# TUGboat

Volume 42, Number 2 / 2021
TUG 2021 Conference Proceedings

## TeX Users Group

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

### Individual memberships

2021 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`.

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 2021 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` or contact the TUG office.

### Trademarks

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

[printing date: September 2021]

Printed in U.S.A.

### Addresses

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

### Telephone

+1 503 223-9994

### Fax

+1 815 301-3568

### Web

`tug.org`
`tug.org/TUGboat`

### Electronic Mail

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

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

Volunteer technical support:
`support@tug.org`

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

**2021 Conference Proceedings**

TeX Users Group
Forty-second annual TUG conference
Online
August 5–8, 2021

# TUGboat

COMMUNICATIONS OF THE TeX USERS GROUP

| | |
|---|---|
| TUGboat Editor | Barbara Beeton |
| Proceedings Editor | Karl Berry |

Volume 42, Number 2, 2021
Portland, Oregon, U.S.A.

# TUG 2021 — Online — August 5–8, 2021

The forty-second annual TUG conference
`https://tug.org/2021` ▪ `tug2021@tug.org`

## Conference committee

Jennifer Claudio
Rohit Goswami
Steve Grathwohl
Jim Hefferon
Tom Hejda
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
DANTE e.V.
Google
Overleaf
STM Document Engineering Pvt Ltd
TEXnology Inc
The University of Adelaide
*with generous assistance from many individual contributors.*

Thanks to all!

# TUG 2021 program

(All times and days listed are UTC.)

| | | | |
|---|---|---|---|
| **Thursday August 5** | 16:00 | Cheryl Ponchin, IDA/CCR-P; Sue DeMeritt, IDA/CCR-La Jolla | *Workshop: Introduction to LaTeX* |
| | 18:00 | Emílio Kavamura, Univ. Federal do Paraná | *Workshop: Graphics with PGFPlots — in Portuguese* |
| | 20:00 | Alexánder Borbón Alpízar, El Tecnológico de Costa Rico | *Workshop: Introduction to LaTeX — in Spanish* |

*(continued)*

**TUG 2021 program** (continued)

| | | | |
|---|---|---|---|
| **Friday August 6** | 16:00 | Simon Porter, Digital Science | *Data-driven documents using Jupyter Notebooks and Overleaf* |
| | 16:45 | Amelia Hugill-Fontanel, Cary Graphic Arts Collection | *Cary Graphic Arts Collection Pressroom tour* |
| | 17:45 | Paulo Ney de Souza, BooksInBytes | *Producing a book for Amazon KDP* |
| | 20:30 | David Crossland, Google | *Variable fonts* |
| | 21:15 | Matheus Rocha | *How to make a logo/symbol for a font* |
| | 22:00 | Oliver Austin, San Jose, CA | *Plane and simple: Exploration of machine interaction with text type for visual based navigation systems* |
| | 23:30 | Vafa Khalighi | *Persian typesetting in TeX: Past, present, and future* |
| **Saturday August 7** | 00:15 | Norbert Preining, Fujitsu Research | *Interview, conducted by Paulo Ney de Souza* |
| | 06:15 | Aravind Rajendran, Rishi T, Apu V, Rahul Krishnan S, STM Document Engineering | *How a LaTeX-based company lived through a modern day pandemic* |
| | 07:00 | Martin Ruckert, Hochschule München | *The WEB to CWEB conversion of TeX* |
| | 07:45 | Vít Novotný, Masaryk Univ. | *Markdown 2.10.0: LaTeX themes & snippets, two flavors of comments, and LuaMetaTeX* |
| | 08:30 | Rohit Goswami, Univ. of Iceland | *Continuous integration and TeX with Org-Mode* |
| | 10:00 | Andreas Papasalouros, Antonis Tsolomitis, Univ. of the Aegean | *latex2nemeth: A direct LaTeX-to-Braille transcribing tool* |
| | 10:45 | Ulrike Fischer, LaTeX Project | *On the road to tagged PDF: about StructElem, Marked Content, PDF/A and squeezed Bärs* |
| | 11:30 | Frank Mittelbach, LaTeX Project | *Taming the beast — Advances in paragraph tagging with pdfTeX and XeTeX* |
| | 12:15 | Jonathan Fine, Retired, Open Univ., UK | *Code and math in the dark* |
| | 14:00 | Paulo Cereda, Island of TeX | *2020: A year in review, living on an island* |
| | 14:45 | Joseph Wright, LaTeX Project | *Any colo(u)r you like* |
| | 15:30 | samcarter | *News from the TikZ zoo: A short introduction to the TikZlings package* |
| | 16:00 | Marcel Krüger, LaTeX Project & Univ. of Hamburg | *Reviving Type 3 fonts for modern LuaLaTeX documents* |
| | 16:45 | Frank Mittelbach, LaTeX Project | *Interview, conducted by Paulo Ney de Souza* |
| | 18:30 | | *TUG Annual General Meeting* |
| | 19:30 | Keiran Harcombe, Open Univ.; Boris Veytsman, George Mason U. | *Creating a VPAT statement for TeX Live* |
| | 20:15 | Alexei Kolesnikov, Towson Univ.; Al Maneki; Michael Cantino; Rob Beezer, Univ. of Puget Sound; Volker Sorge | *Tactile mathematics: Enabling sighted and blind people to share mathematical experience* |
| | 21:00 | Michael Nolan, Todd Pagano, Suhas Chikkanaravangala Vijayakumar, Rahul Jaiswal, Rochester Inst. of Tech. | *Publishing for all: Using LaTeX to help improve the accessibility of an open-access journal* |
| | 23:00 | Ross Moore, Macquarie Univ., Tom Price, Univ. of Akron | *Accessible research reports; case study, including acronyms and glossaries* |
| | 23:45 | Jennifer Claudio, San Jose, CA | *Word search puzzles in Arabic, Cyrillic, and English using babel's multilingual support for LaTeX* |

**TUG 2021 program** (continued)

| | | | |
|---|---|---|---|
| **Sunday August 8** | 00:30 | Antoine Bossard, Kanagawa Univ. | *On typesetting an English–Japanese book* |
| | 01:15 | Vafa Khalighi | *Bidirectional typesetting in TEX: Past, present, and future* |
| | 07:00 | Mathias Magdowski, Otto von Guericke Univ. | *How to generate personalized tasks and sample solutions for anonymous peer review in electrical engineering using LATEX, PGFPlots and CircuiTikZ* |
| | 07:45 | Vic van Dijk, Set your Text | *R and LATEX: Typesetting graphs in a reproducible way* |
| | 08:30 | Jonathan Fine, Retired, Open Univ., UK | *Towards 21st century digital typography* |
| | 09:15 | Ondřej Sojka, Petr Sojka, Masaryk Univ. | *Czechoslovak hyphenation patterns, word lists, and workflow — Why hyphenate Czecho-Slovak simply syllabically?* |
| | 11:00 | Jean-Michel Hufflen, FEMTO-ST & U. Bourgogne Franche-Comté | *Programming bibliographies* |
| | 11:45 | Nicola Talbot, Dickimaw Books | *bib2gls: symbols* |
| | 12:30 | Heinrich Stamerjohanns, Texmlbus | *Texmlbus, a build system to convert documents to XML and other formats* |
| | 13:15 | Michal Hoftich, Charles Univ. | *LATEX to HTML conversion with TEX4ht* |
| | 15:00 | Andy Black, Hugh Paterson, SIL International | *XLingPaper's use of TEX technologies* |
| | 15:45 | Nicolás Vaughan, U. de los Andes | *TEI-XML to LATEX workflow: Issues and lessons* |
| | 17:15 | John Hammersley, Overleaf | *Interview, conducted by Paulo Ney de Souza* |
| | ≈18:15 | *end* | |

## TUG 2021 conference report

Arthur Rosendahl and Jennifer Claudio

As with TUG 2020, TUG 2021 was conducted entirely online, due to the continuing global pandemic. The conference spanned a total of four days, of which one had three workshops and the remainder of the days provided the presentation of talks on a 24-hour rotating schedule. This year, there were over 400 attendees from eighteen timezones (approximate map of attendee locations on p. 216). Speakers ranged in age from at least one high school student to long-time users of TEX in industry.

Like last year, the presentations were scheduled during reasonable waking hours for the speakers, wherever in the world they were, and we tried to group them into consistent sessions. A majority of the talks were given by speakers based in Europe, about a third in the Americas, and a few were in the Asia-Pacific region. This presented some challenges, which we tackled the best we could.

Reflecting the growing interest and activity in accessibility, two sessions were dedicated entirely to producing tagged PDF and mathematical documents for the visually impaired. One particularly interesting entry featured testimonies by blind speakers, one of them a long-term LATEX user. Some of the talks showed a lively interest in TEX from developers not primarily focussed on it.

XML was the topic of two half-sessions (originally a single one that suffered from some time zone changes). Here again, we were treated to a healthy and enjoyable mix of talks by speakers from both within and outside our community, who all used TEX-based processes for typesetting XML documents, sometimes as part of a larger workflow.

Workflows of different kinds and scopes were featured in a number of talks, with applications ranging from version control integration, via automated production of student assignments, to large-scale document processing for scientific publishing.

We had several talks about multilingual typesetting, unsurprisingly given the international make-up, and two talks about bibliographies.

A TEX conference wouldn't be complete without some talks about fonts, and although there were somewhat fewer of them this year, they were not missing: there was one talk about variable fonts, a second about applications to air travel, and a workshop about making a logo.

To help encourage the function of the conference as a means of idea exchange, the communication and media platforms of Zulip and Topia were offered.

Videos of individual talks will be posted on the TUG YouTube channel when they are available: `youtube.com/c/texusersgroup`.

◇ Arthur Rosendahl and Jennifer Claudio
`https://tug.org/tug2021`

## TUG 2021 Annual General Meeting notes

TUG 2021 Annual General Meeting notes

Current TUG president, Boris Veytsman opened the AGM at 18:30 PM UTC via a Zoom webinar. Attendance comprised thirty-four attendees and sixteen panelists on Zoom, with an additional fifteen viewers on the YouTube stream. Duplicate participation is unable to be validated.

In his introduction, Boris thanked the conference committee for their assembly of TUG 2021 online. He acknowledged that in most years the AGM occurs in person during the annual conference, but last year the AGM was skipped due to the online nature and the lack of accommodation for the situation. Since this was the second year that the conference was held online on the Zoom platform, the board felt it was desirable to offer the AGM online. Despite the lack of instruction or guidance for online situations, the bylaws indicate that the AGM should not be skipped, if possible.

Klaus Höppner, Secretary of TUG, provided a TUG status update and addressed the current state of TUG using a screenshare of his slides.

He reported that the current (outgoing) TUG board has 16 members [see addendum], and he reviewed a summary of the 2019 AGM. See accompanying slides.

Klaus also addressed the international conferences. TUG 2020 was originally planned to take place at Rochester Institute of Technology, but due to restrictions associated with the COVID-19 pandemic, an online meeting was organized instead by Paulo Ney de Souza, Arthur Rosendahl, Ross Moore, and others. Other conferences cancelled that year included DANTE 2020 and BachoTEX 2020. ConTEXt 2020 took place in the Czech Republic. GuIT 2020 in Italy and DANTE 2021 both took place online. The DANTE autumn meeting is intended to take place in person on September 18 in Germany and ConTEXt 2021 is scheduled for September 20–25 in Belgium.

The TEX Live/TEX Collection was released in 2020 and 2021 as planned. The team included Karl Berry, Norbert Preining, Siep Kroonenberg, Akira Kakuto, and many others. These were produced in cooperation with various groups including TUG, and the distributions included TEX Live, proTEXt (Thomas Feuerstack, Klaus Höppner), and a CTAN snapshot (Manfred Lotz).

Klaus reported board motions as follows:

2019.3: Klaus as secretary, accepted unanimously

2019.4: Lifetime Membership category in bylaws, accepted unanimously

2019.5: Rate changes, accepted unanimously

2019.6: Budget for 2020, accepted unanimously (no response: 1)

2019.7: Support for students at BachoTEX (accepted, but obsolete due to cancellation)

2020.1: Support for `learnlatex.org` (2000 USD), accepted (yes: 8, abstain: 4, no response: 1)

2020.2: Cancel (physical) TUG 2020 at RIT (accepted unanimously)

2020.3: OSI Affiliate membership, accepted (yes: 12, abstain: 1)

2020.4: Reduction of electronic membership discount accepted (yes: 12, no response: 1)

2020.5: Budget for 2021, accepted (yes: 11, no response: 2)

2021.1: Accept LYX donations, accepted unanimously

2021.2: Discontinue EduTEX committed fund, accepted (yes: 12, no response: 1)

2021.3: Charge a TUG 21 conference fee from non-members, rejected (yes: 3, no: 7, no response: 1)

2021.4: Discontinue libre font committed fund, accepted unanimously

2021.5: Barbara Beeton as registered agent in Rhode Island, accepted unanimously

After Klaus' report, the floor was opened to the general audience to express questions or comments by using the e-hand raising feature in Zoom, to speak using voice on Zoom, or to type into the chat console on Zoom.

At 18:42 PM UTC, Tristan Miller asked via chat: What are the benefits and responsibilities of being an affiliate member of the Open Source Initiative? To which Klaus responded that there is not an immediate or direct benefit from being a member of OSI, but that it helps express that TUG supports it. It is treated as more of a trust statement.

Rohit Goswami, via chat at 18:44 PM UTC, pointed out that he just noticed that TUG on the OSI site links to TODO Group instead of to `tug.org` (on `opensource.org/affiliates`) [see addendum]. He also asked if TUG made a financial contribution to the OSI? Boris responded that we don't pay a fee; Klaus confirmed that it is free of charge.

Herbert Schulz added a correction that the TEX Live release also included MacTEX on the DVD (18:48 PM, via chat function). Klaus addressed this comment by confirming that MacTEX was included in the collection.

Rohit mentioned that we are still streaming the YouTube and whether that was intended.

doi.org/10.47397/tb/42-2/tb131tug21-agm

Jonathan Fine requested to know about the financial report, to which Boris replied that as a non-profit TUG is obligated to publish a financial report that is posted publicly on the website. Boris stated that he will follow up with Karl to have that report posted on the website as the executive reports sent to the IRS [see addendum].

Jonathan Fine asked to deliver a motion, which was acknowledged by Boris. Boris stated that there is a problem according to the bylaws in that there must be fifty people physically present at the AGM to consider a motion. With the virtual format, by nature, this AGM did not have people physically present and furthermore were unable to verify that all participants logged in were current members in this setting. Consequently, Boris felt that he could not validate that we have a quorum. However, in the interest of transparency, he felt it was fair to listen to Jonathan Fine's proposal for a motion.

Jonathan Fine requests for the board to consider: 1) Reduction of the term for board members, such as by holding an election every other year; and 2) For the AGM to be adjourned for at least 35 days and at most 90 days.

In describing his proposed considerations, Jonathan stated that this would reduce the barrier to entry; someone who has something to contribute to the board for specific reason or if the board needs specific expertise means that the person is only committed to two years and doesn't have to wait any longer than one year to be on the board. The intention of this reduction is to increase vigor. The AGM does not control the bylaws, the board does. Jonathan expressed that he is annoyed about the comment from Boris that physical presence is needed, and that there is ample time to adjust the bylaws to facilitate online meetings. Jonathan further expressed that he doesn't know the US/Rhode Island situation, but he knows that in the UK, there has been accommodation to deal with online meetings.

Jérémy Just contributed the comment that in France, a law has been voted at the beginning of the pandemics to allow General Meetings to switch from physical to virtual, independently of bylaws.

Jonathan Fine requested that Boris should chair the meeting and not respond by himself. Boris has suggested that we have a second to the motion by Jonathan, but he states he can also let people speak.

Tristan raised his hand and upon acknowledgement thanked Jonathan Fine for bringing these up. He admits he hasn't been a member of TUG for very many years, but it seems that what is being proposed is eminently sensible. He has given some thought to becoming more involved in the group, and were that to involve sitting on the board, he wouldn't commit to such a long term. He suggested that others might be in a similar situation, such as due to job security over a 3–4 year period, whereas feeling more secure with a 1–2 year expectation of service.

Michael Doob raised his hand and also thanked Jonathan for raising the point. He mentioned that he has served on the board alongside Jonathan a number of times, and qualifies that it takes at least a year to get up to speed to know what is going on within the term. Michael suggested that it might be appropriate for a renewal of someone who is already on the board, but might not be good for someone who is initially entering the board. He disagreed with Jonathan by stating a four year term is likely in order, and that this is the case in other non-profits and boards as well. He additionally empathized with the feeling of not being able to raise motions and agrees that there should be something in place if virtual meetings happen in future years.

In the chat function, Tom Hejda replied with agreement toward Michael Doob's feedback.

Jonathan Fine thanked Tristan for his report. In response to Michael, he has found the working of the board opaque, e.g., he doesn't recall any discussion of the board on the website, and consequently expressed the opinion that people have an authority based on their service or roles, where it seemed to him that actions took place without the board's knowledge. He said that he had been following the board motions carefully and did not see discussion of cancelling the AGM or holding it this year. In connection to his proposal, Jonathan thought that a person joining for two years would have an opportunity for the board to improve its procedures, and that participation on the board should not result in a year of getting caught up.

Tom Hejda disagreed with Jonathan Fine regarding board term in that two years is nothing, and that in two years you barely know what you're doing.

Frank Mittelbach, hand raised, qualified his upcoming statement by saying he actively works with other organizations and boards and that he agreed with Michael in the practical sense that serving for a number of years is a sensible thing. He also reminded attendees that there is no hard obligation to serve all four years of a term. Although it is expected, a change in situation does not bind a person to the role, and there is always allowance to step down, which has happened in the past. He commented that the reason people don't join the board is not due to the four-year term, but rather that people prefer to receive than to contribute.

Jonathan made a clarification of the motion: The motion doesn't instruct the board to reduce the term of office or to hold elections every year. He has asked the board to consider those questions. The board is responsible for making that decision in the bylaws. He stated that passing this motion would only ask the board to consider the matter and to report back in the next AGM the results of that consideration. Rather than the motion being seen as a problem to be solved, he would like it considered as a possibility for improvement.

Ned Hummel requested speaking by raising his hand. He asked: For those of us who don't read and breathe the bylaws, just some basic questions of how big is the board (other than the mention of four years), i.e., how much of the board is turned over each election cycle?

Since this was a purely factual question, Boris felt it was fair for him to answer despite being the chair of the session. He reported that there are typically fewer than the maximum number of members due to people leaving the board. He explained that in the beginning, elections were made so that there would be some continuity, but because of departures, the last election changed many people on the board. As Frank mentioned, no one had stated that a full term is required to be served. Boris reminded all present that a vote could not be held on this proposed motion, but he was grateful to Jonathan for clarifying his request for the board's consideration.

Klaus additionally addressed the length of service of board members, in that approximately half of the board members stay for a very long time (e.g., he has been on the board since 2005), and he cited a few others who have stayed on the board for a long time. He pointed out that there are others who are in and out of the board for a couple years and then return, and of course that some only stay for one full term and do not serve again. He supposed that over half will renew their terms.

