  
Research, *Princeton, New Jersey*

L3Harris, *Melbourne, Florida,*  
[l3harris.com](http://l3harris.com)

L<sup>A</sup>T<sub>E</sub>X Project, [latex-project.org](http://latex-project.org)

MacT<sub>E</sub>X, [tug.org/mactex](http://tug.org/mactex)

Maluhu & Co., *São Paulo, Brazil,*  
[maluhu.com.br](http://maluhu.com.br)

Marquette University,  
*Milwaukee, Wisconsin,*  
[marquette.edu](http://marquette.edu)

Masaryk University,  
Faculty of Informatics,  
*Brno, Czech Republic, [fi.muni.cz](http://fi.muni.cz)*  
Nagwa Limited, *Windsor, UK,*  
[nagwa.com](http://nagwa.com)

Overleaf, *London, UK,*  
[overleaf.com](http://overleaf.com)

StackExchange,  
*New York City, New York,*  
[tex.stackexchange.com](http://tex.stackexchange.com)

T<sub>E</sub>XFolio, *Trivandrum, India,*  
[texfolio.org](http://texfolio.org)

Université Laval, *Ste-Foy, Québec,*  
*Canada, [bibl.ulaval.ca](http://bibl.ulaval.ca)*

University of Ontario, Institute  
of Technology, *Oshawa, Ontario,*  
*Canada, [ontariotechu.ca](http://ontariotechu.ca)*

University of Oslo,  
Institute of Informatics,  
*Blindern, Oslo, Norway, [uio.no](http://uio.no)*  
VT<sub>E</sub>X UAB, *Vilnius, Lithuania,*  
[vtex.lt](http://vtex.lt)

## 2023 T<sub>E</sub>X Users Group election

### TUG Elections committee

The terms of TUG President and five TUG Directors will expire as of the 2023 Annual Meeting, expected to be held in July or August 2023. Three positions are open; thus eight are to be filled.

The terms of these directors will expire in 2023:  
Barbara Beeton, Paulo Cereda, Ulrike Fischer,  
Jim Hefferon, Norbert Preining.

Continuing directors, with terms ending in 2025:  
Karl Berry, Johannes Braams, Kaja Christiansen,  
Klaus H $\ddot{o}$ ppner, Frank Mittelbach, Ross Moore,  
Arthur Rosendahl.

The election to choose the new President and Directors will be held in early Spring of 2023. Nominations for these openings are now invited. A nomination form is available on this page or via [tug.org/election](http://tug.org/election).

The TUG 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 2023 must be paid before the nomination deadline. The term of TUG President is two years, and the term of Director is four years.

A list of informal guidelines for all TUG board members is available at [tug.org/election/guidelines.html](http://tug.org/election/guidelines.html). It describes the basic functioning of the TUG board, including roles for the various offices and ethical considerations. The expectation is that all board members will abide by the spirit of these guidelines.

Requirements for submitting a nomination are listed at the top of the form. The deadline for receipt of completed nomination forms and ballot information is

**07:00 a.m. PST, 1 March 2023**

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](mailto:office@tug.org); receipt will be confirmed by email. In case of any questions about a candidacy, the full TUG Board will be consulted.

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 T<sub>E</sub>X-related electronic media.

## 2023 TUG Election — Nomination Form

### Eligibility requirements:

- TUG members whose dues for 2023 have been paid.
- Signatures of two (2) members in good standing at the time they sign the nomination form.
- Supplementary material to be included with the form: passport-size photograph, a short biography, and a statement of intent. The biography and statement together may not exceed 400 words.
- 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 2023 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, dues payment) must be received at the TUG office in Portland, Oregon, USA, no later than

**07:00 a.m. PST, 1 March 2023.**

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.

- 2023 membership dues paid
- nomination form
- photograph
- biography/personal statement

T<sub>E</sub>X Users Group  
**Nominations for 2023 Election**  
P. O. Box 2311  
Portland, OR 97208-2311  
U.S.A.