Boris pointed out again that we neither had the quorum nor an ability for attendees to cast a secure vote on this motion. Boris stated the intention to accommodate this by having a discussion with the board and the result of that discussion will be published. He explicitly stated that he did not want to violate the bylaws in any way.

Jonathan pursued the conversation, commenting: We have to make the best of where we are. We're not technically capable of voting on the first motion, and it's very important for motions to be voted on so that a decision can be made. He stated that he will say more about this and his second proposition in the coming months. He acknowledged that the discussion was very helpful and that it was good that people expressed interest in being on the board.

To follow up, Tristan asked, after raising his hand, would the present board members commit to making some arrangements such that there will be procedures in place for the next AGM to enable electronic voting? By chat, Arthur verified that he could make such a commitment. Boris also agreed to look into this possibility. Frank echoed the sentiment that we can assure Tristan that if there is a likelihood we end up with a third year of online conference only and not something like a hybrid (online + physical), then at least from his perspective, he could certainly help make sure that there is a method in place for voting.

Ulrik pointed out via the chat function that DANTE had online voting, and it worked very well. Via chat, Ross Moore commented: The trouble with doing an on-the-spot poll is that people in some locations are highly disadvantaged.

Steve Grathwohl pointed out via chat that Zoom has a polling function, but Paulo Ney de Souza reminded the audience that it does not have the legal force of a vote.

Besides voting issues, Tom Hejda raised the request that even if future conferences are in person, could we still include an online component, such that there would not be exclusion of online participants. Frank agreed that it is a profitable and positive effect of having dual in-person and virtual meetings.

Jonathan Fine finally suggested that the TUG board set up a working group online for online-conference-related matters.

In reply, Frank defended the board and its online conference committee by saying that we have already begun this and that the board had been working hard in even trying to get the conferences going. The board has had to learn their way to even work on these conferences and interfaces, and he expects that it is a growing experience. Whether it's a formal working group or not, he will pick up Jonathan's thoughts on this and bring it to the board for whatever form this will take.

Steve wondered if the legal status is different in the US and Germany, and Klaus clarified that DANTE had considered the Zoom polling for voting, but there were many disadvantages. Specifically, votes are counted per Zoom connection, i.e., there is no mapping for the use cases of multiple members within a household sharing a Zoom session, members that are both personal members and represent an institutional member, or voting as a proxy for another member. Ultimately, DANTE used an external, commercial voting system for the board elections during the AGM.

Boris thanked everyone for their contributions. Since he could not make a motion to adjourn due to lack of quorum, he closed the discussions of the AGM at 19:33 PM UTC.

◇ Notes recorded by Jennifer Claudio

**Post-conference addendum**

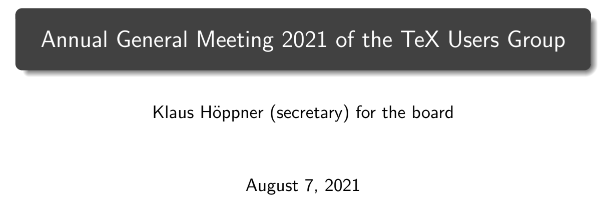
Regarding the "TUG board has 16 members" (p. 1): The start of the AGM is the moment at which a prior election takes effect. Since 2021 was an election year for TUG, at the start of the AGM the board went from the stated 16 members (15 directors plus the president) to the 13 members now sitting (12 directors plus the president). See `tug.org/board` for the current (and past) list of board members, and `tug.org/election` for the election report.

Regarding Rohit Goswami's remark that the TUG entry on `opensource.org/affiliates` links to the wrong site (p. 1, col. 2): The OSI has been contacted and the entry will be corrected; thanks to Rohit.

Regarding financial reports (p. 2): Tax returns are promptly published at `tug.org/tax-exempt` after being submitted to the IRS, which generally happens by August. Summary financial statements are published annually in *TUGboat*; the latest is at `tug.org/TUGboat/tb42-1/tb130treas.pdf`.

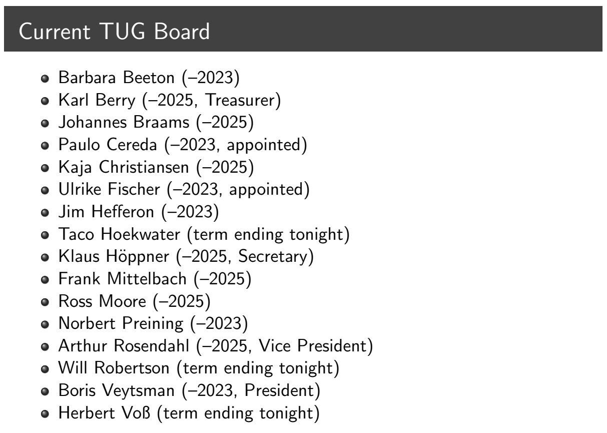
— — ∗ — —

Following are a selection of the slides presented by Klaus at the meeting.

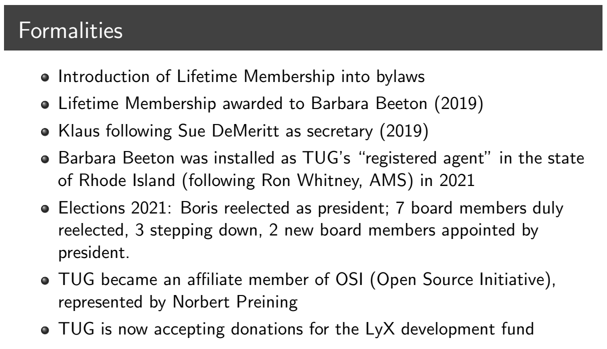

Annual General Meeting 2021 of the TeX Users Group

Klaus Höppner (secretary) for the board

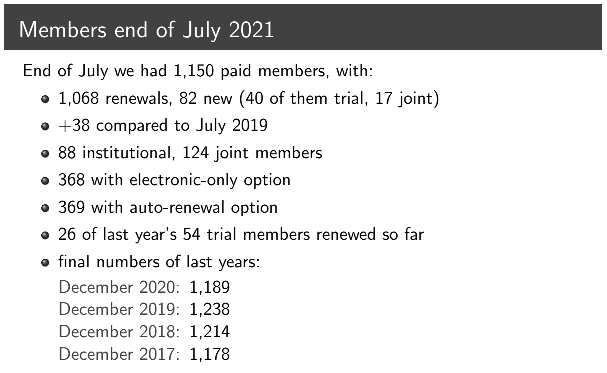
August 7, 2021


Current TUG Board

- Barbara Beeton (–2023)
- Karl Berry (–2025, Treasurer)
- Johannes Braams (–2025)
- Paulo Cereda (–2023, appointed)
- Kaja Christiansen (–2025)
- Ulrike Fischer (–2023, appointed)
- Jim Hefferon (–2023)
- Taco Hoekwater (term ending tonight)
- Klaus Höppner (–2025, Secretary)
- Frank Mittelbach (–2025)
- Ross Moore (–2025)
- Norbert Preining (–2023)
- Arthur Rosendahl (–2025, Vice President)
- Will Robertson (term ending tonight)
- Boris Veytsman (–2023, President)
- Herbert Voß (term ending tonight)


Formalities

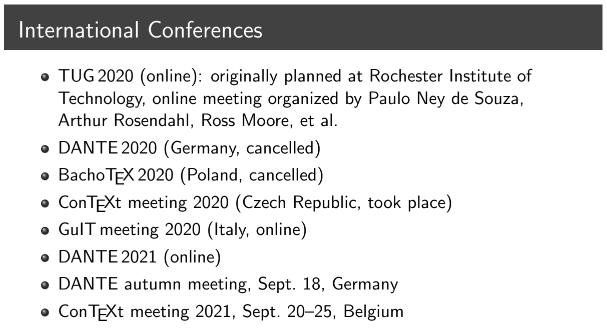
- Introduction of Lifetime Membership into bylaws
- Lifetime Membership awarded to Barbara Beeton (2019)
- Klaus following Sue DeMeritt as secretary (2019)
- Barbara Beeton was installed as TUG's "registered agent" in the state of Rhode Island (following Ron Whitney, AMS) in 2021
- Elections 2021: Boris reelected as president; 7 board members duly reelected, 3 stepping down, 2 new board members appointed by president.
- TUG became an affiliate member of OSI (Open Source Initiative), represented by Norbert Preining
- TUG is now accepting donations for the LyX development fund


Members end of July 2021

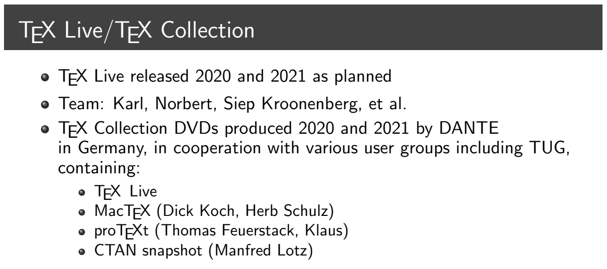
End of July we had 1,150 paid members, with:

- 1,068 renewals, 82 new (40 of them trial, 17 joint)
- +38 compared to July 2019
- 88 institutional, 124 joint members
- 368 with electronic-only option
- 369 with auto-renewal option
- 26 of last year's 54 trial members renewed so far
- final numbers of last years:

| | |
|---|---|
| December 2020: | 1,189 |
| December 2019: | 1,238 |
| December 2018: | 1,214 |
| December 2017: | 1,178 |


International Conferences

- TUG 2020 (online): originally planned at Rochester Institute of Technology, online meeting organized by Paulo Ney de Souza, Arthur Rosendahl, Ross Moore, et al.
- DANTE 2020 (Germany, cancelled)
- BachoTeX 2020 (Poland, cancelled)
- ConTeXt meeting 2020 (Czech Republic, took place)
- GuIT meeting 2020 (Italy, online)
- DANTE 2021 (online)
- DANTE autumn meeting, Sept. 18, Germany
- ConTeXt meeting 2021, Sept. 20–25, Belgium


TeX Live/TeX Collection

- TeX Live released 2020 and 2021 as planned
- Team: Karl, Norbert, Siep Kroonenberg, et al.
- TeX Collection DVDs produced 2020 and 2021 by DANTE in Germany, in cooperation with various user groups including TUG, containing:
  - TeX Live
  - MacTeX (Dick Koch, Herb Schulz)
  - proTeXt (Thomas Feuerstack, Klaus)
  - CTAN snapshot (Manfred Lotz)

# TUG 2021 conference organization reflections and recommendations

Paulo Ney de Souza and Jennifer Claudio

## Introduction

TUG 2021 showed improvement in several areas including more automation in the production of the web site, better security and handling of sessions on the Zoom platform, and broadening of subject and language coverage of the workshops.

This document summarizes the reflections by (some of) the organizing committee members. People referred to by name include: Jennifer Claudio, Tom Hejda, Norbert Preining, Arthur Rosendahl, Paulo Ney de Souza, Alan Wetmore.

## 1 Major points of agreement

- Dual in-person and online conference: an online-component committee should be separate from the local organizing committee.

  Paulo's specific recommendation are to build a kit with a projector(s), camera, lapel mics, mixer table, for use in the conference room and project to the online audience on a side-wall of the conference hall, and integrate them into the conversation.

  Jennifer prefers a slightly different arrangement, where we could have a pair of people designated to moderate through a Zoom account used to broadcast and relay questions from outsiders who log in, and that would be the single point of entrance into the conference participation.

  Tom favors a solution where the entire organization of the e-meeting part is left up to the local committee and that they should be free to organize in any way they choose.
- Chairing of the sessions was well done. Speaker introductions were well-conducted and chairpersons were good at asking at least one question or providing a comment for each talk. Having at least two chairpersons was helpful.
- The mixture of interviews, pre-recorded presentations, and live presentations worked well and should be preserved for future conferences.

## 2 Suggestions for improvement

### 2.1 Conference organization

To help orient the attendees to the conference environment and to set the tone that it is not just a string of webinars, but rather a collaborative experience, we should ensure having short opening and closing sessions that are marked on the schedule.

### 2.2 Speaker organization

The following idea was suggested by Paulo: The inclusion of a submission form would enable speakers to submit their titles, abstracts, photos, slides, videos, and supporting documents in one central hub, rather than having a person manually request each component. These would then be automatically deposited. This information could then be used to build a page for each speaker, listing his or her talk(s), affiliations, contributions to CTAN, GitHub, etc. This point was also raised by Tom.

Arthur strongly disagrees and believes it should all be handled by email, due to the small size of the conference.

Alan proposes the idea that a single point-of-contact could respond to initial submissions with other committee members backing up missed responses. Based on this year's volume of submissions, he has volunteered to send initial acknowledgements and would welcome reminders if he misses any. Besides this, Alan suggests that each submission could be tracked by date and a member for initial receipt, acknowledgment, etc.

Alan also suggests that, starting at the end of the submission window, a weekly update email to all would-be presenters would eliminate individual replies to panicking presenters worried about the dwindling time remaining.

Other suggestions relevant to speaker organization include:

- Make sure we use the reminder-to-speakers program built by Norbert and Paulo (TUG 2020).
- Include a briefing session for speakers to review Zoom tools for speakers (Jennifer volunteers to do this). Tom's original point raised here might also refer to providing audience members with an orientation regarding how to raise their e-hand or to request to speak or ask a question.
- Make audiovisual checks standard procedure prior to full conference days.
- Train chairpersons or have a separate "IT team" for troubleshooting issues within the platform.

### 2.3 Communication of information

Participants need to feel secure in terms of confirmation and followup before the conference. Suggestions to accomplish this:

- Make several minor (local) improvements for the registration form, attendees list, what's on, etc.
- Integrate the schedule and the rest of the website, as well as the what's-on page, so we have a single point of entrance to the conference.

doi.org/10.47397/tb/42-2/tb131desouza-tug21

- Create a submission page for prospective speakers, so they can upload their material and follow up on the process of approval and scheduling.
- Automate the generation of the new page from the metadata file.
- Make sure information in emails is well presented and well structured. Many people complained about not having the Zoom password. Perhaps it could be the first item in the email. The message was also almost certainly marked as spam for some recipients.
- List the conference in `researchseminars.org`.
- Produce Calendar files for Google, Outlook, iCal and Yahoo.
- Announce it through other TEX user groups around the world.
- Active social media presence of the conference on Facebook and Twitter.
- Prepare a production calendar with detailed deadlines for each task of conference organizing.

## 2.4 Compliance with privacy policies

The general recommendation is that TUG should come up with a conference *Privacy Policy* and *Terms of Service*, and that participants of the conference should have to agree to it at the time of registration. The strongest of these policies comes from the European side and protects three set of rights: the right to know what will be done with your personal information, the right to refuse its use, and the right to have the information removed upon request within a reasonable time frame. Among the proposed items is to make the publication of name/affiliation optional on the attendees list.

## 2.5 Conferencing platforms

The following things should be considered for future online presence, of which details here are described relative to the Zoom platform. Other platforms and social interactions are discussed in a later section.

- We should switch to using the Adelaide Zoom via the API. An example is that the Brazilian Math Congress was able to open 16 rooms and did this well. The advantages are as follows:
  1. No sharing of the Adelaide password.
  2. Open concurrent rooms for Workshops at the same level.
  3. Can automate the room opening and handout to the chairs.
- We should ensure that transcripts are available, by enabling the option within Zoom. However, automated transcripts are of poor quality with our TEXnical material.
- We should take precautions to avoid overly long YouTube sessions. Specifically, closing the feed after each block of sessions should help mitigate the potential for losing the YouTube records. (YouTube does not preserve a session lasting more than 12 hours.)

## 2.6 Post-production

Automate post-production. Norbert has automated the post-production of the videos as much as it can possibly be done. We should build the scripts that upload the videos to YT. There are many scripts already — it is a question of adapting them to use our metadata and our channel settings.

## 3 Communication media and social interaction options

Social interaction on the Zoom platform, especially in the webinar form, is not inherently easy. Attendees may not feel comfortable communicating or holding conversation through the main stage platform (Zoom), or may benefit from informal small group follow-up conversations. As a result, additional platforms Zulip and GatherTown were used as out-of-Zoom options in 2020 in order to allow more natural interactions between attendees. Zulip provided the platform at no cost to TUG.

## 3.1 TUG 2021

Zulip allowed continued free use to TUG, however had a low rate of new signups in 2021. Concerns about it included that it was hard to follow strings of conversations and announcements and it was not well organized. An advantage of it is that it allows conversation and connections to persist beyond the conference.

GatherTown was not offered in 2021 due to its change in pricing plan and restrictions to access.

Topia was found as an alternative to allow social encounters, but had criticism regarding ease of use and its style or features. To help facilitate virtual social gathering, setting times in the schedule designated as coffee or social breaks, rather than just a general break, may make this platform more fruitful. An earlier introductory or orientation session to Topia might also remedy the issue of people looking for each other, not finding each other, and consequently exiting. A future proposed alternative to Topia may be Wonder since it has a more straightforward interface without extra cuteness.

Slack and Discord have been suggested as alternative clients for the conference. Slack is growing in popularity for businesses and organizations as a self-contained communication medium; however, it comes with a heavy annual price tag in order to have full access to it. Discord is similar to Slack in terms of functionality for chat, voice, and video, and it is free. Discord, however, is widely used within the gamer culture and does not have as strong an identity in the professional world.

## 4 Survey of attendees

A short anonymous and optional feedback survey was administered from 10 Aug until 14 Aug to all participants. Forty-nine responses were collected. Most responses expressed positive feelings toward the conference.

Regarding future conference participation, 41/49 indicated that the respondent would likely recommend participation in future TUG conferences to friends or colleagues, regardless of whether the conference was held online, and only one indicated that they would not recommend it. Similarly, 42/49 would recommend online participation in the future. Of note is that while seven people provided an intermediate response for TUG conferences in general, only four did so for prospective TUG online, and three total responded negatively toward this.

Praise earned by TUG 2021 online included:

- Accessibility without travel
- Breadth of topics
- Representation of updated work
- Organization and delivery
- Prompt availability of recordings
- Length of talks was just right

An aspect in which we will need to consider improvements is that nearly 69 percent of participants were unable to view or attend all of the sessions that they wanted. Self-reported reasoning for this was mainly attributed to the time zones during which talks were offered and the quantity of personal free time (including household commitments).

Desired quantity of interaction was spread across the board, with almost equal numbers of responses for too much, too little, and just right. Written feedback was also divided, with several other suggested platforms named. In this section, several comments also stated general dissatisfaction with the talks, or that their expectations for some talks were not met.

In the free-response section, several pieces of constructive feedback were provided, summarized into these categories:

- Incorporate digital attendance, even if the conference is in person.
- Create a url for each talk, such as by using GitHub.
- Increased workshops for non-English speakers.
- Improve speaker presentations (audio issues, content delivery).
- Make TUG more attractive.
- Earlier call for proposals.
- Avoid issues/conflict during the AGM.
- Use free or non-proprietary software as a platform.
- Clarify the use of the social platforms early on.
- Manage both YouTube and Zoom participant commentaries.

## 5 Proposed course of action

- Discuss strong points of disagreement.
- Create guidelines or handbook for online component of TUG.