(email: [office@tug.org](mailto:office@tug.org); fax: +1 815 301-3568)

## Calendar

### 2022

- Sep 11–16 XML Summer School, St Edmund Hall, Oxford University, Oxford, UK.  
[xmlsummerschool.com](http://xmlsummerschool.com)
- Sep 12–18 16<sup>th</sup> International ConT<sub>E</sub>Xt Meeting, Dreifelden, Germany.  
[meeting.contextgarden.net/2022](http://meeting.contextgarden.net/2022)
- Sep 20–23 22<sup>nd</sup> ACM Symposium on Document Engineering, San Jose, California.  
[doceng.org/doceng2022](http://doceng.org/doceng2022)
- Sep 23–25 Ladies of Letterpress EconoCon[ference], “More letterpress learning for less”, St. Louis, Missouri (in person and online).  
[ladiesofletterpress.com/conference](http://ladiesofletterpress.com/conference)
- Oct 15 *TUGboat* 43:3, submission deadline.
- Oct 28–29 TypoDay2022, “Typography for Children”, Hosted online by IDC School of Design (IDC), Indian Institute of Technology Bombay. [www.typoday.in](http://www.typoday.in)
- Nov 19 TeXConf 2022 (Japan; online)  
[texconf2022.tumblr.com](http://texconf2022.tumblr.com)

### 2023

- Mar 1 **TUG election:** nominations due, 07:00 a.m. PST [tug.org/election](http://tug.org/election)
- Mar 24 *TUGboat* 44:1, submission deadline.

- May BachoT<sub>E</sub>X 2023, “A model kit. Modeling and implementing text typesetting in T<sub>E</sub>X and other systems”, 28<sup>th</sup> BachoT<sub>E</sub>X Conference, Bachotek, Poland.  
[www.gust.org.pl/bachotex](http://www.gust.org.pl/bachotex)
- May Association Typographique Internationale (ATypI) annual conference, ATypI Paris, France. [www.atypi.org](http://www.atypi.org)
- Jun 26–29 SHARP 2023, “Affordances and Interfaces: Textual Interaction Past, Present and Future”, Society for the History of Authorship, Reading & Publishing. Hosted online by the University of Otago, New Zealand.  
[www.sharpweb.org/main/conferences](http://www.sharpweb.org/main/conferences)
- Jun 28–30 Twenty-first International Conference on New Directions in the Humanities, “Literary Landscapes: Forms of Knowledge in the Humanities”, Sorbonne University, Paris, France.  
[thehumanities.com/2023-conference](http://thehumanities.com/2023-conference)
- Jul 10–14 Digital Humanities 2023, Alliance of Digital Humanities Organizations, “Collaboration as Opportunity”, Graz, Austria. [dh2023.adho.org](http://dh2023.adho.org)
- Sep 5 The Updike Prize for Student Type Design, application deadline, 5:00 p.m. EST. Providence Public Library, Providence, Rhode Island.  
[prov.pub/updikeprize](http://prov.pub/updikeprize)

**Owing to the COVID-19 pandemic, schedules may change. Check the websites for details.**

*Status as of 1 September 2022*

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](mailto: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](http://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](http://tug.org/calendar.html).

*Science is what we understand well enough to explain to a computer. Art is everything else we do.*

— Donald E. Knuth



*the confluence of art and science of text  
processing in the cloud!*

- empowering authors to self-publish
- assisted authoring
- T<sub>E</sub>XFolio — the complete journal production in the cloud
- NEPTUNE — proofing framework for T<sub>E</sub>X authors

STM DOCUMENT ENGINEERING PVT LTD

Trivandrum • India 695571 • [www.stmdocs.in](http://www.stmdocs.in) • [info@stmdocs.in](mailto:info@stmdocs.in)