### 5.1 Shorter talks, longer breaks

Paulo: this would remove the scattered breaks; if a speaker slightly overshoots the time, it would still be fine. Longer breaks would allow for more discussion time, for better or worse. The current setup felt very discontinuous to me.

Jennifer: The Zoom environment doesn't naturally promote a lot of discussion even when the time to discuss is offered. One solution that has been proposed a few times is the use of breakout rooms, but this is very unnatural and results in extensive work by the hosts since it cannot be assumed that all attendees would know how to enter a breakout room or that they would have an up-to-date version of Zoom which allows it.

Also, a person who thought they were interested in a potential discussion might end up in a room where others are talking more one-on-one since not every speaker is trained in inclusive talking management, and consequently end up "stuck" in an irrelevant discussion that is not convenient to leave and without being able to easily communicate with the host once in the breakout.

### 5.2 A closing thought

At the price of one evening session, we could allow people from Asia/Pacific to participate live. For instance, 8pm Rochester (New York) is 8am Malaysia, 9am Japan and 10am Australia East coast, all quite bearable.

◇ Paulo Ney de Souza and Jennifer Claudio
`https://tug.org/tug2021`

## Interview with Norbert Preining

Paulo Ney de Souza

This interview took place on 7 August 2021, during the TUG 2021 online conference.

**Paulo Ney de Souza (PN):** Good morning, Norbert. Are you awake, did you take your coffee already?

**Norbert Preining (NP):** Yes, and actually you see my cup.

**PN:** Oh, my God. The eternal BachoTEX!

**NP:** Yes. I still have two cups of those so I'm quite happy.

**PN:** Good time. How are you? Are you ready for heavier conversation.

**NP:** I'm completely fine already. I think most others are asleep, right?

**PN:** Yeah, well the Europeans, probably. I see Jerzy [Ludwichowski] here and I see a few other guys, Herb Schulz, so there are a few Europeans up.[1]

Well, I'm not even going to spend the time introducing Norbert because most of the people here know him way more than I do. And for the ones that don't know him and are seeing him here for the first time, he's a mathematician, a logician specifically, and an accomplished programmer with manifold interests, and I'd like to talk to him about a few things because this guy has many hats, including TEX Live. If there is a manager of TEX Live, he's probably the one.

**NP:** No, no it's Karl [Berry], not me.

**PN:** Well, but Karl has so far refused to have an interview...

**NP:** Yes.

**PN:** So what I would like to do, Norbert, is start out where that interview that you gave to TUG stopped[2] and...

**NP:** That's a long time ago!

**PN:** Yes, yes. And I'd like you to tell us what brought you to do this trip to the Far East. It's a trip that's in the reverse order most scientists, which comes...

**NP:** There are a lot of scientists in Japan! They get a lot of Nobel prizes, I would say.

**PN:** From Europe and from the US?

**NP:** I think it's on the same level.

**PN:** On the same level? Oh, that's interesting. So what brought you specifically for...

**NP:** It was back in 2008 or something. I was thinking about leaving academics completely and working full time as a mountain guide because I'm also a professional mountain guide, and at that time, Professor Ono from the JAIST[3] here, close by where I live, invited me to come over and work as associate Professor there, and it was of course tempting.

I mean, in Austria, there was always, I mean, you always have, every two to three years, you have to write project applications to fund your own salary and all this kind of stuff. So that's the reason I thought work as a mountain guide is easier. And then, when the option to go to Japan, well, at least for a few years, it was interesting, yes. And I, of course, I thought a lot. I wasn't that young back then. (I'm much older now.) And I thought about, well, I'll just move into it completely. I have been here before, I mean I've been in 2006 and 7, each one month for research work here, so I knew what to expect. Well, I thought I knew what to expect. That was more or less the way I came here. So it was for work, for university work.

**PN:** You are on a new job these days, is that correct, or not?

**NP:** Well, since quite some time, since 2016; in 2016 the JAIST decided, in the spirit of internationalization, to throw out *all* foreigners, besides those in the English department. That was especially very much welcome, because in the very same month my daughter was born, so it was a good coincidence, to go into unemployment in Japan with a newborn

[1] Editor's note: Herb is actually in the Chicago area, so not all Europeans.

[2] `tug.org/interviews/preining.html`, 2011.

[3] The Japan Advanced Institute of Science and Technology

child. Yeah. And then after about five to six months I joined the company in Tokyo in the research and development lab. And just this year in February I changed to Fujitsu Research.

**PN:** Wow! Can you tell us, I mean do you apply logic in there, was it a complete separate part of your life as well, or...

**NP:** In the previous company, it was more or less completely separate. I mean, I did... Nowadays, what I do a lot is machine learning, right? Like most people. So there is not much logic there. Now within Fujitsu Research and I'm working on Neuro-symbolic AI, so this is something where my logic background and TCS background is hopefully merged with work in machine learning. So that's it.

**PN:** So does Fujitsu support basic research or is it...

**NP:** Yes, yes, well... basic research; I would say they want, of course, return of investment, right? In some sense. No, they do, really do. They support a lot of research, in a lot of areas, so really also basic research. Of course, you have to always argue. But I know several people doing this. From February, when I joined, to the end of March, Fujitsu Research was a separate company. It was only for research, and it was just merged. It was, everything is rebuilt currently in Fujitsu, so it was merged into the main company directly under the top officials. But yeah, basically it's the same. There is a lot of research going on. Of course, I mean, it's company research. You cannot expect doing whatever completely esoteric things; you have to at least explain why it is useful for Fujitsu, right?

And the stuff I do since 30 years, my research, my personal, I just continue in my private time.

**PN:** Uh huh, uh huh. Are they still heavy on computing?

**NP:** Yes.

**PN:** I remember, you know, when I started my computing life and my first laptop was a Fujitsu laptop; I would only go with Fujitsu. And I'll see them in the market, these days, but...

**NP:** They are still in the market with hardware, but it's, of course, so within Japan, but also in Europe in several places, there are big companies who have only worked with Fujitsu machines. There is a lot of service, especially public service. So from the state, from the government, from local governments where Fujitsu is very strong here. So yes, it's still a big company, I mean worldwide, in America, Europe, so they have parts in Sunnyvale, in Madrid, and in London and a few more places in Asia, I guess.

**PN:** So let me ask you, let me change directions a little bit, and ask you about something that always intrigued me. You seem to have a lot of interests in open source material.

**NP:** Yes!

**PN:** I mean...

**NP:** Not only.

**PN:** Let's put aside mountaineering, which we can enjoy by going over to your blogs and your website and so forth, and see, you know, pictures of stuff that I would never be able to visit in my own lives, but they're amazing to see, But you seem to have a lot of interest in open source software outside TeX.

**NP:** Yes...

**PN:** Where does it come from? Is it political? Is it...

**NP:** Good question.

Most of the time, I guess, I come around these projects because I'm not content with the current state, and I want to fix it, and that simply works with open source quite nicely, and it doesn't work well with any other source. So that is most of the time one of the big driving forces of me getting into something.

For example, one of the recent projects is this smart speaker. With FOSSASIA we're developing a privacy-aware smart speaker and personal assistant. That's software that runs on your RaspPi. You can talk to it, you get answers without Google or Amazon registering everything. It's just because I'm so fed up with Google and Amazon because they're so stupid. I have devices in every room, more or less, because my wife likes to talk to them, and my daughter, and get music back.

Most of the time it doesn't work out, so there are things I want to fix, and then open source projects are nice, right?

**PN:** Aha! So there is a personal need to make something work. I sometimes stop by your blog and your web page, and when I see the description of the projects you're involved with, I get tired only from reading them. So long it is...

**NP:** Yeah, but, for example, the biggest one I recently got involved in is the KDE/Plasma desktop for Debian, because I was so fed up, because in Debian, it was horrible old, it was completely outdated, and nobody did the work, and so I mean it was like two years back, and it wasn't really what it is; well, then I do it myself. Right?

And I did it first all myself, without the team, outside of Debian just for my own. I made all the packages, 300, 400 packages all for my own, because I wanted to have an up-to-date, recent, properly working system and then other people started to use it, and, well, it took two years and now, well, hopefully it's in much better shape in Debian. So that was a lot of time to adjust, but it's most of the time that I'm just not content with the current state that I start to fix things.

**PN:** But don't ... tell me one thing. How do you juggle this all? Do you get home and work until midnight?

**NP:** No, I normally get up at three or four o'clock in the morning.

**PN:** And how do you find time? How do you, ..., for example, tell me how many hours you put in on TEX Live, for example, every week.

**NP:** On TEX Live, not that many any more. It is, I mean, there were huge peaks in 2017, 18, 19,[4] but since then it has steadily decreased like what is necessary. I mean, bug reports are getting less and less, also new feature requests are getting less and less, right? So there is not so much to do any more.

The daily grind work of updating packages is mostly done by Karl unless Karl is on holiday or somewhere, then I do it. But I mean also this is mostly automated, so since we're doing it, Karl and me automatically, because at the end it's just a lot of repetition. I mean it's just emergency fixing for problems, like when was it recently, two weeks ago, when the GPG key[5] again expired and one part wasn't updated, and so these are the things. I wouldn't say that I put in TEX Live more than three to five hours a week, or if that would be a lot actually, I guess.

**PN:** So you consider it to be in steady maintenance mode, or no?

**NP:** Yes, yes, at the moment. I mean there are several things I have started working on, but I never could convince myself to finish, within TEX Live; also because there was never the need for it.

I mean Karl puts much more work in it, right? I mean the daily update of packages is mostly all due to his work.

**PN:** Which involves taking the package, installing the package, and...

**NP:** Yes.

Well, this is also 90–95% automated, right? I mean, we have written several scripts that automate all these updates and unpacking and repackaging and running LATEX and everything, so that, at the end a package for TDS (the TEX directory standard) comes out. But this is like mostly...

We need to do work for new packages or very special packages, but this has to be done only once for new packages and not permanently. So it works quite nicely at the moment.

**PN:** We've got a guest.

[There is an interruption here as Norbert's daughter, Haruna, makes an appearance. Her father tries to shoo her away, not entirely successfully, but she finally gives up.]

**PN:** That was beautiful. Beautiful that that no one can integrate this way.

**NP:** It sounds beautiful but, in reality, sometimes I wish I could go to an office because concentrated work is harder when you... That that's the reason, what I told you before, I get up at three or four o'clock in the morning, because then I have at least three to four hours of peaceful work.

**PN:** Yeah, but how long do you take to commute to work and commute back?

**NP:** I don't commute. Since 2016 I'm fully remote.

**PN:** Fully remote.

**NP:** Fully since 2016, so long before corona. Because my previous company was in Tokyo and I'm not in Tokyo, so I (and I don't want to move to Tokyo; I'm actually on the other side of the island), so with the previous company, I visited the company like once, twice a month for Tokyo. And now at Fujitsu I would normally also go once, twice to Tokyo, but I mean with corona, everything is shut down; no traveling.

**PN:** So it's unimaginable that the Tokyo subways and metro are, you know, are not being used at the moment but...

**NP:** They are used, of course. They are completely filled, of course. I mean completely full? — I don't know, but I guess that many people still have to go to work.

**PN:** Cool.

So, since our conversation is handed over to programming and TEX Live and so forth, can you know what can be expected on the year?

**NP:** I think nothing considerable changes. Nothing changes in the next years. I think the biggest change over there next — at some point in the future we'll be looking at 64-bit Windows, I guess. Something I tried to support, but I gave up because it's too much.

So that would be something that lots of people are asking for. Within the Japanese community,

[4] 2007–2009; spoken incorrectly in the interview

[5] The GPG key used to sign TEX Live packages.

actually, this is quite close due to the large fonts and often 32-bit Windows just runs out of memory. So there seems to be problem with huge CID-keyed fonts. Especially if you load several, I mean, of them for different stuff, so there is some need for this that might be something that comes in the future.

We don't know actually how. Maybe we just drop 32-bit and only work with 64-bit, but I mean there are a lot of people who still say that they want to keep 32-bit. And the program is really... I mean that is hard work. I mean the most painful part of the code are the special cases for Windows, and duplicating and making this work for 32- and 64-bit platform for Windows is a challenge. I've started three or four times; I have still the branches in my git repository somewhere. But then they [were] never realized, also because I have, well, recent with Fujitsu, I started to have to use Windows nowadays, but I myself hardly don't use Windows. It's hard for me also to test things.

**PN:** Mm hmm.

It's just we got into the subject, so let me ask you another question. The Gartner Group estimate that more than 80, more than 80% of the world shipping of operating systems, right now, do not run TeX, the situation which is...

**NP:** Gartner say this? Really! Wow.

**NP:** Yeah.

**PN:** This is like, you know, Fifteen years ago. Fifteen years ago we didn't have Android, we didn't have iOS...

**NP:** But we have TeX running on Android, that's not the problem, right?

**PN:** Well, we do have some sort of TeX running on Android.

**NP:** It's not integrated at the level you would like to see it, I agree. But...

**PN:** It's not that bad. That's what I want to see.

**NP:** But actually, there is some Chinese have, don't they have some application, something where they... I'm not sure. Well, the problem is with the security obligation of this. I mean it's actually getting more and more complicated, even on Macs: I remember one and a half years ago there was a discussion whether we can actually install TeX Live going further, because Apple removes interpreters, locks down installation. It's getting harder and harder. I mean if people want to use hampered and devices controlled by big companies, then please go ahead, yes. But it's getting more and more difficult for volunteers to support this.

**PN:** But do you see a time that it's going to be difficult to acquire a machine that that you'll be able to install, you know, an operating system on it and be running when all...

**NP:** No, you can always install Linux, right?

**PN:** Well, as I said, you know that the situation fifteen years ago was such that every single operating system that was shipped in 2000, fifteen years ago every single operating system shipped ran TeX. With iOS and Android, the situation is such that they kind of run TeX, but it's not the kind of TeX you want to use, so it's a, if you're really with a tablet, you know, on the couch, and you want to do some TeXing...

**NP:** Yes, I completely agree. I mean, but that I think in principle is a question of developing a nice user interface. I think with Android, at least, I mean at least me... I'm not sure. I use Android, but I'm just a dummy user. But I guess, as a rooted device, you can do everything. The question is how to interface with that, right? You want an application where you can just type. You can run, I mean you want an Emacs, right?

Not the whole Emacs, but what you want is to edit and kick off compilation from the edit screen and automatically switch to a PDF viewer. Yes, that would be nice, right? But that is something an Android application developer has to provide, right? This is not something...

I think the actual code, the binaries or something, you can compile on Android. I think there are several projects compiling for Android. And iOS should be also, I mean, we have binaries for MacOS, and so I, well, it's not easy compiling for a different hardware architecture. It's not that trivial, especially with iOS — with Apple just throwing blocks at developers — but in principle this should be possible, right? It's just who is doing it, right?

I mean the lack of expertise, right? We had some people who could do this, but I mean they also have their private life, or whatever. As long as nobody steps forward and say, "Okay, I know Android. I can develop on Android, and I'm also pro-open source. I'm writing an Android application that I don't sell and/or just offer", then it won't happen.

Always like this, you have to have the interest and fix it yourself, right?

**PN:** Right.

**NP:** Well, I say this.

**PN:** Yes, I guess so, so yeah. Somebody said that if you want something done, you should do it yourself. I don't know exactly who said it.

**NP:** And since my phone is small, I won't run Android on it, because typing on this is anyway a pain, with my clumsy fingers.

**PN:** So, so let me, in talking about that, which puts me on a parallel track...

Is there anything that you think that should be done with the TEX Live on the distribution streams? You know if you run, you know, TEX Live from Ubuntu, you'll get something which is like, you know, five years old, and to get TEX Live from Solaris is 12 years old.

**NP:** Okay, yeah, there are a few things that would be great. So the biggest problem at the moment is that packagers are stupid. Unfortunately, we have to say. There are too many, and I'm allowed to say this because I'm also a packager, right?

**PN:** Okay.

**NP:** And I answer all your questions on the TEX Live distro list about the packages, so... The problem is, most of the packagers are not really *users* of, I've seen this often, not users of TEX Live, they just want to package it without really understanding how stuff works. I agree, this is complicated then, and leads to disasters. We have seen too many times.

I also agree that the installation of TEX Live is not trivial. It could be. So why? Because everyone is expecting `configure; make; make install;` ready. Right? That is, a lot of people are expecting, and whatever, I mean it could be `cmake` and then `make` or whatever, so there are a lot of options.

So there we are far away from that; that is something. I mean the compilation — the binaries — they can be made like this, but then, a lot of handwork — I mean, not a lot of handwork — but a lot of stuff — I mean you have to merge the main tree with the built binaries, and you have to link some binaries, so it's not completely trivial. That would be something that would help, and that has been requested a few times that would help packagers and distributors to be more up to date, easier to include. Of course, there was a proposal from some in the group to switch to a different build system like whatever `ninja` or `cmake`, which is a nice idea, but nobody has ever come up with an actual replacement of the current system, so...

It's also a problem that the build system was written by Peter [Breitenlohner], who unfortunately passed away, so we have a serious lack of... Because none of us, I mean I can do autoconf, and Karl, of course, is an autoconf guru, but none of us... I mean this build system is quite involved.

So yes, there are some things for distributors that can be improved, I agree. There is, if I would have time, energy and interest. But for Debian I know how to package it myself, right? And, and for the others, I mean, I cannot care for others, they have to learn. It would be nice if you provide an easier way like `configure; make; make install` and everything works; that would be nice. The problem is often with the splitting of — because it's so big, right? Because we are now at seven gigabytes, and most distributions split it at whatever borders seem right. And the problem is, then, if you split it, you have to take care for whether formats and font map files are properly installed or not, or activated, and if they are activated but not available then you'll get errors and these are this stuff. I actually talked about this, I don't know, during FOSDEM or something, how to package it.

Yes, I agree that there are things; the problem is that the TEX system as a whole is, I mean not the whole packaging, the whole way how things work, is quite old. I mean this `fmtutil.cnf` and `updmap.cfg`. Over the last years, now last five years, I've worked that we can use multiple of them (the configuration files), which makes it already much easier for distributions to ship file and users to add additional fonts and this kind of stuff. It still isn't perfect and it doesn't work out for all kinds of settings. So yeah, there are things to do to make it easier for distributors.

But, for example, if you say Ubuntu, Ubuntu is not that far behind. The problem is with the release schedule of Ubuntu and TEX Live; also the release schedule of Ubuntu, Debian and TEX Live doesn't line up and it means they're always at least one year behind.

**PN:** Yeah.

**NP:** But the one year behind, it's often only the binaries and what they say, right? In Debian, for example, we did the last checkout from the CTAN `tlnet` in I think December, and that it was going into Bullseye[6] so the status of December 2020 is going into the next release; that was more or less the last cutoff date before the freeze.

Right. So it's not that old of course, well then, there are two years where this distribution is frozen more or less, and then you cannot update it besides critical. But on the other hand, now be honest, most people only have "updatitis", right? They want to get updates every day; it's not that they need it. I mean 90, I guess 99% of the TEX documents run even on a five-year-old TEX Live installation I guess.

**PN:** That's correct.

[6] The code name for the Debian 11.0 release, made on August 14, 2021.