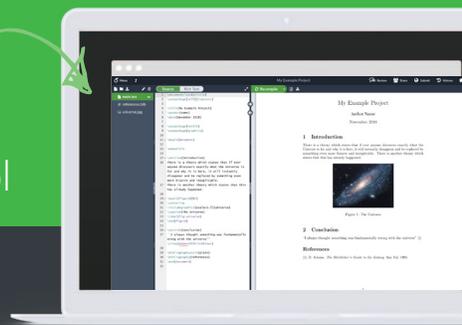
# Overleaf

A free online **LaTeX** and **Rich Text collaborative** writing and publishing tool

**Overleaf** makes the whole process of writing, editing and publishing scientific documents much quicker and easier.

#### Features include:

- **Cloud-based platform:** all you need is a web browser. No software to install. Prefer to work offline? No problem - stay in sync with Github or Dropbox
- **Complementary Rich Text and LaTeX modes:** prefer to see less code when writing? Or love writing in LaTeX? Easy to switch between modes
- **Sharing and collaboration:** easily share and invite colleagues & co-authors to collaborate
- **1000's of templates:** journal articles, theses, grants, posters, CVs, books and more – simply open and start to write
- **Simplified submission:** directly from Overleaf into many repositories and journals
- **Automated real-time preview:** project compiles in the background, so you can see the PDF output right away
- **Reference Management Linking:** multiple reference tool linking options – fast, simple and correct in-document referencing
- **Real-time Track Changes & Commenting:** with real-time commenting and integrated chat - there is no need to switch to other tools like email, just work within Overleaf
- **Institutional accounts available:** with custom institutional web portals



Find out more at [www.overleaf.com](http://www.overleaf.com)

## T<sub>E</sub>X 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](https://tug.org/consultants.html). If you'd like to be listed, please see there.

### Dangerous Curve

Email: [typesetting \(at\) dangerouscurve.org](mailto:typesetting@dangerouscurve.org)

Typesetting for over 40 years, we have experience in production typography, graphic design, font design, and computer science, to name a few things. One DC co-owner co-authored, designed, and illustrated a T<sub>E</sub>X book (*T<sub>E</sub>X for the Impatient*).

We can ■ convert your documents to L<sup>A</sup>T<sub>E</sub>X from just about anything ■ type up your handwritten pages ■ proofread, copyedit, and structure documents in English ■ apply publishers' specs ■ write custom packages and documentation ■ resize and edit your images for a better aesthetic effect ■ make your mathematics beautiful ■ produce commercial-quality tables with optimal column widths for headers and wrapped paragraphs ■ modify bibliography styles ■ make images using T<sub>E</sub>X-related graphic programs ■ design programmable fonts using METAFONT ■ and more! (Just ask.)

Our clients include high-end branding and advertising agencies, academics at top universities, leading publishers. We are a member of TUG, and have supported the GNU Project for decades (including working for them). All quote work is complimentary.

### Hendrickson, Amy

57 Longwood Avenue Apt. 8

Brookline, MA 02446

+1 617-738-8029

Email: [amyh \(at\) texnology.com](mailto:amyh@texnology.com)

Web: [www.texnology.com](http://www.texnology.com)

Full time L<sup>A</sup>T<sub>E</sub>X consultant for more than 30 years; have worked for major publishing companies, leading universities, and scientific journals. Our macro packages are distributed on-line and used by thousands of authors. See our site for many examples: [texnology.com](http://texnology.com).

■ *L<sup>A</sup>T<sub>E</sub>X Macro Writing*: Packages for books, journals, slides, posters, e-publishing and more; Sophisticated documentation for users.

■ Design as well as L<sup>A</sup>T<sub>E</sub>X implementation for e-publishing, print books and journals, or specialized projects.

■ Data Visualization, database publishing.

■ Innovative uses for L<sup>A</sup>T<sub>E</sub>X, creative solutions our speciality.

■ L<sup>A</sup>T<sub>E</sub>X Training, customized to your needs, on-site or via Zoom. See <https://texnology.com/train.htm> for sample of course notes.

Call or send email: I'll be glad to discuss your project with you.

### Dominici, Massimiliano

Email: [info \(at\) typotexnica.it](mailto:info@typotexnica.it)

Web: [www.typotexnica.it](http://www.typotexnica.it)

Our skills: layout of books, journals, articles; creation of L<sup>A</sup>T<sub>E</sub>X 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.