**NP:** So it's just "Why do I have TeX Live 2019 in Ubuntu? No, it's so old!" And they want something new, not that it's really needed for them, right? But okay.

**PN:** Well, I'll tell you something in this interview; it's not me interviewing myself, but I'll tell you something, an area that does require update of all the binaries. We were working with submissions by people that are submitting to a proceedings of a conference and working on continuous submissions, so that they deposit their files on DropBox and they write their books and write their papers on DropBox or on Overleaf, and on Overleaf it is not a problem; they write it completely within Overleaf and we use the Overleaf environment. But for people who prefer to use their own TeX on their own boxes using DropBox, for example, I mean the very big problem, the initial problem, we tell them, you have to update `biblatex` to the latest `biblatex`, otherwise...

**NP:** Well that's true. You're completely right, of course, there are some areas — `biblatex/biber` is one — where you want newer versions. I agree completely, right? But honestly, I write also a lot of TeX code, but I have still haven't used `biblatex` for any of my documents right.

**PN:** Oh, you *should*!

**NP:** Yes, but I have already all stuff in a BibTeX `.bib` file and I'm just too lazy to start using it, because it works any way so...

**PN:** Alright, this was so nice. Tell me, when we are going to see you again in person? At BachoTeX?

**NP:** At BachoTeX, well, that would be nice, but BachoTeX is always when — I told you before — when there is one of the three holidays in Japan this is this Golden Week, which means that it's one of the time when I can make holidays with the family and...

**PN:** So this little girl can run everywhere.

**NP:** I have, I would have to convince my wife that she wants to go once more to BachoTeX. She was there once, right. But I mean for her there is not so much sightseeing there right around BachoTeX. And then there are a lot of freaks.

**PN:** That's right. I mean with bringing a companion to BachoTeX involves leaving them at Warsaw for a little while so that they can have some fun in Warsaw.

**NP:** Yes, yes.

**PN:** And then, when they... Well, I miss seeing you in person.

**NP:** I miss this nice traveling,

**PN:** And hope to see you again soon.

**NP:** Yeah. Well, I don't know when the next conference is. I hope the next TUG will be in person; that would be the biggest chance, right?

**PN:** Well, yeah, ConTeXt is planning a conference for next month, in person in Belgium.

**NP:** Yeah. No way. [Shakes head.]

**PN:** I know.

**NP:** I hope that I can go in December to Europe that would be my dream, but yeah.

The problem is, I mean, with Japan is quite strict with quarantine, when you return from abroad, so that's...

So, for example, you're not allowed, from the airport, you're not allowed to take public transport home, and that means I have to drive a few hundred kilometers across the whole island to the airport, leave my car there, and drive home myself. And then I have to stay at home, for I think ten days or two weeks without leaving. It's already better now because before you had to stay in a government-decided hotel without leaving the room for ten days.

**PN:** Shee!

**NP:** Yes, yes. Now imagine being with a five-year-old daughter locked in a room. That would be quite funny.

**PN:** Well, but the conditions in Japan are extreme. You know the island is small, the population is huge, cities like Tokyo are immense and humongous, and it does need some, you know, pretty tight controls. Otherwise, you know, things will, I mean you see what happens in Brazil and what happened in India, when you don't control.

**NP:** Yeah, yeah. Now let's hope, in December, I hope you can go to Europe and then BachoTeX would be really nice, I mean, yes, I... [Holds up BachoTeX mug.] My morning coffee is most of the time in this cup.

**PN:** All right, hope to see you there.

Thank you, I enjoyed the conversation a lot and we're going to continue this at some time.

**NP:** Yes, sure enough. Also on the social platform, which we are using anytime.

**PN:** Yes, yes, yes.

**NP:** Okay.

**PN:** Thank you, thank you very much.

**NP:** Thanks for your time. Bye.

**PN:** Bye.

# Interview with Frank Mittelbach

Paulo Ney de Souza

This interview took place on 7 August 2021, during the TUG 2021 online conference. Frank Mittelbach has been leading the LaTeX Project since August 1989, i.e., for exactly $2^5$ years at the time of the interview.

**Steve Grathwohl:** Welcome to the interview section of the program. Today we have Paulo Ney de Souza who's going to be interviewing Frank Mittelbach so take it away Paulo.

**Paulo Ney de Souza (PN):** Thank you, Steve. Good morning, Frank, good afternoon or good night, in fact.

**Frank Mittelbach (FMi):** It's about seven o'clock in the evening, so you can say anything.

**PN:** Nice to see you, man. How are you doing these days?

**FMi:** I'm doing well; can't complain. With the restrictions we all go through in the last one and a half year or more, but otherwise I'm doing fine.

**PN:** Are you completely done with HP? Are you completely devolved from that project now?

**FMi:** I decided in 2015... I stopped working as an architect at HP and resigned. And since then I'm only doing freelance work and largely devoting myself to research and LaTeX, yes.

**PN:** Well, start by telling me how does your day look like. How much time do you put in the LaTeX project, how much on other stuff?

**FMi:** Well, at the moment, not counting trying to finally finish the third edition of the *LaTeX Companion*, which is sort of a side project, I spend right now the whole time on LaTeX and the tagging project, which is huge, and, well, after that I have other plans going in the same direction in terms of further research in typography and computer typesetting.

**PN:** Wow!

**PN:** My next question is related exactly to what you what you're talking about, I think, the first time I ever saw you talking, it was a talk in San Francisco at the end of the 80s, where you were complaining about LaTeX. I don't know exactly...

**FMi:** Is that the Stanford meeting, the one where...

**PN:** Yeah, the San Francisco Bay area, most likely Stanford.

**FMi:** At the end of the 80s, it must have been, I guess, the one where Don decided that TeX 3 comes out?

**PN:** Yeah well, I mean, I walked into the talk under the belief that LaTeX was the most wonderful thing in the world and I, and I go there and I see this *lunatic* complaining about LaTeX. I said what the heck is this, and it was, it took me years to process your talk, it took me literally *years* to understand, fully understand what was wrong with LaTeX, and you know now in retrospect, if I could see how correct you were, and I think that the drawback of that talk is that you inherited the LaTeX development, so you guys complaining, that's good.

**FMi:** That's more or less in fact what was happening, and I mean I was young, and so bold, and not seeing the consequences of all this, but, but basically I started working with TeX in the mid-80s or so, and discovered later a draft manual [of LaTeX]. It was before the book came out and I thought this was great ideas, and then I was trying to actually use that on the computers in the university and it died on me, because the computers didn't have enough memory to process LaTeX at that time. So I had to implement my own format based on the ideas that Leslie put forward in his manual. And that gave me a huge sort of insight into all these inner workings, because I had to make it even smaller than it was back then, as far as LaTeX is concerned. So with this starting point I and Rainer Schoepf, we sent an enormous number of bug reports to Leslie when we finally managed to get LaTeX going on the university Multics system. And then I got a chance to go to Stanford.

Well, and then I gave this talk "What's wrong with all this?" I mean — we came from Europe. Basically, I came as sort of telling people why LaTeX is not a good thing. And also from Europe, we came in this year to tell Don Knuth that he really should take care of his cultural background, coming basically, his grandparents and so, from Scandinavia and Germany, to actually take a little bit [of] care about something like diacritics, and so better than a 7-bit system could do. That was a big thing in the Stanford meeting, that we actually managed to convince Don to produce TEX 3, when he basically finally, sort of agreed that there *is* something that needs doing. The other thing that happened was that I personally then met Leslie and we had a long session there during the meeting and afterwards, and then he basically dropped the whole thing on me and Rainer and Chris, and that was the start of it ... so it was '89.

**PN:** So it was more a handover from him and from Leslie directly to you, so the community was not really involved, or was there much more that...

**FMi:** Before that time the whole development and maintenance of LATEX, there was no development happening any more, but the maintenance, if there were bugs or so, you could send them to Leslie and Leslie would potentially do something or not about it. It was all done by him at that time, and he wanted to move on, and so he, I think he was in the end, he was quite glad about having somebody who says, things are wrong, but not just pushing them to him, but actually being prepared to take it on. So we then jointly with Leslie, that was the idea, and this is what we actually did. We jointly devised LATEX 2$_\varepsilon$: a certain set of extensions after, beside the stuff that I said needs fixing, like getting better font support, getting a lot of things done that was not in LATEX 2.09, but we also jointly worked on specifying the abstraction for the color support; Joseph spoke about that earlier. This was implemented by David, but it was a joint effort between David, Leslie, Chris and myself to work on those specs that then became graphics and color packages as the new standard, and this is part of what then ended up in Leslie's book. So his last involvement with LATEX was doing that second edition of the book and going jointly together to release, if you like, the 2$_\varepsilon$-version. But the actual coding for 2$_\varepsilon$, everything which was quite a substantial change from 2.09 to 2$_\varepsilon$, he was not involved in that. He was only sort of a consultant at that time and after this he completely dropped out. He sometimes in the years after sent me a bug report because something in his research paper didn't work; he expected it to be or something like that and yeah, so this is the way it worked. In some sense I think this is a great way for him to do it, because a lot of people can't let that baby "grow up", and just need to stick to it without then actually sort of keeping up with it in some sense.

But he did that, I mean he gave the whole responsibility and everything over to us, and did not interfere with it afterwards, which is great, because that allowed us actually to do a lot of things which otherwise would have been basically not possible and probably LATEX wouldn't be here anymore.

**PN:** Well, so, so let me ask you something else. I now have a better perspective of this whole thing and I kind of see things fall into place. You know color, with a talk by Joseph just a few minutes ago, and in things like accessibility and things like PDF reflow and so forth. So things are falling into place right now very nicely. Do you still have any complaints about LATEX?

**FMi:** Oh, I have a lot of complaints! But yes and no, you're perfectly right in saying that things are currently starting off falling into place, and this is quite interesting because in some sense in the early 90s, so between the TUG conference, e.g., after the time we took over from Leslie and [then], we were thinking of producing LATEX 3.

I mean LATEX 2$_\varepsilon$ was never meant to be, in some sense. The idea was, I mean, I went to Stanford and said this is all wrong, and this is wrong, and mathematics doesn't work, and they should be put into LATEX, and not as a completely separate format,[1] which it was back then, and all these kind of things. So we had these ideas that you could produce a much better LATEX, and we actually produced a much better LATEX during the years 1991–92, and we actually had a new version of LATEX that could compile its own manual.[2] The only problem was, the whole interview we're now doing wouldn't have been enough to get the manual compiled, which was only about 100 pages.

[[connection unstable for a few seconds]]

**PN:** You dropped...

**FMi:** [We had] a lot of [ideas and we implemented them]

**FMi:** but they were too early. There are about two decades [too early] ... (My Internet connection is unstable, can you hear me?)

**PN:** (Yeah, I can. It froze for a second and then it came back.)

[1] Explanatory note by Frank: What is now the package `amsmath` was at that time a separate format called `ams-latex`, because it required NFSS (the New Font Selection Scheme), and that was not part of LATEX either back then.

[2] Explanatory note by Frank: The famous LATEX3 that was never made public back then. It only saw the light in 2020, where most of it was added to the LATEX format as the LATEX3 programming layer.

**FMi:** (Yeah okay, this isn't, this is not as old as the stuff we're talking about, but that's an old computer [here] right now.) Anyway, so we had all these ideas and we had to abandon them. I mean, there were lots of them, I mean basically what is nowadays known as cascading style sheets, we had that for LaTeX before it existed for HTML. But we gave it up, because it was just not workable in the context of this, so there was all these kinds of ideas were around and sometimes implemented.

But now, after, well, what is it 20, 20–30 years now, the problems that we tried to solve back then and only very partially solved was $2_\varepsilon$ — they are still with us, and what is now falling into place is, in my view, that a lot of these ideas are still extremely current. And so, by now, with computers being vastly more powerful, that you can do on your iPhone things that I couldn't have done with the computers I had back then. They are now sort of coming to fruit; I mean they blossomed over the years and, I think last year, I said [the ideas from LaTeX3 finally arrived after 30 years, without people noticing it, in][3] the layer underneath, inside the kernel, and we can actually start making use of it. Last year we built the hook management into LaTeX, which is now nicely being used in various packages already and it's used by us. A lot of the ideas from back then can now be actually implemented and used, and as part of the tagging project, a lot of those harmonizations and improvements will happen, because we will need that as part of making LaTeX accessible and what Jonathan Godfrey said, and rightly complained, is that LaTeX didn't change from 25 years ago to now in terms of what it could do in the sense of making accessible documents. That was just not possible in the past, and now it becomes possible and *is* possible. So, I have still a lot of complaints, but many of them will be addressed in the next few years, and this is now a good thing.

**PN:** It's really nice to see this information, I mean this. I hear complaints about people, you know about waiting for LaTeX 3, but I came to realize that, that I just need to be patient, because the changes that were required were enormous, and you can't stop a train that is moving and change the wheels while it's running. And so I came to appreciate over the years and I guess we'll just have to wait a little bit more and start, and I'm starting to use some of these packages which are coming out of the LaTeX 3 project and they all seem pretty nice and pretty round.

**FMi:** Yeah, I mean this is certainly something you have to take into consideration. LaTeX has a huge user base and, in fact, in the last years I think one can see that it is actually growing because there is quite a need for structured documentation nowadays and LaTeX is pretty much well placed to produce that. And if we are now getting to the point that we can actually produce structured output as well, not just nice looking documents that have great mathematics inside or something, and have all the cross references properly resolved, in contrast to some other systems, but now actually being able to have in some sense reusable documents, accessible documents, and so I think it has a very good place in the world and the future as well. My impression is that the user base is growing rather than lessening in the last years.

**PN:** Alright.

So let me ask you a question which is, I guess it's a matter of opinion as well. I mean programs like InDesign and Word are sort of fast implementing TeX goodies. You know InDesign and Word has had a plugin for math formulas for a number of years now, where you can enter mathematical formulas using LaTeX notation, and previously they would use their own algorithms to decide how the formula look like. And I get in over the last eight years or so, both of them have abandoned their own processing of the formulas and they have left it to TeX, and so we use this literally TeX algorithms to do the positioning, and InDesign right now is implementing microtypography and paragraph breaks very much TeX-like I mean the algorithms are out there, Knuth–Plass and so forth, are public, and so they're bringing it in so you have this instance of where you have microcosms of TeX within an interactive environment outside. Do you see a chance that we'll ever see an interactive LaTeX in the future?

**FMi:** A qualified "no", I would say. I mean we have to be careful of what we're talking about here. Something like the paragraph breaking or how to display a formula in terms of spatial relationships and so, these are algorithms which are very sort of localized.

**PN:** Localized, uh-huh.

**FMi:** I mean that's fine, and in some sense it was always surprising that even though something like the paragraph breaking algorithm was out and public, nobody else, like Word or so, took it up, and Leslie told me the story that... I mean Leslie is employed by Microsoft these days as a researcher. He's not working for Microsoft, he's doing research paid for by Microsoft, if you like. But anyway, he was talking to the Word development and said, well why don't you do this? And see, here's the algorithm.

[3] Again a connection drop :-(

Why is Word producing this horrible output? And that department, the development people, they told him there's no market, we are not putting a complicated algorithm that produces problems for us to sort of maintain and everything into the product if the appreciation from the market is not there.

So the Word market was targeted at something, at least in their thoughts, where an improved typographic quality would not actually be an improvement. Now for something like InDesign that is different, and to my understanding, InDesign was using the TEX algorithm for line breaking already for some time.[4]

**PN:** Yes.

**FMi:** I don't know about the math formulas, but my understanding was they were using it for the paragraph breaking. But coming back. If you have a designer tool like InDesign, where you work page by page, and the focus is on getting, per page, the pages right, you certainly want to have very good quality in terms of line breaking and getting sort of a shot at that automatically in the best possible manner. But the focus was, working with InDesign is more individualization of pages, not focus of a structured document where LATEX has this very big strength, which is describing the content in a reusable, largely reusable fashion, and that is quite independent of the underlying engine.

I mean, if you think about the history, then these ideas that Leslie put into LATEX were actually coming from a system called Scribe, to a large extent. Leslie had a lot of new ideas as on top of it, and he made it popular. The way to describe documents as a structure is actually not, was not done in the system that had TEX underneath. [Scribe] had its own typesetting, which to my understanding was pretty horrible because it was more typewriter-like output if you like, if my understanding is correct, because the underlying engine to actually produce the print was not very good. But these are two different things, and if you think about a huge document like... let's take the book that I'm trying to finish for a couple of years now again: the *LATEX Companion*, which has 1600 pages.

**PN:** Mm hmm.

[4] Editor's note: InDesign was the first program to implement microtypography features (for ordinary text), such as font expansion and character protrusion, in the early 1990s, as part of the "HZ-engine". Peter Karow's memoriam for Hermann Zapf discusses this in some detail: Digital typography with Hermann Zapf, *TUGboat* 36:2, `tug.org/TUGboat/tb36-2/tb113zapf-karow.pdf`.

**FMi:** So that is not going to be interactive because there are dependencies from page one to page 1600, like the correct index at the end, and all these kind of things. So what you *can* have, I think, we will see maybe more is something like "instant TEX", where it appears to be more or less simultaneously updating. I mean, you all remember the talk by David Fuchs a year ago, I think, where he is working on a system where you compile real time, and you have the ability to get the pages you see very naturally, sort of being shown immediately when you make changes. Now that is sacrificing some of the power of what LATEX does, and going beyond in this direction is certainly something that I would not be surprised to see that this is happening. This is still in essence a batch system. I mean, it's just the batch system that you, you don't really realize, because a lot of the batch processing goes in the background, on the stuff that you are currently not looking at, so it appears to you as if it would be instant, but from the core idea I would say batch orientation is part of the story, and therefore it is a different kind of thing.

**PN:** Right right. Nice to have your perspective on this. I try to look at this issue from the point of view of global and local, and interactivity is just like a... This is probably a change that happens very fast, and that you worried only about the local stuff, but the separation between local and global in LATEX seems to be hard.

**FMi:** First of all, it is right now hard in TEX, but the moment you would have system variations of the TEX engine that actually may make this kind of instant result. I mean, Textures was trying to do that years ago which, again, I think it was before it [the world] was ready for something like this, but nowadays that could be a solution not too far in the future, when that appears to be instant, and then some system like LATEX could adapt to that, I would say.

You get more like the feeling you are actually working on an interactive system. Okay, you enter in one screen, but you don't even have to. David showed in his example that you could actually enter in the output format, and it would correctly translate to the source, so he had a two-way, if I remember correctly.

**PN:** I have to watch this talk. I haven't seen it. This is interesting to know.

**FMi:** No, it must be two years ago. It was an on-site meeting, it was the one, the last one that I attended in person, so it was two years ago. Last year was already pandemic. Yeah.

**PN:** I'll have to look for it.

**Arthur Rosendahl:** That was the meeting in Palo Alto. Sorry, right, it was the meeting in Palo Alto. You were there for a while, Paulo.

**PN:** No, I was there for, not for the whole thing, but I had to be out for like one day, and that may be the day that he spoke.

**FMi:** Yeah, check it. It was a question that was unrecorded. It's a pity.

**PN:** Yeah. Now I'll look for whatever he has written up and so forth.

I wanted to ask you about something else. You are also famous for a very big statement that you have used TEX on every single platform you dealt with, going back to VMS and Multics and Sun and Solaris and so forth.