### Latchman, David

2005 Eye St. Suite #6

Bakersfield, CA 93301

+1 518-951-8786

Email: [david.latchman](mailto:david.latchman@texnical-designs.com)

(at) [texnical-designs.com](http://texnical-designs.com)

Web: [www.texnical-designs.com](http://www.texnical-designs.com)

L<sup>A</sup>T<sub>E</sub>X consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized L<sup>A</sup>T<sub>E</sub>X packages and classes to meet your needs. Contact us to discuss your project or visit the website for further details.

### L<sup>A</sup>T<sub>E</sub>X Typesetting

Email: [enquiries \(at\) latextypesetting.com](mailto:enquiries@latextypesetting.com)

Web: [latextypesetting.com](http://latextypesetting.com)

L<sup>A</sup>T<sub>E</sub>X Typesetting has been in business since 2013 and is run by Vel, the developer behind [LaTeXTemplates.com](http://LaTeXTemplates.com). The primary focus of the service is on creating high quality L<sup>A</sup>T<sub>E</sub>X templates and typesetting for business purposes, but individual clients are welcome too.

I pride myself on a strong attention to detail, friendly communication, high code quality with extensive commenting and an understanding of your business needs. I can also help you with automated document production using L<sup>A</sup>T<sub>E</sub>X. I'm a scientist,

designer and software developer, so no matter your field, I've got you covered.

I invite you to review the extensive collection of past work at the Showcase [latelytypesetting.com/showcase](http://latelytypesetting.com/showcase). Submit an enquiry for a free quote!

### Monsurate, Rajiv

Web: [www.rajivmonsurate.com](http://www.rajivmonsurate.com)  
[latexwithstyle.com](http://latexwithstyle.com)

I offer: design of books and journals for print and online layouts with L<sup>A</sup>T<sub>E</sub>X and CSS; production of books and journals for any layout with publish-ready PDF, HTML and XML from L<sup>A</sup>T<sub>E</sub>X (bypassing any publishers' processes); custom development of L<sup>A</sup>T<sub>E</sub>X packages with documentation; copyediting and proofreading for English; training in L<sup>A</sup>T<sub>E</sub>X for authors, publishers and typesetters.

I have over two decades of experience in academic publishing, helping authors, publishers and typesetters use L<sup>A</sup>T<sub>E</sub>X. I've built typesetting and conversion systems with L<sup>A</sup>T<sub>E</sub>X and provided T<sub>E</sub>X support for a major publisher.

### Sofka, Michael

Email: [michael.sofka \(at\) gmail.com](mailto:michael.sofka@gmail.com)

Professional T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X consulting and programming services. I offer 30 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X: Automated document conversion; Programming in Perl, Python, C, R and other languages; Writing and customizing macro packages in T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X, **knitr**.

If you have a specialized T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

### Veytsman, Boris

132 Warbler Ln.  
Brisbane, CA 94005  
+1 703-915-2406  
Email: [borisv \(at\) lk.net](mailto:borisv@lk.net)  
Web: [www.borisv.lk.net](http://www.borisv.lk.net)

T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X consulting, training, typesetting and seminars. Integration with databases, automated document preparation, custom L<sup>A</sup>T<sub>E</sub>X packages, conversions (Word, OpenOffice etc.) and much more.

I have about two decades of experience in T<sub>E</sub>X 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 T<sub>E</sub>X-related journals, and conducted several workshops on T<sub>E</sub>X 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 T<sub>E</sub>X manuscripts as well as typesetting of books, papers & journals, including multilingual copy with non-Latin scripts, and more.

### Warde, Jake

90 Resaca Ave.  
Box 452  
Forest Knolls, CA 94933  
+1 650-468-1393  
Email: [jwarde \(at\) wardepub.com](mailto:jwarde@wardepub.com)  
Web: [myprojectnotebook.com](http://myprojectnotebook.com)

I have been in academic publishing for 30+ years. I was a Linguistics major at Stanford in the mid-1970s, then started a publishing career. I knew about T<sub>E</sub>X from editors at Addison-Wesley who were using it to publish beautifully set math and computer science books.