**FMi:** Mainframes.

**PN:** And mainframes and... I would say *even* mainframes...

**FMi:** No, mainframes were the most problematical ones actually, because mainframes are EBCDIC, they are not ASCII, and TEX is very much ASCII, and they don't have a file system; well, not one to speak of.

**PN:** Since the mid-80s, when TEX made its way into PC and Macintosh, we have had a decent version of TEX on every single operating system there is. And the last 15 years I've seen tectonic shifts in operating system usage. Gartner right now says that 80% of the shipment of OS in the world are Android, iOS, and Chrome OS — Chromebooks. You know, none of them run a decent version of TEX. None of them run TEX Live, for example. Do you see implications of this for the future of TEX?

**FMi:** Again I would say *no.*

First of all, I think your statement is not quite correct because on iOS, at least, you get a decent enough TEX to work with, and that is not using TEX Live, but similar, so you can install your own packages on top of what is already provided and it comes with... Well, at least the last time I used it, it was already quite good.

**PN:** Quite good. It *is* the best one of the three.

**FMi:** Yeah, but the point of why I think it is a "no" is... except for tablets. Tablets are a little bit sort of in between. I think this is apples and pears.

I use my phone for an enormous amount of things these days. But a huge Excel field with 20 columns, and try to enter data in it from my phone, or stuff like that, and I don't think that a phone screen is in any way suitable to do programming. To do LATEX documents, that is more than just plain text. And I certainly think that on the tablet level there is a gray area where you can these days, and it will more and more become something like a laptop for you, both in size, as well as in capabilities, but then I would expect that something like, systems like TEXwriter, which I think is the one that I was talking about on iOS. Those and similar ones will show up to enable you to do something like a decent TEX system on such tablets as well. And on the iPhone, they would work on the iPhone but you wouldn't want to work with them on the iPhone. So the fact that you have 80% of all the sort of computer operating systems being sold nowadays being some of these handy[5] devices is one thing, but for programming or for doing graphic design or photograph handling, or what have you, I mean not for shooting the photograph — the iPhone is getting better and better and the other phones as well to do photographs — so Canon has a problem, and all these people. But if you want to postprocess the stuff you still like to have a decent screen size.

**PN:** Right.

**FMi:** ... I would say. I don't think that is going away, because it has more to do with the physics, and so the same I would assume is the fact when you're producing an article and you want to do research in parallel, you want to have two/four windows open to do things in parallel, and that means you're talking, you have a computer. I mean, I like to have a small laptop; most people say, how can you work on the 13-inch laptop?

It is problematic in some cases, which is why I also have a big desktop computer, for stuff where a laptop doesn't work. My argument is it's the balance between nice and light, and I can sit outside in the garden with it. But I wouldn't go down to a phone, so the 80% yes, that is the way the world goes, but that doesn't take away the 20% of the professionals doing certain kind of jobs, and I think stuff where TEX plays a role or LATEX plays a role will be in this 20%, and this is not going to change, Then it will go into the online side. I mean something like Overleaf is taking more and more proportion of what processing of structured complex STEM documents, is going to happen. But again, Overleaf on the iPhone or on your Android is okay. As long as you can just dictate, and your text is not more complicated than that, that may be a future, that you don't have LATEX or something. You just dictate your thing and the

[5] Explanatory note by Frank: handy = "mobile device", a German/English false friend.

artificial intelligence puts your tags in. And I don't know, maybe 10 years from now, who knows?

**PN:** What I meant more, the intermediate formats, more like the tablets, and so forth. For example, you know the Chrome OS is running on very large laptops right now and Microsoft has this line of computers, their Surface laptops are tablets that work as full Windows computers.

**FMi:** Yeah, but those are not the 80%. I think that's the whole stuff, and now I agree with you. I mean this proportion comes up, and this is a certain niche or maybe it is becoming more than a niche, but then it wouldn't surprise me if the TEX community, or one of the other Open Source projects that ports it onto that more as a commercial one. I mean, some of those tools that we have seen on iOS in the last few years were not for free; for ten bucks or something, which is fine. I mean some developers have to live, sometimes, so we shouldn't be too concerned about that. But getting a decent system onto that if the other physical parts fit. So that there is a need. I mean that's the point; you have to have a need, if you have a laptop or tablet which is big enough to actually do this kind of thing. And, yes, I would think if enough of those show up, and we will see that also at some point in time, as an engine, a supported engine. My guess.

**PN:** Well, Jerzy wants to join the conversation. He has a question for you.

**PN:** *LATEX Companion* 3 and LATEX 3, how are they related?

**FMi:** First of all, I already said last year we try to abandon a bit the word "three" because that came back from the days when we were young and were thinking we can do this in two or three years, and then found out the problems are so hard and the computers so slow that we couldn't do it. This kind of LATEX 3 that we had in mind back then is never going to happen.

What is happening, as I announced that last year, is we will go and take the whole LATEX community on a journey with very many safety nets to modernize LATEX rather than building a second system which then nobody's going to use because it takes too long a time to be as a system on its own usable. So there *will* be something like a modernization going on right now, and it will not take another 30 years. We are now in a position where we can actually make these kind of things.

And the three in both cases is just because I only have time about every 10 to 15 years to write a book. And it's good that these books actually stay current for that long period of time — we can't really say that the *LATEX Companion* 2 is still current, but it's still not completely useless, so...

**Jerzy Ludwichowski (JL):** Yes, it's still handy in many cases, but yes, one has to resort to watch online for packages because they update, and so on.

**FMi:** Potentially.

**JL:** Yeah.

**FMi:** But this is going to change. I don't want to make promises, see. The book is in its last stages, and last stages, that still means it goes to professional copy editing and it goes to professional indexing, and I have to do the layout after the copy editing because everybody knows that whenever you touch a system like LATEX, TEX, changing a couple of words, everything changes in the line breaks. So there's still a lot of work to do in parallel to all these projects that I *actually* want to do. But the majority of the work is done, which was hard enough. I went through 5000 packages on CTAN, documentation by documentation, made notes about what is good, what is not, tried things out, checked what has changed in various things. By the time I got around with one chapter, then five chapters later this was no longer current, so I had to do it again, because it was just a huge, huge undertaking, but I think it gets around... well, how should I say, it takes too much time sometimes, but...

**JL:** It's a moving target, isn't it?

**FMi:** Yeah, yeah, sometimes it's a moving target, which is why it is good to get a book out, because then...

**JL:** A reference point, yeah.

**FMi:** It's a reference point, and people accept that. When we talked about it and decided, Oh, you know a lot of those things when the *LATEX Companion* 2 book came out where we're improved because I tried to explain them. When I couldn't explain them I talked to the author of the package and things sort of got sorted. But then the book was out, and so the documentation said the package can do this, and this, in this form, and these kind of things then stayed, because they were a record through the book. Because of doing the same, though, with the *LATEX Companion* 3, but not putting the book out... I mean I started it in 2018, I think, so it's quite a long time ago. I cannot really tell people not to change their minds when I'm not being able to do my promise after having sort of talked them into doing something to improve that further.

So yes, it is a changing target, and which is also why it's good to finally come to a close, because then it is no longer a changing target, it's more like a point of reference.

**JL:** I was, I asked this question because perhaps it's good to make an official statement that people don't confuse those trees.

**FMi:** Okay, well…

**JL:** I am waiting for, for the 1600 pages or so, to get my hands on, but I'm confident. I understand that LaTeX 3, in the sense like the language and the system, is evolving, and it cannot be frozen into the book or the book frozen into it, but that doesn't…

**FMi:** Well, this is, this is definitely independent. The work that we're currently doing, I already documented the new hook system because that is ready. It is part of the book.

Everything that has gone around in the tagging world around good "getting accessible documents", that will have some major improvements and functionality coming along in the next years. This is not going to be in the book. I'm not going to wait for that. That's not important. The focus of the book…

**JL:** "Accessibility Companion". Yeah, that's a, that's a thing to be done.

**FMi:** That could be a small, small booklet. That is fine, but, but the point of the *LaTeX Companion* is to be a companion to your normal work, and this is what all these packages or, let's say, selection of the packages, those that I thought is worth having at your fingertips. For example, one of those from every sort of area. Like the last *Companion*, the aim is to be in itself sufficient most of the time to do whatever you want to do in terms of LaTeX. This is not so much about the inner workings and the improvements of design, it's more about all these packages that are out there.

**JL:** Thank you for the explanations.

**PN:** Thank you for this wonderful interview, Frank.

Does anybody else want to join the conversation or has any other questions for Frank?

Well, with that I'd like to…

**JL:** Invite him to BachoTeX.

**PN:** Ahh!

**FMi:** I hope it's going to happen next year again.

**JL:** Yeah, yeah. Chances are probably not zero. Hopefully the pandemic will let us get there. We might restrict the audience like being vaccinated or some such. Be on the safe side.

**FMi:** That's the… Okay, this is, this is what in my case, for myself, it's sort of the biggest sort of restriction right now. I was fortunate enough that I, because I decided to resign from, say, industry work, and so I was not affected by the pandemic in that sense. I did work from home already at HP for 10 or 15 years because I was doing international work so that would not have changed anyway, but what I really missed in the last years was the sort of personal contact, by email and by phone or Zoom or something, and things like those conferences. I mean I haven't been on any since 1919,[6] and then this.

**JL:** As Paweł Jackowski puts it, the ideas are being created between heads. So if you have those heads together in one place, and people don't go away, then there are chances for ideas. New ideas or solutions and things like that. And it's nice to hug people.

**FMi:** Yes, and to have a joint beer. And all these kinds of things that I'm currently missing. So I'm sitting here with a glass of water, that's all.

**JL:** And a glass of beer underneath the table.

**FMi:** I am not telling.

Okay. Thanks, Paulo, nice talking to you would love to see you in person, but if you are speaking, you have to put your unmute button.

**PN:** Yes, no, nice to see you in person, nice to be such a good sport for an interview like this one. And I sincerely hope to see you all and BachoTeX is probably the best environment for all of this.

**FMi:** Oh yeah, that is absolutely lovely.

**PN:** And now just the time that you have to be able to talk to people outside the talk settings. It's absolutely wonderful and I hope I'll have the courage to make that trip the next time that happens.

**JL:** Thank you.

**PN:** Thank you, thank you.

**Steve Grathwohl:** Thank you, Paulo and Frank. That was wonderful. I too hope to see all of you in person next year somewhere.

**FMi:** Yeah. We haven't seen each other for a long time.

**Steve Grathwohl:** Yes, yes. It's been challenging for all of us.

[6] Explanatory note by Frank: of course I meant 2019 :-)

## Interview with John Hammersley

John Hammersley, Paulo Ney de Souza

This interview took place on 8 August 2021, during the TUG 2021 online conference.

**Paulo Ney de Souza (PN):** Hello everyone, good morning or good afternoon, John.

I'd like to introduce you all to John Hammersley. He's a founder and CEO of Overleaf. I'd like also to invite all of you for a conversation. Overleaf is new to most of us that have used TeX in other forms, in other ways, and so you're welcome to join and make this interview a joint conversation with everybody.

The first thing I'd like to say is that Overleaf is a financial supporter of the TUG conference. They don't have any editorial relation to us and John has not seen any of the questions that I'm going to ask him here today. Is that correct, John?

**John Hammersley (JH):** That's correct, yes. I'm flying blind!

**PN:** So, welcome to our conference and hope you can join us here in the future.

I'd like to ask you first, how did you get interested in TeX? Was it through mathematics, was it through something else?

**JH:** Yeah, yeah, so I did a mathematics and physics undergrad degree, and that was where I first came across TeX. I think we used it for one of our group projects in like the third or fourth year. And that was kind of how I first came across LaTeX. At the time it was just a nice, ... I think someone else had used it, and so we just used it. It was just a nice way to write up the reports that we were doing, and I didn't really think much more of it in a way. But then I did go on to do a PhD in mathematics at Durham in the UK, and it's you know, what everyone uses in mathematics, and so I used it for my papers and thesis and it was very much just [something I] picked up. I installed MiKTeX, got going; my supervisor I think probably shared some files, and so I had some templates to go on and stuff, and I used LaTeX for writing my papers. I liked it, and I did my CV in it as well then, after deciding to leave academia.

I had used it for things slightly outside of papers, and I was lucky enough to go into a job then, working for a company that was working on driverless taxis, where we were in a research group. The company itself had spun out of the University of Bristol's engineering mathematics department and so again, we were still using LaTeX internally and... I'm running ahead in a way, that's kind of where Overleaf came out of, that research group.

So essentially I got into it at university :)

**PN:** So you *were* involved with other technology projects before Overleaf?

**JH:** Yes, I guess in terms of my career as it were. So I left after my PhD and decided that I kind of wanted to move into industry because I just, I didn't really feel like staying in academia. I felt there were lots of people doing really great stuff in academia and so many papers coming out. I guess I didn't feel like I could maybe make much of a difference there. And so my now-wife was moving to Bath in the UK to do a teacher training course, so I moved with her and looked for jobs in the area. And there was this company doing driverless taxis, and they were building the world's first system at Heathrow Airport, and...

I see someone spotted the LEGO sets in the background; I guess I'll get on to that in a second.

But yes, so I was working at this company doing driverless taxis, and it was in a way before driverless cars became cool! It was a few years ago, and the group there, we were doing a lot of research into empty vehicle management. There was a lot of queueing theory involved in vehicle redistribution and how you in a network make sure that you don't have a surplus of empty vehicles where you don't need them, and how can reroute them as needed without making too many trips, because a lot of the ideas behind driverless vehicles was to minimize wasted journeys, and try and optimize the network and things. So we were doing a lot of research and publishing it, and actually, we were starting to work... so, when I'd written my papers at uni, I was just working either with myself or with other mathematicians, and so everyone knew LaTeX at a reasonable level, whereas we were now writing papers with people from other fields and other disciplines that were more used to Word and also writing for conferences or for journals where Word was the requirement to submit. You

doi.org/10.47397/tb/42-2/tb131hammersley-desouza

know there perhaps wasn't an opportunity to submit in LaTeX and, and so we were then both, like I say, trying to work with people who hadn't used LaTeX before, and I think as we all know, if you email someone a LaTeX file and they've not used it before, there's quite a barrier then, for them to be able to collaborate, and so that was really, I think, where the original motivation for WriteLaTeX, as we were called back then, came from. And it was [built by] John Lees-Miller; he is far more a programmer than I am. We're both mathematicians by background, but John is a computer scientist as well, and a systems engineer, and he — over one weekend — developed the prototype for WriteLaTeX.

[It came about] basically because Etherpad had recently come out and we were using it, and Etherpad was this great way to collaboratively write notes. And we'd actually used it for writing LaTeX documents, but the problem was you couldn't compile it, and so you had to then take it to... you know, move it locally to do a compile, and so really the first version of WriteLaTeX was taking that idea of just a collaborative notebook and adding a button to compile a PDF. And then things obviously grew on and out from there.

I guess I might just take a tangent to pick up on Sam's [samcarter] comment in the chat about LEGO sets, just because, you know, I do quite like having this background [leans to show shelves], and clearly... [waves hand at camera] the resolution is good enough that you can see lots of classic space LEGO. You can see classic space LEGO there as well, and some M-Tron and Blacktron and bits and bobs around. You can't see over there, where there's Star Wars LEGO and many other things. I blame lockdown for this!

**PN:** Wow!

**JH:** And then going over to my parents' house and digging up my old LEGO sets, and then coupled with eBay, which is a source of cheap old LEGO :) ... errm, this room is a tip... I could try and turn the camera a bit without breaking things. Let me see if we can show it over there. [Moves camera.] So you've got the Death Star up at the top, the International Space Station and the Saturn V, that's about it, loads of LEGO up there...

**PN:** You're a hardcore technologist.

**JH:** Something like that. I mean, I guess. You probably saw the Saturn V. I think it was actually the space race that got me into science and innovation and stuff because it happens that my sister was born on the day that Neil Armstrong and Buzz Aldrin set foot on the moon. And so, every year, obviously with my sister's birthday, we also have the anecdotes in the family that it was the same day, you know back in 1969 when man first set foot on the moon. So growing up, I was very much into space and space exploration. And we got to go, luckily, with my parents we took a trip over to Cape Canaveral and saw the Kennedy Space Center, and saw the, you know, they've got a Saturn V rocket on its side, I think, or part of the section. Yeah, we should definitely have a LEGO TikZLEGOs package... I think that's an excellent idea.

**PN:** [laughs] Don't give ideas to samcarter or she'll get to work!

**JH:** Yeah, I grew up being interested in technology in general and, and I think the idea, like the pioneering nature of what they've done on the space race. I think that's one reason why working on driverless cars was quite attractive after uni, because it really felt like it was something that was, yeah, like something that could change the world.

And then actually, with the WriteLaTeX prototype, one day John Lees-Miller came round and we were chatting and he was saying, you know, people are using WriteLaTeX, it's starting to cost a bit of money with the servers, and so maybe we, maybe it would be a good idea if we looked at it more seriously and decide whether we could turn this into something. Neither of us had kids at that point, and so we could actually consider doing a startup and consider doing this crazy thing of quitting our jobs, you know, on driverless taxis and working on LaTeX stuff full time, at least for a bit. And yeah... I've always been interested in new things, not in the "oh, it's shiny and new" sense, but [in the pioneering sense]. I've got a book on my desk, the recent one *Liftoff*, about SpaceX, and what they've been doing, which is amazing, and I think it's amazing, all of this inspirational stuff that's happening right now.

**PN:** So that was about, what was that, about ten years ago, John, when you guys got out of driverless taxis?

**JH:** Yeah, it was about eight or nine or ten years ago, yeah, I think the very first prototypes that John put together were in late 2011, and then 2012, late 2012, is when me and him decided to quit our jobs at the time, and go for it full time. Yeah, I know. Crazy! I feel like it makes me start to feel old now. Ten years ago.

**PN:** So, your other job now is kids; you guys have kids now. How do you, how do you balance those two things you know, pre-pandemic, post-pandemic, if there will ever be a time, how do you find time?

**JH:** Yeah, I mean, I mean I love… So I've got two daughters, Julia, who's five and three-quarters, and Annabelle, who's three and three-quarters, and obviously, the three-quarters is quite important when you're in single digits, and… I've found it amazing! First of all, I think we never would have started Write-LaTeX if we had young children around. I think… it would have seemed like a crazy idea! So I think it was good that we started WriteLaTeX, which is now Overleaf, [when we did]. And in many ways that has been described as like the first child, in a sense, because you are looking after this startup in many ways in a similar way to a kid, and it's very dependent on you in its early days: it needs looking after, it needs a lot of attention. So now having had kids, I can see that the amount of hours that John and I put into WriteLaTeX in the early days — it's not in the same way like kids — but you certainly have to have that time free to dedicate to it. But no, overall I think it's been brilliant.

I mean, hey, it's given me a legitimate reason to have LEGO all around the house, um, because you know, kids like LEGO! That's, that's why we have LEGO there! And [with the] pandemic, I think that we're lucky in that our kids are young enough, and John's is young as well, in that they didn't really notice so much. They have a bit, but they're too young to… Secondary school kids I think have had it much harder in sense of not being at school and things, whereas, yeah, we were okay, and with Liz being… Liz, my wife, is a teacher, was previously a teacher, and so when it came to homeschooling, we did all right there as well, because Liz was able to pick that up.

**PN:** Cool, cool, cool. So I guess you know when I, when I… If you could switch a little bit to Overleaf and some technical things. When people look at it, they see a lot of different things. I mean the ability to work with it, without having to do installations, the ability to work everywhere, the ability to share a file with somebody or even a large group all over the world is just superb. Somebody's seen that, we've been looking for this everywhere over the last, you know, 30 years. You know, I guess, since webdev started in the mid '90s, people start sharing LaTeX files and doing development together and so forth. This is just great that this now has formed into a solid solution for all of us.

But then there's a couple of things that people don't see right away, and I was recently shown, told by people, how much they like that. One of them is the automated processing. The automated processing is just like fantastic, you don't have to worry about what kind of indexes you are using, how you're post-processing them, or if you're using BibTeX, or if you're using `biblatex`, or if this paper is in this framework or this other paper in this framework, and everything. It's taking care for you. I think that can make, I mean, that has made a few of my co-authors lazy. [Laughs.] They no longer take care of their own indexes, and so forth. They just wait for the thing to be ready for them. The other one is one that I have met about four years ago, which is a continuous submission process into a journal, in which ways, you can not just submit, but you can let the author work with you as it's going through the processing within the journal production, as it's going through the book production. And they just love that and my question is, how much up the sleeve you have? What are we going to see?

It's this many features. I mean mathematicians are waking up now to the fact that, that they can do submissions via Overleaf and they're just discovering that. You know 99% of the mathematicians you talk about there, you talk with them and they don't know about that feature, they don't know about the journals which are in and the journals which are not in yet. But tell us, what's coming our way.

**JH:** Yeah. So that's great, and it's a great lead-in to things, and I think… taking it back to the main benefits, certainly the one that it was built for was collaboration, and actually lowering the barrier by meaning you didn't have to install anything was a slight byproduct of that. The main reason for us to use it was to collaborate, but then, of course, as soon as it's there as an online compiler… if you are a student who's just been told, "we need to use LaTeX", and you search for trying to install LaTeX, [but find] you can just try it out in the browser without installing; for a lot of students who are very new to it, I think that really let them try it out for the first time, and I think that's what having an online, accessible through the browser, TeX compiler has done: it's just meant that people who have never used this before can immediately see the results.

I think we've done a lot to try and lower the barriers for new users while still keeping… I'll say the full power of LaTeX — it's not, clearly there are things you can't do on Overleaf — but it's still a LaTeX editor, so you can edit the full source and everything. It's very much focused on providing a really great LaTeX editor as far as we can, within the boundaries of what we can do with online compiling.

There's hundreds and hundreds of things on the wish list. Overleaf's usage has grown, and quite a lot of time is spent on just maintaining that level of service and trying to keep improving it. We did the big integration with ShareLaTeX a few years ago now. I think that was one where we tried to definitely keep it as smooth as possible for the users who are using both platforms, the original Overleaf and the new [to us] ShareLaTeX, and tried to make that merge happen with the minimum number of road bumps for the users, which has meant that behind the scenes we've had to do quite a bit of cleaning up work then to tidy up the systems, and to make it all, bring it all back together again into something that we can take on.

I think we've got a few things that we were trying to do, I guess, picking up on the question of publishers. A few years ago we have put a lot of effort in with various publishers, and I don't think we actually got as far as we hoped. I think we had hoped that by this point, we would have a... you know, you submit to a journal, and then editors and reviewers can then leave comments, or propose changes, or however, depending on the journal and the particular sort of process, and then take it all the way through. And then the journal could then take the LaTeX on, or could run the LaTeX through an XML compiler and get the XML out for the website, and we ran into some challenges there; it's just that it is a very complicated process and all the different journals use a lot of different systems. And also, you know, most journals accept LaTeX, and Word, and so are looking ideally for a solution which works for both, and I think for us as being very focused on the LaTeX world it was then difficult to take it all that way. So where we got to with a lot of them is trying to help make sure the templates that they had, their LaTeX templates, were up to date and trying to provide a good way for authors to get going and then just trying to make sure authors had the files they needed to take into the submission systems.

I think we've done a few notable exceptions where it did work really nicely; the best integration we had for a long time was with F1000Research, where their editorial team did leave comments and track changes for the authors in Overleaf, who then could come in and accept those or make further changes and then could go through that editorial process as needed. And even that integration was a very early one, it was one of the first ones we did I think, and even with that there are many things we could have improved on it. So, I think with the publishing world, where we've struggled a bit is trying not to add an extra complication, not somehow make the workflow more complicated in a route to making it less complicated! And so, now, we generally focus on trying to make it as easy as possible on the authoring side and then on the submitting side. There was a talk from Heinrich [Stamerjohanns] earlier on, who talked about the LaTeX to XML conversion that we looked into as well, and again, ideally it was something that we were hoping to incorporate into Overleaf so you could hit a button and get an XML output, as well as getting the PDF output. I think we made a lot of progress, and Heinrich has made a lot of progress there with that, and obviously [others have] with LaTeXML as well, and all of the other work in the ecosystem that's improved out there... [but] again it's still something that we haven't quite worked out how it best fits in, in a way.

One of the challenges we have sometimes is that Overleaf is being used by a lot of people for a lot of different things, and in the publishing world getting the XML output out is useful, and it's useful in certain workflows. For others of our users, let's say students writing group projects, they have no need for the XML and it's something that's almost unnecessary. So we have to try — with the development team and the product team that we have — to work out where can we try and add the most value to the most users, or which bugs do we need to prioritize fixing, or how do we keep Overleaf up to date with the new TeX Live releases that come out, and the new developments there.

I see there's a question in the chat about "why is there is still no Overleaf app for mobile devices?", and this is a good question because we've already looked at this.

**PN:** I'd like to precede that question which, with a preamble... it's a question that I have asked the other two interviewees in this conference.

Right now, Overleaf is the only decent TeX that we can run on just about 80% of machines out there in the world. And a lot of people say, I do not need to run LaTeX on my car player, or my car dash, or whatever, but there are a lot of very powerful tablets which are straightly replacing desktops right now, and Chrome OS is the dominant operating system distributed in the world right now, and the only way that we have to achieve a decent and complete distribution of LaTeX is with Overleaf.

I even go as far as saying that it's important for me to have it on my phone. Sometimes I am at, not during the pandemic, but with people and that the only thing I have in my hands is my phone and I wanted to be able to share a result, to be able to share a display of a PDF that has been just produced by Overleaf. And I think that's very important, but the problem is, the controls of a web browser within a phone are very difficult, so when are these apps coming? Are these apps ever coming?

**JH:** That's a great question. I think we first looked into doing an app probably in 2013 [laughs], which is a long time ago now, it was in the first years that we were really getting going. I think for a long time, for a good few years, because tablets were used, and people would use the browser... really it became a question of, for an app, there's two use cases. There's either like you say, "I want to be able to get maybe the PDF or get projects", "see my list of projects", but not necessarily edit, but be able to have access to certain things. Or there is the wanting to be able to edit whilst offline, because if you're on a tablet and you're online, then using a decent web browser is generally, has generally, been okay. And so what the Overleaf app to us needed to coincide with was being able to use Overleaf offline in some sense: an offline mode. We looked at this ages ago, and I think we were... it was something that we were considering in 2016, 2017 and then that was when we merged with ShareLaTeX and we kind of moved into the integration project that was bringing the two platforms together. And, like I say, we're still unpicking some of the stuff there and we're very close to having completed a lot of that work, and then that does unlock making an offline mode of Overleaf available. That is still very much in the realm of to be determined exactly what that means, but the ability to see projects, [edit projects], and at that point an app starts to make sense, because at that point you have things you could usefully use an app for... so I guess that was a little bit of a waffley answer, in that it's certainly not going to be in 2021. I think looking at an offline mode for Overleaf is something that we might want to look at next year, and whether that involves an app as well, that's the kind of related question, but it becomes much more realistic next year.

And I guess, whilst I'm looking at the questions, so Vít's asked about Git and continuous integration and the fact that that makes it easy to collaborate, and does this diminish the value of Overleaf as a service?

It is interesting because I remember in the early days when WriteLaTeX first came around, there were quite a few people that said well, "why do you need Overleaf or WriteLaTeX? You've got Subversion or you've got Git, you can use a local editor and you can collaborate with others and it works", and I think that's true for people who are familiar with that, it does, it *is* very possible, and certainly GitHub has made it easier to collaborate on that kind of repository style way. Overleaf just offers it in a different way, I think, and for people who are just getting into this and, not necessarily computer scientists getting into this, just anyone getting into writing a technical document online, Overleaf and just that "going to a browser and being able to use LaTeX" I think helps more, or is an easier path in, than perhaps if you've never used Git or GitHub before. If you've used GitHub or Git before then I agree, then that workflow works for you, but if you've never used Git or GitHub before I think Overleaf probably offers a lower barrier in.

I should say actually that I went to a talk given by Vince Knight — who [incidentally] recorded loads of Overleaf videos ages ago, before we ever got around to making some intro videos he did a load of nice snippets — he's at Cardiff University and he's been heavily involved in the open science work there. He gave a nice talk about Overleaf, and the way he put it was that there's this nice sort of triangle of ways of using Overleaf so you can either edit online in Overleaf in the LaTeX source mode, [or] if you've never used LaTeX before and you don't necessarily really want to use it, you have the Rich Text mode, which is still in beta in Overleaf; it's been in beta for a long time, but you can hide some of the code away and just edit the text if you're not looking to do that. But equally, if you have a preferred editor offline, like if you're already set up with all of that, then you can collaborate using Git, using the Git bridge, or GitHub to then push and pull things and sync with people who are using the online interfaces. And I know, none of these are perfect... there's things we'd like to improve in the LaTeX source editor, there's definitely things we'd like to improve in Rich Text, and there's stuff we'd like to improve with the Git bridge, but it does mean that it makes collaboration easy between people with different levels of experience and different levels of comfort with technology and with collaboration.

**PN:** Wow. Any other questions for John? Do you want to join the conversation? You can unmute yourself and join us.

For me, we just recently had an experience, John, with the editorial services at the IMPA, a math institute down in Rio de Janeiro, and we did the International Congress of Mathematicians about three years ago. They, we have been working on this continuous submission process for a long time because mathematicians are not acquainted with having their proceedings ready at the first day of the conference like the physicists and the astronomers and so forth.

And so we tried to coach them in being able to work before the proceedings and have it ready for the first day, so you can walk into the meeting and have it in your hands. That requires a lot of work and fast work and during the ICM two years ago we had about, about, I would say about 80% of the people sharing their files on DropBox and participating in development, and only about 10% of them on Overleaf.

This year for the Brazilian Mathematical Congress, we had 51 authors, out of which 45 chose to do their sharing on Overleaf. It was a tremendous change in, within this two year timeframe. And people are enjoying a lot to learn about quality TEX from the editorial and production staff. You know, when the editorial production staff tells them "we're not using `eqnarray` any more", and authors are very glad to ask why and see their files changed in that way and see how it's better. So I think it's a very positive change, and this has been really great.

**JH:** Now that is fantastic. I see Boris has his hand up; I just want to use Paulo's comment now to highlight something else which I think really helped Overleaf over the years, and it's the fact that we have an in-house support team. So it started off with, obviously, if you have a software as a service, you have users writing in with issues or bugs, or just problems using it, which is fantastic, actually, it meant we always have a lot of feedback from people, but John and I were answering those initially. And very quickly it becomes a big part of what you're doing, and Lian Tze, who I'm sure many, many of you here know Lian Tze from a LATEX community, [well] we put out an ad for a sort of TEXpert to join the team and Lian Tze was just amazing and still is, she's still working with us now and has been fantastic. We've grown that support team over the years, we now have a really broad range of skills and backgrounds, you know, from engineers, teachers, to people who've done lots of different things. But because we have the support team in-house, it means that if you write into Overleaf with a problem, you get hold of someone at Overleaf. And, most of us, in fact all the support team, are really great with LATEX as well, you know, coming at it from different backgrounds, but often can help the users and solve their problem.

I think this was an indirect benefit of having LATEX compiling in the cloud, is that it's not just the authors that can see it, but if someone needs support on it, they can get it. And so a lot of authors who write to us and ask us for help, and are happy for us to look at their projects if we can help, we *can* fix something really quickly and that gives them a really positive experience not only of Overleaf, but of LATEX as well, and it lets them keep going. Overleaf has grown a lot, and I think it's helped people get past some of the initial issues that you have with LATEX, where you get stuck on something, and you maybe have to spend hours and hours fixing it, with the fact that you could either ask someone from Overleaf to try and fix it, or I'm sure what happens out there, with classes being taught and everything, is people asking their teacher for help, or asking fellow students for help and being able to fix and spot each other's issues.

And that was very much, if you'd like, a side effect of the collaboration, it lets someone else help you get unstuck in a way that, if you're working locally on your machine, and it's just you, and you aren't in a classroom with other people, or you don't have someone else to come and look over your shoulder, you know you've got someone there that can fix it, and you *know* that it's fixed because it's compiling on the same machine in the cloud, rather than I could send you my LATEX file and you could try and fix it, but then, if you've got a different LATEX installation running and stuff it might not look quite the same...

**PN:** Right, right, right.

**JH:** So, Boris you've had your hand up for a while and I should...

**Boris Veytsman (BV):** Thank you, thank you. I said this exactly a year ago at our last online conference, but I would like to repeat it now, that as a old timer, an old-time TEX user, I was completely sure I would never need an Overleaf, so I never looked at it. And, in the last couple of years, I found that it's so much enhanced my collaborations that I now can work with people who would not go through the task of installing clutter on their computers and they became... or installing version control, something like Git on their computers and now they are very productive collaborators. And the nice thing [is] that with Git I can just work on my computer and just use Overleaf as a big Git repository and they can use it for editor.

I am now convinced that what you did was one of the several most important changes in the TEX world for the last years. You probably revolutionized the ways that a lot of intelligent people are using TEX and I just wanted to say is that what you have done and what you're doing is absolutely amazing.

**JH:** Well, it's hard to quite know what to say to that other than thank you... and it's certainly not just me, I mean so John Lees-Miller, you know, from a technical perspective, is very much the first name, but also James, James Allen, and Henry Oswald, who launched ShareLaTeX, and actually, though, everyone along the way that has contributed. One of the brilliant things, for me personally, is that we have such a great team at Overleaf now. And we have had people who've come and gone over the years. But it's been a really nice environment, and I think in a way, like the whole TeX community, is very friendly. There's a lot of support given for new users, and there's a lot of time put into the packages which help people, in different disciplines. And it has always been open and collaborative, and it's nice to see. I think Overleaf and this sort of cloud-based LaTeX, helping to continue it on, and I think helping LaTeX be more accessible to people, in the distribution sense of accessible.

I also think we were also just lucky with the timing I think in many ways, and they say this a lot, DropBox had recently come out and there was a lot of other things that people were starting to do through the browser rather than installing directly. Like I can't remember off the top my head, but things like Google Docs then came out after Etherpad, and people just became a lot more comfortable, or a lot more used to using the browser as the entry point for programs, rather than the directory of locally installed stuff, and so I think we were around at the right time. John [Lees-Miller], he's led the engineering, and he and Tim [Alby] did most of it in the early days themselves, and we have gone on since then in how we continue to build on that.

But thank you, Boris. I do appreciate that.

**BV:** Thank you.

**PN:** If you want to join the conversation, you know, have your questions come in, you can just come in online any of the panelists, or, if you want to, raise your hand.

I think lowering the bar has been absolutely, from all these features that we've seen out there and being available everywhere, it's been such a game changer.

You mentioned one thing which was the submission process, and the submission process has two faces, you know, as the face of the author has to fit to certain standards and also has the side of the production staff that receives, on the other hand, and has to make sure that what he sees is what the guy intended to write. And Overleaf bridges that divide, the divide of, you know, yes, what I have, it's here, and then the next minute, the production staff can continue to work with it and be sure that he is working on what the guy intended to, rather than you know some display of a specifically Russian font on a English manuscript that's not happening properly on page 147. That's really great.

**JH:** Yeah, and I guess the best one I think you know. Oh sorry.

**PN:** No, go ahead, go ahead, go ahead.

**JH:** I was just gonna say, a couple other publishers that we have worked with a lot of years are the IEEE and the Optical Society. I think, in particular, now the Optical Society uses Overleaf for compiling the submissions that come through their sites, because I think one of the benefits they found was, I think we've all been there, when submitting a paper to a particular journal, is you might upload the LaTeX files and then it tells you that it can't compile them. And it maybe gives you the error logs, but then you've got to figure it out, and probably you're submitting it because it does compile on your machine. And so you've now got to work out why, why is it saying it doesn't compile on whatever, you know, system the particular journal is using. And so what's really nice in how we've managed to get it to work with the Optical Society is if you go to there, we... well, first off, we worked with them on templates, and then we worked with them on the submission from Overleaf, so if you submitted it from Overleaf it could go straight in.

But now the way it works is if you go to their submission portal directly, which most authors do because I think it's natural, you go to the OSA and try and submit, and when you upload your LaTeX files, it sends them to us to compile and if they do compile, then that's great, PDF comes back, everything's fine you know, it's the same as it would do anyway, if it's all gone through well. But if it doesn't work, the nice thing is that you get the option to open the project in Overleaf to fix the errors and it means that you and the editorial team could both see that to fix that or... And it means that you know that if you do fix the errors here, they're definitely fixed rather than, you know, trying to fix them on your machine when, from your perspective, it already seemed okay and everything. Like you say, I think having that common view of the document is actually very valuable, not just for reducing the frustration between co-authors who have maybe got different systems installed and see different layouts and things, but also for the publishers, who know that if they know they're seeing what the author intended to submit, then they maybe avoid some of the miscommunication further online, or even having to try and get the author to fix problems which aren't due to the author's system.

One of the things I remember from, it would have been a few years ago now: on the publishing side, because we used to get questions come in from authors, or sometimes from the editorial teams, I think in a few cases we were actually able to help fix underlying issues with, let's say, the bibliography layout. I remember one where DOIs weren't visible in the references, and it was because the reference style had been designed before DOIs were used. But what the journal had been doing, because they didn't really understand exactly, necessarily understand, the details of it, they were just asking the authors to display the DOIs, and so I think each author had to keep trying to hack in a way to get the DOIs shown. And because some of those questions came into us, we were able to suggest updating their bibliography style so that it did that automatically. Then the publisher is happy because they no longer have these frustrated authors who they're trying to get to do a thing which was really a problem with the journal template itself.

I see there is now a couple of questions in the chat and Jonathan has his hands up. So Boris's question about the best address for feature requests, I would say, always sending stuff into `support@overleaf.com` is the best way to get it triaged and passed to the right person.

A question from YouTube: Are we thinking of putting Python TEX in Overleaf once again in the short term?