Long story short, I started using L<sup>A</sup>T<sub>E</sub>X for exploratory projects (see website above). I have completed typesetting projects for several journal articles. I have also explored the use of multiple languages in documents extensively. I have a strong developmental editing background in STEM subjects. If you need assistance getting your manuscript set in T<sub>E</sub>X I can help. And if I cannot help I'll let you know right away.

---

## Reports and notices

- 86 TUG 2022 conference information and program
- 89 *Jim Hefferon* / TUG 2022 conference report
- 90 *Robin Laakso* / TUG 2022 Annual General Meeting notes
- 93 *Karl Berry* / David C. Walden, 1942–2022
  - for more on Dave's many lifetime projects, see [walden-family.com](http://walden-family.com)
- 96 *Paulo Ney de Souza* / interview with John Lees-Miller, CTO of Overleaf
- 100 *Paulo Ney de Souza* / interview with Boris Veytsman, scientist, T<sub>E</sub>X programmer and current TUG President
- 209 TUG 2022 abstracts (Austin, Blakesley, Castañeda, Chernoff, Cheung, Claudio, Fine, Gundlach, Hickman, Jimenez, Khalighi, Luc, Mariano, Moore, Ohri, Park C., Park E., Preining, samcarter, Schmah, Vrabcová, Wu)
- 214 From other T<sub>E</sub>X journals: *La Lettre GUTenberg* 45 (2022); *MAPS* 52 (2022); *Die T<sub>E</sub>Xnische Komödie* 2/2022
- 215 Institutional members
- 216 *TUG Elections committee* / 2023 T<sub>E</sub>X Users Group election
- 217 Calendar
- 218 TUG 2022 advertisements: STM Document Engineering Pvt Ltd; Overleaf
- 219 T<sub>E</sub>X consulting and production services

## Introductory

- 159 *Paulo Cereda, Phelype Oleinik* / The story of a silly package
  - development of the `sillypage` package, based on the Monty Python silly walks
- 155 *Éric Guichard, Jean-Michel Hufflen* / Introductory L<sup>A</sup>T<sub>E</sub>X workshop, en français
  - overview of this TUG'22 workshop, conducted in French
- 156 *Lloyd Prentice* / Self-publishing, L<sup>A</sup>T<sub>E</sub>X, and Markdown
  - roadmap for making L<sup>A</sup>T<sub>E</sub>X more usable for the vast self-publishing author population
- 108 *Dag Spicer* / A stroll through computer history at the CHM
  - overview of exhibits and resources at the Computer History Museum, [computerhistory.org](https://computerhistory.org)

## Intermediate

- 142 *Apu V, Rishi T, Aravind Rajendran* / L<sup>A</sup>T<sub>E</sub>X profiling of author submissions — completeness & usability checking
  - description of a pre-submission validation tool, using scripts and L<sup>A</sup>T<sub>E</sub>X3 hooks
- 207 *Karl Berry* / The treasure chest
  - new CTAN packages, April–August 2022
- 104 *Carlos Evia* / The future of technical documentation starts with its *recent* past
  - overview of DITA and family and relationship to L<sup>A</sup>T<sub>E</sub>X
- 127 *Island of T<sub>E</sub>X* / IoT theatre presents: The Tempest
  - T<sub>E</sub>X Live docker images, T<sub>E</sub>Xdoc online, Albatross, checkcites, arara past and future
- 148 *L<sup>A</sup>T<sub>E</sub>X Project Team* / L<sup>A</sup>T<sub>E</sub>X news, issue 35, June 2022
  - document metadata interface, `latex-lab`, marks, floating point
- 109 *Steven Matteson* / Type design: Catching up to the past
  - illustrated examples of adapting historical fonts for technological demands
- 172 *Chetan Shirore, Ajit Kumar* / The `luatruthtable` L<sup>A</sup>T<sub>E</sub>X package
  - using Lua to generate truth tables for general logical expressions
- 134 *Marnanel Thurman* / `yex`: a T<sub>E</sub>X-alike typesetter in Python
  - implementing T<sub>E</sub>X in the Python environment, focusing on HTML output
- 130 *Boris Veytsman* / Using `knitr` and L<sup>A</sup>T<sub>E</sub>X for literate laboratory notes
  - background, desiderata, and solutions for lab notes
- 162 *Joseph Wright* / Key–value setting handling in the L<sup>A</sup>T<sub>E</sub>X kernel
  - generic key–value and option support with both L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub> and `expl3` syntax
- 164 *Joseph Wright* / `siunitx`: Launching version 3
  - development support from standard L<sup>A</sup>T<sub>E</sub>X tools for a major revision
- 165 *Joseph Wright* / Case changing: L<sup>A</sup>T<sub>E</sub>X reaches Unicode-land
  - progress toward uppercasing, lowercasing, titlecasing, case folding, in full generality