That is a good question. I know, I think what they're referring to is that used to work in [Overleaf] v.1, I think, and now it doesn't work. I need to double check, because it has been a long time. I don't really use Python myself, but I will take that one offline and we will follow up so if the person that responded... we'll try. Yeah, it was a question from YouTube, so I guess we might follow up there...

Jonathan, you've had your hand up for a while.

**PN:** John, one of the things that we have a hard time within the TEX community is to have an idea of how many TEX users are out there and how distributed they are around the world, and who is really using it, and so forth. I guess you have a much better bird's eye view of that action. I don't know how much of that you can share with us, but is there at least a brush that you can give us of how TEX is used and uh, and how broad...

**JH:** It really is used all over the world. A lot of our users are students at universities and, if you look at probably where there are technical universities or universities with big STEM undergraduate populations, you'll probably get a good... that will probably correlate a lot with the usage of Overleaf. We definitely see Overleaf, and just cloud-based LATEX, has really helped students pick up on Overleaf, and I think for the student it's particularly nice because it's a way for them to produce a report which maybe stands out, like especially if LATEX isn't necessarily always used in their group projects or something. It's a way for them to show that they've learned something — LATEX — and that they can produce this [type of document], and it's kind of a way to get ahead of things. So, yeah, I think it really is all over the world, we have a lot of users in the US, a lot of students in the UK as well, users in South America as well as users in India, and as well as, really in places all across the world... we have a lot of users in Japan and there's a Japanese community around TEX and who use Overleaf a lot, and...

I do remember, that we always seem to get — once a year — usage from the International Space Station, but then that's just the good old April the first on Google Analytics, for anyone who has ever checked their Google Analytics on April the first.

[In summary] Yeah, I think it's if you follow the student population.

**PN:** Uh huh, uh huh. Thank you very, very much.

**JH:** Jonathan still has his hand up.

**Jonathan Fine (JF):** Hi, John, and thank you for coming to be interviewed, and thank you for establishing Overleaf which really has made a big contribution to the TEX community, but I sort of have mixed feelings and in part it's because I'm an old timer.

And it comes down, mixed feelings have come down to this... I'm trying to find a way of saying it that causes the least offense. The TEX Users Group has got, had responsibilities, and I think still has responsibilities, and I feel as though part of our release remit has become things that the TEX Users Group or the TEX developers, the TEX community generally should be doing for themselves.

So the previous discussion of the distribution of TEX users (where are the TEX users?), that's something that sort of the TEX Users Group should know independently of the information you're kindly sharing with us. And another example is the strength of Overleaf, as you quite rightly pointed out, is that it provides a reproducible environment for the typesetting of TEX documents. But you know it's that reproducible environment, I think, is something that the TEX developers should be solving themselves and not relying on, can I say, the heavy resources of Overleaf, because that reproducible environment is hard work to maintain I am told.

And where I see this, for example, is that a PDF document now has the embedded fonts it needs, so you can send the PDF document to somebody and they can read it. And similarly a Word document, together with the Word program, has all the resources it needs, whereas I can't send a TeX document to a third party in such a way that they can readily get the resources they need to compile it the same, and I feel that that's a problem that the TeX community as developers should be dealing with because we want our TeX source documents to be archival and not rely on the massive resources of Overleaf. We'd like to be able to say, here's the secure hash that gives you the true in repository that tells you exactly what you need.

Now that's a technical thing, if you like, on one side, but those two or three things are examples I think where we're really grateful that Overleaf is solving the problem for us, but perhaps we would be better off suffering a bit ourselves and solving it ourselves, so I'm not sure who I'm talking to and whether I've offended anybody, but I hope you can accept, hope you can all accept I'm representing a significant point of view, and I'd like you to comment and give your response.

**JH:** You certainly haven't offended me. I think it's a very good question. Certainly, when WriteLaTeX was originally created, it was to solve a problem that we had within our research group and I think it's great that it's helped — it's grown and helped — so many other people get into LaTeX and use it, and this sort of helps LaTeX, helps more people become aware of it. We use the TeX Live distribution on Overleaf and a lot of the new things that people enjoy in Overleaf, new things that people have created, just to pick an example from someone that's here, you know samcarter's `tikzducks` packages, people enjoy that and they like using those things, and that is through the TeX community.

From my perspective, I'm a researcher originally and I think it is important that we try both not lose sight of wanting to get new users using things, but then also like you say, things compile with certain versions, and, you know, we've had... and this is where some of the discussions around, should there be a way of saying when something compiles what it compiled with, and how you get that, how you could recreate that exact document, what exactly did it compile with. The way I look at it is that these questions are important from a certain perspective, and to certain people, and if we can help find a solution which means that a LaTeX document can take with it enough information for that to be compiled that instant, then I think that would be very valuable for a lot of people. They're looking to write documents and they want to keep them, keep being able to write them, and so a lot of the work we do on compatibility... So, for example, when we release new TeX Live versions, you know, new documents use our new TeX Live version. Previous documents still continue to use the previous version they had, so that from an author's perspective, their document still compiles when they reload it. So I think a lot of the stuff we look at from that user interface and user experience perspective, is maybe something that can help with this.

But that was my first thought on it, and it is definitely worth thinking about.

**JF:** Thank you for your response. It's very helpful, and I think we're almost out of time.

**PN:** There's one very quick question, John, about support for ConTeXt, coming from Vít Novotný.

**JH:** Oh sorry, yes, working out the chat box!

So it is a good question and, and like you said, there are some interesting new things happening with ConTeXt.

It's not something we've discussed internally recently. We do support the other LaTeX engines, but ConTeXt is not something that we have supported, and it's not something we've worked on recently. It is certainly something we can look at again. But I guess it's not in the short term roadmap, so it's a piece we have to look at and then take a view on and, given the number of things already on the list, I imagine it wouldn't happen immediately. But if it *is* being used a lot, and I think if it is, if there are new things with it, then maybe it's time we have another look.

I see Frank has his hand up. And I know we're nearly at the end.

**Frank Mittelbach (FM):** What I wanted to ask your thoughts about is something like this: when TeX originally came into life and Don wrote his book, he wrote something like, "Join the user group" as Appendix J of *The TeXbook*. And in the early days user groups were sort of essential for basically everybody who was using TeX. People got together to get things going, and that was the way to do it. And out of that the user groups evolve, by providing services to worldwide users.

By that means they became less and less visible to users. Most users these days have no idea where the services come from that they actually consume in all kinds of ways, like having CTAN run by a handful of people doing nightly jobs to keep submissions in and so on and so forth, and the world turns and goes on. And what you did with Overleaf is, in some sense, if you like, a missed opportunity of the TEX community as a nonprofit organization to keep the whole universe going, and it's the right thing to do. And it is a step forward that we currently sort of have, to keep the whole ecosystem alive and kicking and improving.

I think it's a great step forward, but there is a foundation underneath which has or is the danger, I would think, of at least partially collapsing, and the services that you are now providing, and by this way, I think, actually largely enlarging the ecosystem in some sense, is underneath that, run through volunteers that are getting less and less possibilities to actually get this work managed financially, in other words. So what's your thought on how this is going on in the future, and how much do you think are companies that benefit from it, like Overleaf, obviously in some sense, because if that would not be there, it is not going to sort of evolve further.

How is this going to coexist and evolve together in some sense?

**JH:** Yeah, I think it is a really good question. I think it is worth exploring, like how we can help support some of, say, some of the initiatives and even not even the new initiatives, like CTAN which is an amazing [resource]. CTAN *is* amazing, and we make a small financial contribution, and I think we could probably do more. But equally I think we want to try and find a way to help it be sustainable in the long term as well because I think that's what's the most useful; right. It's not about short term things, it's about how, like you say, how we make sure that there is enough people coming into it, that will help continue it and how it can be sustainable. Not just from a financial perspective, but the people that are actually doing the work. We're very open to discussion on this. I know we haven't got time now, but we're very happy...

**FM:** It was certainly not meant as a question to resolve.

**JH:** Right.

**FM:** But it is something I think which is extremely important to sort of get a joint discussion on it in the future.

**JF:** I'd like to make a quick contribution to what Frank has said. The MathJax Consortium, I think that's what they call themselves, have managed very well to get funding and they're now part of the NumFOCUS group that funds a large amount of data science projects, and they get extensive funding from people who got real stakes.

I'd say that we have to look to, as a TEX community, we have to look to ourselves and the way we manage things and that TUG has an income of $100,000 a year and I don't think it's well spent and it's very hard to go begging for money when you're in that situation. That's controversial I know.

The other thing is about reproducible document compilation. The remark about ConTEXt actually made everything very, very clear.

Overleaf is providing reproducible document compilation, on a client-hub basis. On a client-hub basis we have reproducible document compilation in the same way that Subversion provides version control. There's a hub and there's clients, whereas Git is peer-to-peer, and I think the TEX community, as developers go, have got a real responsibility to develop peer-to-peer reproducible document compilation, and we do this with or without Overleaf; with Overleaf, I hope.

Yes, and that will make a lot of things much easier. It means that I can write a `beamer` presentation and send it out to people, just as a TEX file, which greatly makes, it would considerably reduce the latency of giving presentations, for example.

So I'll stop at that point.

**PN:** I'd like to thank John very, very much. We enjoyed this conversation and we hope to do this in the future again.

**JH:** Thank you for inviting me and thanks for running a great conference. I have a lot of YouTube stream to catch up on!

I do want to just highlight from the chat that `tikzbricks` already exists, apparently, and samcarter is already on it. So, if nothing else, we have inspired some LEGOTEX in the future.

It's been great to be here today, and I want to catch up on the other presentations and then continue the wider discussion on all of this, as we go.

**BV:** Thank you very much.

# The tikzlings package

samcarter

## Abstract

The Ti*k*Zlings package provides a selection of cute little animals (and other beings) which can be used in Ti*k*Z. The following article will give an overview of the available creatures, their options and how to customise them.

## 1 Introduction

For most users their first contact with LaTeX and friends is for something serious like writing a thesis or a report. However the LaTeX world has much more to offer. There are many enthusiastic users and authors for whom LaTeX is also a tool to create joyous documents like greeting cards or educational material for children.

The Ti*k*Zlings package is a collection of cute little creatures which can be used with Ti*k*Z in many different situations. Figure 1 shows a family portrait of all of them.

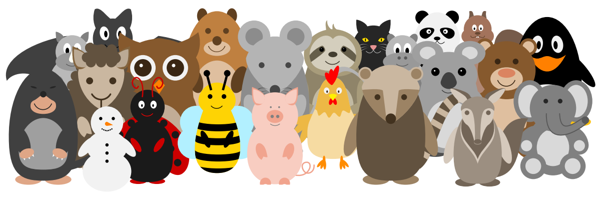


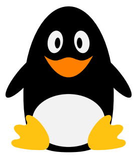
**Figure 1**: Family portrait of the Ti*k*Zlings.

## 2 Basic usage

The basic usage of the package is simple. After loading the package with `\usepackage{tikzlings}`, a command for each creature is available, which can be used in a `tikzpicture`:

```
\documentclass{standalone}
\usepackage{tikzlings}

\begin{document}
\begin{tikzpicture}
  \penguin
\end{tikzpicture}
\end{document}
```



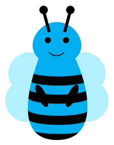
Besides the `\penguin` from the example above, the other available macros are `\anteater`, `\bat`, `\bear`, `\bee`, `\bug`, `\cat`, `\chicken`, `\coati`, `\elephant`, `\hippo`, `\koala`, `\marmot`, `\mouse`, `\moles`, `\owl`, `\panda`, `\pig`, `\rhino`, `\sheep`, `\sloth`, `\snowman` and `\squirrel`. And for the undecided, there is the macro `\tikzling`, which will draw a random Ti*k*Zling.

If for any reason, a users does not want to load the full Ti*k*Zlings package, there are also individual packages for all animals, which can be loaded via, e.g., `\usepackage{tikzlings-bears}`.

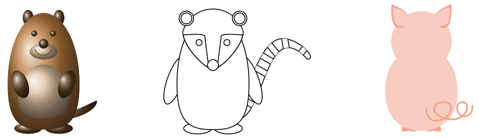
## 3 Options

To make them more individual, the Ti*k*Zlings come with several options for customisation. Via a simple key-value interface the user can adapt properties like the body colour:

```
\begin{tikzpicture}
  \bee[body=cyan]
\end{tikzpicture}
```



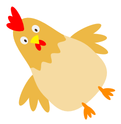
There are also options to make the Ti*k*Zlings appear three-dimensional (`3D`), only show their outlines (`contour`) or from behind (`back`):



The available options vary for the different creatures and giving a exhaustive list here would exceed the scope of this article, but a full list of all options can be found in the documentation [1].

Besides these options provided by the Ti*k*Zlings package, also all typical Ti*k*Z options can be used:

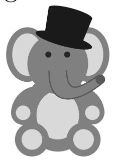
```
\begin{tikzpicture}
  \chicken[rotate=20]
\end{tikzpicture}
```



## 4 Accessories

The Ti*k*Zlings come with an assortment of accessories, which can be used as an optional argument:

```
\begin{tikzpicture}
  \elephant[tophat]
\end{tikzpicture}
```



As the Ti*k*Zlings come in many different shapes and sizes, not all accessories will fit them off-the-peg. Therefore accessories can also be added separately with the `\thing` macro, for which one can use all the typical Ti*k*Z options to scale and shift it until it fits the Ti*k*Zling.

```
\begin{tikzpicture}
  \squirrel
  \thing[harlequin,
    scale=0.65,yshift=1.05cm
  ]
\end{tikzpicture}
```

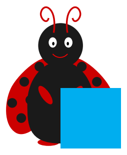
doi.org/10.47397/tb/42-2/tb131samcarter-tikzlings

For a full list of available accessories, please consult the package documentation [1].

## 5 Further customisation

Everybody for whom all these options and accessories are not enough, can further customise the Ti*k*Zlings. At their heart, they are just paths in a `tikzpicture` so any Ti*k*Z code can be added:

```
\begin{tikzpicture}
\bug
\fill[blue] (0,0)
      rectangle (1,1);
\end{tikzpicture}
```



To make such additions easier, the Ti*k*Zlings package provides several hooks for the user to add code at different layers of the drawing. For example one can use the `\mousehookbelly` macro to give the mouse a dress, which will be located between the belly and the arms:

```
\begin{tikzpicture}
\newcommand{\mousehookbelly}{%
  \fill[red!80!black]
  (0.55, 1.35) -- (0.65, 0.3) --
  (-0.65, 0.3) -- (-0.55, 1.35)
  -- (0.0, 0.9) -- cycle;
}
\mouse
\end{tikzpicture}
```

Other available hooks are

- `\`⟨*tikzling*⟩`hookbody`
- `\`⟨*tikzling*⟩`hookbackground`
- `\`⟨*tikzling*⟩`hookforeground`

## 6 Suggestions or problems

The package source is hosted at `https://github.com/samcarter/tikzlings`, along with a tracker for bug reports, feature requests or contributions.

## 7 Acknowledgements

I would like to thank the TEX Users Group for the possibility to present the Ti*k*Zlings package at their annual meeting and in particular the organisers for facilitating such an interesting conference despite the challenges of an ongoing world-wide pandemic.

My thanks also to all the people who have contributed code or ideas to the package — without their valuable help, the Ti*k*Zlings package would be nothing like it is today!

## References

[1] samcarter. *The TikZlings package — Version 0.8*, 2021. `ctan.org/pkg/tikzlings`.

◇ samcarter

---

# Texmlbus, a build system to convert documents to XML and other formats

Heinrich Stamerjohanns

## Abstract

Here I present an automatic open source build system that supports the conversion process of a collection of documents written in LaTeX or other TEX formats. With Texmlbus [4], the TEX to XML BUild System, documents can not only be converted to PDF, but also to other output formats — such as markup languages like HTML. In particular, conversion to XML, HTML and MathML is supported via LaTeXML. Texmlbus can schedule jobs among several workers (possibly on different hosts), extract and analyze the outcome of the conversion process of each document and store results in its own database. Result documents as well as statistics about the results of the build process can be easily retrieved using a web browser.

## 1 Introduction

LaTeX is the preferred document source for many scientists and authors who publish results that include mathematical formulas. In addition to print-oriented formats (PDF), there is also a need to export such documents into other formats, such as XML-based documents that more easily support additional services like search and navigation, as well as integrating the document or parts of it in web pages.

Automatic conversion of such collections of TEX documents can be a time-intensive task. The conversion result needs to be checked for errors and the visual appearance needs to be verified for each document. A build system that collects the result and status of conversions offers easily clickable links to result documents and log files, and collects and

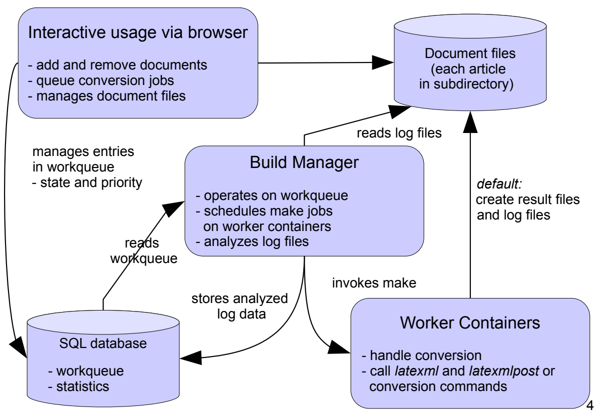

Interactive usage via browser
- add and remove documents
- queue conversion jobs
- manages document files
Document files (each article in subdirectory)
reads log files
manages entries in workqueue
- state and priority
Build Manager
- operates on workqueue
- schedules make jobs on worker containers
- analyzes log files
default: create result files and log files
reads workqueue
invokes make
stores analyzed log data
SQL database
- workqueue
- statistics
Worker Containers
- handle conversion
- call latexml and latexmlpost or conversion commands
4

**Figure 1**: Components of Texmlbus

Heinrich Stamerjohanns
doi.org/10.47397/tb/42-2/tb131stamerjohanns-texmlbus

show statistics can provide helpful support in order to manage such conversion tasks. Texmlbus is such a system; its main components are shown in Figure 1.

Using LaTeXML [3] as the default conversion processor, this system is based on the arXMLiv build system [5, 6], which I had written as a member of the arXMLiv group at Jacobs University Bremen. It was used to convert large collections of scientific publications of the Cornell e-Print archive ar$\chi$iv to XML and Content-MathML. That build system had not only been useful for converting documents, but also by generating statistics for conversion errors and warnings; thus our group was able to generate feedback to the LaTeXML developers on where to focus improvements.

While the conversion of ar$\chi$iv documents is now being done with CorTex [2] the original arXMLiv build system has now become Texmlbus. Texmlbus still uses LaTeXML as the main conversion processor, but can easily be extended to support other conversion processors and formats as well. It still can be used for conversions of thousands of documents, but now development is focused in particular on

- easy installation on any platform
- simple (interactive) use of system
- possibility for targets other than XHTML
- using the same targets with different systems

To make the system more easily installable and runnable on any platform, docker containers have been chosen. To simplify usage, interactive components have been added to the system. Documents can now be added, queued and removed via a web browser. It is also now possible to organize document collections in self-defined sets. This makes it easier to navigate through and work with many documents.