## Intermediate Plus

- 176 *Oleksandr Baranovskyi* / L<sup>A</sup>T<sub>E</sub>X classes for doctoral theses in Ukraine: Interesting tips and painful problems
  - status, issues, and directions for `vakthesis` and `mon2017dev`
- 182 *H. Andrew Black, Hugh J. Paterson III* / XLingPaper's use of T<sub>E</sub>X technologies
  - use of X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X in an authoring tool for linguistic publishing
- 167 *Ulrike Fischer* / Using spot colors in L<sup>A</sup>T<sub>E</sub>X
  - new support in the kernel, and via the `colorspace` package
- 120 *Peter K. G. Williams* / The Tectonic Project: Envisioning a 21st-century T<sub>E</sub>X experience
  - overview of a Rust reimplement of X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X, targetting modern documents

## Advanced

- 202 *Hans Hagen, Mikael Sundqvist* / Pushing math forward with ConT<sub>E</sub>Xt lmtx
  - generalizing atom classes and spacing; breaking multiline formulas
- 136 *Jean-Michel Hufflen* / Extracting information from (L<sup>A</sup>)T<sub>E</sub>X source files
  - a generalized Scheme tool for external processing of L<sup>A</sup>T<sub>E</sub>X documents
- 197 *Dennis Müller, Michael Kohlhase* / A L<sup>A</sup>T<sub>E</sub>X-based ecosystem for semantic/active mathematical documents
  - structured math knowledge; RusT<sub>E</sub>X, a T<sub>E</sub>X reimplement in Rust to support MMT

## Reports and notices

- 86 TUG 2022 conference information and program
- 89 *Jim Hefferon* / TUG 2022 conference report
- 90 *Robin Laakso* / TUG 2022 Annual General Meeting notes
- 93 *Karl Berry* / David C. Walden, 1942–2022
  - for more on Dave’s many lifetime projects, see [walden-family.com](http://walden-family.com)
- 96 *Paulo Ney de Souza* / interview with John Lees-Miller, CTO of Overleaf
- 100 *Paulo Ney de Souza* / interview with Boris Veytsman, scientist, T<sub>E</sub>X programmer and current TUG President
- 209 TUG 2022 abstracts (Austin, Blakesley, Castañeda, Chernoff, Cheung, Claudio, Fine, Gundlach, Hickman, Jimenez, Khalighi, Luc, Mariano, Moore, Ohri, Park C., Park E., Preining, samcarter, Schmah, Vrabcová, Wu)
- 214 From other T<sub>E</sub>X journals: *La Lettre GUTenberg* 45 (2022); *MAPS* 52 (2022); *Die T<sub>E</sub>Xnische Komödie* 2/2022
- 215 Institutional members
- 216 *TUG Elections committee* / 2023 T<sub>E</sub>X Users Group election
- 217 Calendar
- 218 TUG 2022 advertisements: STM Document Engineering Pvt Ltd; Overleaf
- 219 T<sub>E</sub>X consulting and production services