The new build system also supports targets other than XHTML. With only a few lines of code, it is possible to add additional targets such as specific XML-dialects or other validators to the system.

Also, so-called *stages* have been added to the system. A stage is defined as a combination of a target (e.g., XHTML) and a specific docker image that implements the conversion. It is therefore possible to have several stages that generate the same target, but with different versions (e.g., TeX Live 2020 and TeX Live 2021).

## 2 System

In order to ease deployment on any platform, the system runs inside several docker containers, which share a common file system containing the documents. There is a docker container for the build manager (the web server backend which hosts the application), a container for a relational database,

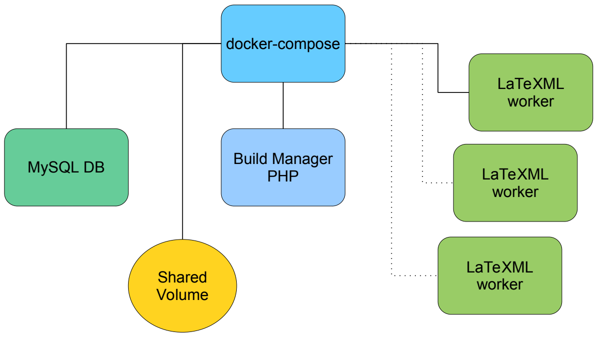
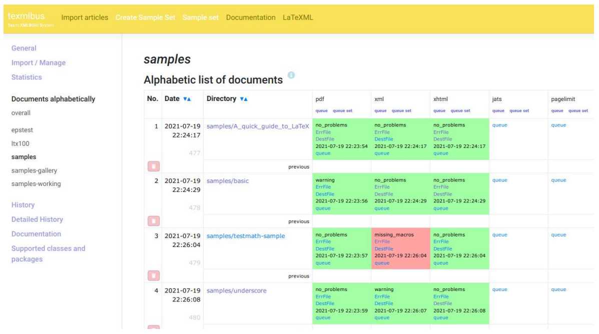


**Figure 2**: Docker setup using `docker-compose`



Documents alphabetically

overall

epstest

ltx100

samples

samples-gallery

samples-working

*samples*

Alphabetic list of documents

| No. | Date | Directory | pdf queue queue set | xml queue queue set | xhtml queue queue set | jats queue queue set | pagelimit queue queue set |
|---|---|---|---|---|---|---|---|
| 1 | 2021-07-19 22:24:17 477 | samples/A_quick_guide_to_LaTeX | no_problems ErrFile DestFile 2021-07-19 22:23:54 queue | no_problems ErrFile DestFile 2021-07-19 22:24:17 queue | no_problems ErrFile DestFile 2021-07-19 22:24:17 queue | queue | queue |
| | | previous | | | | | |
| 2 | 2021-07-19 22:24:29 478 | samples/basic | warning ErrFile DestFile 2021-07-19 22:23:56 queue | no_problems ErrFile DestFile 2021-07-19 22:24:29 queue | no_problems ErrFile DestFile 2021-07-19 22:24:29 queue | queue | queue |
| | | previous | | | | | |
| 3 | 2021-07-19 22:26:04 479 | samples/testmath-sample | no_problems ErrFile DestFile 2021-07-19 22:23:57 queue | missing_macros ErrFile DestFile 2021-07-19 22:26:04 queue | no_problems ErrFile DestFile 2021-07-19 22:26:04 queue | queue | queue |
| | | previous | | | | | |
| 4 | 2021-07-19 22:26:08 480 | samples/underscore | no_problems ErrFile DestFile 2021-07-19 22:23:59 queue | warning ErrFile DestFile 2021-07-19 22:26:07 queue | no_problems ErrFile DestFile 2021-07-19 22:26:08 queue | queue | queue |

**Figure 3**: Texmlbus document overview

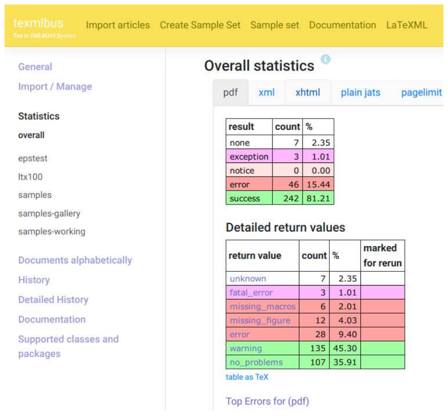
which stores information about the documents and result statistics, and one or more worker containers that actually run the conversion. The workers also run a minimal web server, so jobs can be invoked via API requests over HTTP. An overview of the docker setup is shown in Figure 2.

Documents can be either copied to the file system and then scanned for import or uploaded as zip files via a web browser. It is also possible to directly import and update files from Overleaf via a git-bridge.

Then, documents (or whole sets) can be added to a work queue interactively. The build manager operates on the work queue and schedules jobs on distributed worker containers. It keeps an internal list of available hosts (each container is a host) and distributes conversion jobs among these hosts. The conversion jobs on the worker containers are invoked via an API request over HTTP. A worker container typically runs a make process to invoke `latexml` and `latexmlpost` for the conversion to XHTML.

After the conversion has finished on the worker container, the build manager is notified and then collects data by analyzing log files and stores the result in the database. The results are then shown in an overview.

The build manager can, for example, collect the names of missing macros that are not yet supported



Overall statistics

| result | count | % |
|---|---|---|
| none | 7 | 2.35 |
| exception | 3 | 1.01 |
| notice | 0 | 0.00 |
| error | 46 | 15.44 |
| success | 242 | 81.21 |

Detailed return values

| return value | count | % | marked for rerun |
|---|---|---|---|
| unknown | 7 | 2.35 | |
| fatal_error | 3 | 1.01 | |
| missing_macros | 6 | 2.01 | |
| missing_figure | 12 | 4.03 | |
| error | 28 | 9.40 | |
| warning | 135 | 45.30 | |
| no_problems | 107 | 35.91 | |

**Figure 4**: Texmlbus statistics page

by LaTeXML or give an overview of files for which the conversion failed.

The document overview (see Figure 3) lists documents in collections (sets) and shows the conversion status for each stage. On that page the TeX source, error logs for each stage, and the result documents are reachable with one click. The backend can also create cumulative statistics of the result log data and creates lists such as top fatal errors or most missing macros on-the-fly.

To convert TeX documents to XHTML, LaTeXML needs so-called binding-files (with the suffix `.ltxml`). By using the list of missing macros one can easily determine which macros need more support in order to improve the conversion results.

It is also possible to easily extend the system and create additional stages that convert to other formats or use another TeX environment. As a stage is defined as a combination of target and container, one can easily create a container that uses a different TeX Live distribution. Therefore one can easily compare the results of different distributions and gather statistics about conversion results (Figure 4).

Furthermore, the system is not limited to conversions to other formats. Last year, W. Duivesteijn wrote about “How to cheat the page limit” [1] for conference papers and identified typical TeX commands that are used to circumvent page limits. With a few lines of code I have created a pagelimit stage that tries to find these typical commands in a document and categorizes the document accordingly.

## 3 Conclusion

The Texmlbus build system allows for easily converting LaTeX documents to XHTML and MathML or other destination formats, and gathers statistics about the conversion results. Since binding files’ coverage of LaTeXML is still not complete, this system helps to identify missing style file support for document collections and therefore gives feedback on where to focus development efforts.

Other converters can also be used for conversions. Docker containers that provide such converters need to be extended to provide some additional HTTP support so they can be invoked via an HTTP API. It should be simpler to directly plug in such containers, so additional stages can be added even more easily.

## 4 Acknowledgements

I am grateful to Overleaf for its support of the project.

Please email me with any problems or questions.

## References

[1] W. Duivesteijn, S. Hess, X. Du. How to cheat the page limit. *WIREs Data Mining and Knowledge Discovery* 10, Feb. 2020. `10.1002/widm.1361`

[2] D. Ginev. `CorTeX`: A general purpose processing framework for corpora of scientific documents. `https://github.com/dginev/CorTeX`

[3] B. Miller, D. Ginev. `LaTeXML`: A LaTeX to XML converter. `https://dlmf.nist.gov/LaTeXML/`

[4] H. Stamerjohanns. `texmlbus`: A build system to convert documents to XML and other formats. `https://github.com/stamer/texmlbus`

[5] H. Stamerjohanns, M. Kohlhase. Transforming the ar$\chi$iv to XML. In *9th International Conference, AISC 2008 15th Symposium, Calculemus 2008 7th International Conference, MKM 2008 Birmingham, UK, July 28–August 1, 2008*, S. Autexier, J. Campbell, et al., eds., Intelligent Computer Mathematics, pp. 574–582. Springer Verlag, 2008.

[6] H. Stamerjohanns, M. Kohlhase, et al. Transforming large collections of scientific publications to XML. *Mathematics in Computer Science* 3(3):299–307, 2010. `https://kwarc.info/kohlhase/papers/mcs10.pdf`

◇ Heinrich Stamerjohanns
Oldenburg, Germany
`heinrich.stamerjohanns (at) gmail.com`
`https://github.com/stamer/texmlbus`

# Continuous integration and TEX with Org-Mode

Rohit Goswami

## Abstract

Virtual or cloud computational resources in the form of build servers on public `git` hosting servers have become increasingly common. Herein we discuss how these may be leveraged for providing an asynchronous distributed collaborative workflow.

Additionally, we demonstrate how the Org-Mode markup language can be used to provide a user-friendly point of contact for novices and experts alike, and includes data analysis techniques.

## 1 Introduction

Since the acquisition of ShareLaTeX, and the integration with publishing houses, Overleaf has increased its prominence in academic circles. This has been accelerated by the global pandemic [8]. In the interests of preventing the promulgation of commercial monopolies we investigate the usage of continuous integration (CI) services to generate documents on the fly. At the same time, for many data-intensive studies, rapid analysis and plotting during the ideation phase has become de rigueur. This has led to the rise of Jupyter notebooks which meld analysis and context together in executable formats and have been a major driver in the rapid adoption of Python in scientific circles [5]. This can be seen also in the trend in publishing which has recently veered towards the adoption of such data-driven executable content [1, 6].

In these scenarios, the usage of TEX is often dependent on the programming language and environment used; e.g., R provides an almost complete format for preparing reports (RMarkdown and `knitr`) which exports to TEX [9, 12]. Here, we demonstrate the flexibility of the `org-mode` markup language, which can be extended in a straightforward manner to replace Jupyter notebooks for data analysis in a transparent, language agnostic manner.

## 2 CI setup

We will consider here the specific examples which are intended to run on the free "GitHub Actions" CI infrastructure. Given that most build-bots are configured in a similar manner, the key concepts are applicable to other setups as well. In broad strokes, a `texlive` installation controlled by a minimal `profile` is used for TEX, since the default build systems tend to be pinned to older distributions (e.g., Ubuntu 18.04 or even Alpine Linux).

### 2.1 Emacs and Org

To ensure that TEX exports are feasible on a CI without provisioning an entire `emacs` configuration in each repository, the following listing contains the minimal execution setup script. Links to more complete versions are given in section 4.4.

**Listing 1**: Minimal Lisp execution script

```
(require 'package)
(setq package-check-signature nil)
(add-to-list 'package-archives
  '("melpa" . "https://melpa.org/packages/") t)
(package-initialize)
(unless package-archive-contents
   (package-refresh-contents))
(package-install 'use-package)
(package-install 'org)

(dolist (package '(use-package))
(unless (package-installed-p package)
   (package-install package)))

(use-package org-ref
   :ensure t)

(require 'ox-latex)
(setq org-latex-packages-alist 'nil
 org-latex-minted-options 'nil
 org-latex-listings 'minted
 org-latex-default-packages-alist
  '((""     "graphicx"  t)
    (""     "lipsum"  t)))

(defun org-export-to-pdf-dir (files)
  "Export␣all␣FILES␣to␣latex."
  (interactive "ORG-->TEX")
  (save-excursion
    (let ((org-files-lst ))
      (dolist (org-file files)
        (message "***␣Exporting␣file
␣␣␣␣␣␣␣␣%␣s␣***" org-file)
        (find-file org-file)
        (org-latex-export-to-latex)
        (kill-buffer)))))

;; Export all org files from CLI
(org-export-to-pdf-dir argv)
```

This can now be executed on a CI as seen in Listing 2.

### 2.2 Caching and determinism

For continuous integration, control over dependencies is of paramount importance. In this workflow,

doi.org/10.47397/tb/42-2/tb131goswami-ci-org

**Listing 2**: Using Lisp exporter on the CI

```
- name: Generate TeX
  run: |
 emacs -q -nl -script \
 scripts/org2tex.el src/super.org
```

the primary dependency considered is the TeX installation. There have been many attempts to ensure reproducibility and automation of installation, ranging from language-specific measures like `tinytex` for R [11] to the alternate TeX-compatible systems like Tectonic [7]. There have also been recent CI-specific projects like the Island of TeX which provide Docker images for ensuring reproducibility [3]. Our approach to this problem is to leverage the standard TeX Live installation approach, along with caching to prevent unnecessary strain on the build servers. Packages are obtained on-the-fly from the Comprehensive TeX Archive Network (CTAN) via the `texliveonfly` package, which is a Python script for dependency resolution; it uses `tlmgr` to install packages needed for a given compilation.

## 3 GitHub bots

The workflow in the previous section can be augmented to use “Bots” to perform additional automated tasks. Beyond the convenience factor, the usage of GitHub allows for using the embedded rich text editor, and even Codespaces. Rendered documents can also be viewed on GitHub, which is a bonus. This section will assume the usage of GitHub as the platform of choice.

Other “Bots” not mentioned here include running `latexdiff` on pull requests to enhance the collaborative workflow.

### 3.1 Branches and deployments

To “deploy” the rendered document in a manner which is accessible from the web interface, a PDF may be deployed to an orphan branch which will be recreated each time, without history so as to not take up excessive space. Though this may be accomplished by manually adding a commit step, `actions-gh-pages` makes this even simpler, as shown in the Listing 3 fragment.

Pushing to a branch requires a personal access token. However, since each GitHub Actions runner automatically creates the `GITHUB_TOKEN` secret for the authentication of the workflow itself, this bot abstracts the authentication mechanism from the user.

**Listing 3**: PDF rendering fragment on CI

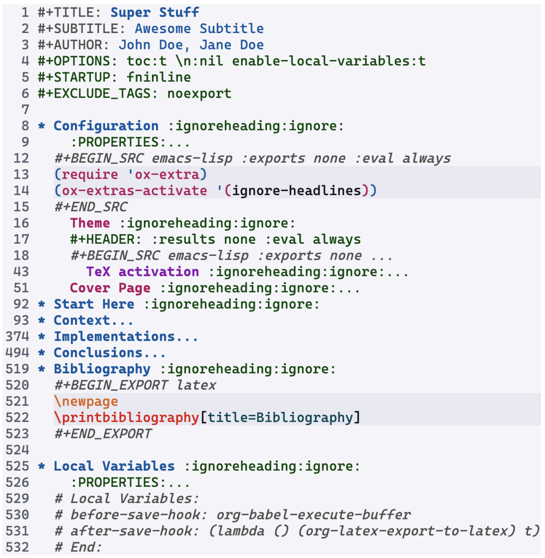
```
- name: Render to Branch
  uses: peaceiris/actions-gh-pages@v3
  with:
    github_token: |
        ${{ secrets.GITHUB_TOKEN }}
    publish_dir: ./pdfdir
    publish_branch: pdf
    force_orphan: true
```

## 4 Org markup and TeX

Generating a TeX layout with `org-mode` demands an understanding of the standard layout of LaTeX documents, and also the Emacs Lisp variables and hooks which control and modify the export process from `org` to TeX. The `org-mode` syntax is described in great detail in its manual at `orgmode.org/manual/`. Furthermore, syntax highlighting of `org-mode` is provided by extensions to most editors, including Visual Studio Code. Finally, `org-mode` is rendered for HTML previews on GitHub, GitLab, and other servers, making its use and adoption easier.

### 4.1 Standard modifications

The basic setup for working with `org-mode` files in general is depicted in Figure 1, where importantly, the `:ignore:` tags can be used to demarcate the document into logically consistent parts. With code folding and narrowing to trees, this allows for an optimally efficient setup.



```
#+TITLE: Super Stuff
#+SUBTITLE: Awesome Subtitle
#+AUTHOR: John Doe, Jane Doe
#+OPTIONS: toc:t \n:nil enable-local-variables:t
#+STARTUP: fninline
#+EXCLUDE_TAGS: noexport

* Configuration :ignoreheading:ignore:
    :PROPERTIES:...
  #+BEGIN_SRC emacs-lisp :exports none :eval always
  (require 'ox-extra)
  (ox-extras-activate '(ignore-headlines))
  #+END_SRC
    Theme :ignoreheading:ignore:
    #+HEADER: :results none :eval always
    #+BEGIN_SRC emacs-lisp :exports none ...
      TeX activation :ignoreheading:ignore:...
    Cover Page :ignoreheading:ignore:...
* Start Here :ignoreheading:ignore:
* Context...
* Implementations...
* Conclusions...
* Bibliography :ignoreheading:ignore:
  #+BEGIN_EXPORT latex
  \newpage
  \printbibliography[title=Bibliography]
  #+END_EXPORT

* Local Variables :ignoreheading:ignore:
    :PROPERTIES:...
  # Local Variables:
  # before-save-hook: org-babel-execute-buffer
  # after-save-hook: (lambda () (org-latex-export-to-latex) t)
  # End:
```

**Figure 1**: Standard document layout with hook and code folding

**Listing 4**: Sample demonstrating the usage of a “super” class

```
#+HEADER: :eval always :results none
#+BEGIN_SRC emacs-lisp :exports none
(org-babel-tangle)
(add-to-list 'org-latex-classes
'("super" "\\documentclass{super}"
("\\part{%s}" . "\\part*{%s}")
("\\chapter{%s}" . "\\chapter*{%s}")
("\\section{%s}" . "\\section*{%s}")
("\\subsection{%s}" .
 "\\subsection*{%s}")
("\\subsubsection{%s}" .
 "\\subsubsection*{%s}")
("\\paragraph{%s}" .
 "\\paragraph*{%s}")
("\\subparagraph{%s}" .
 "\\subparagraph*{%s}")))
(setq org-latex-packages-alist 'nil)
(setq org-latex-minted-options 'nil)
#+END_SRC
```

**Listing 5**: Snippet of a custom class

```
#+header: :results none
#+header: :tangle super.cls
#+header: :exports none
#+begin_src latex :eval always
....
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{super}
....
#+end_src
```

### 4.2 Archive emulation

Perhaps the simplest approach is to leverage the `.org` file as a plain text archive, which expands to a layout defined by `#+begin_export latex` and `#+end_export` directives for layouts in the document itself. For the preamble and to define what traditionally falls under the purview of `.cls` files, it is easiest to literally export the TEX commands to a file before loading it.

To accomplish this, it is necessary to provide an Emacs Lisp snippet which disables much of the underlying machinery, while augmenting it with the desired `.cls` as seen in Listing 4.

While the class itself can be within the same file, as shown in Listing 5, the `:eval always` header and the `org-babel-tangle` call in Listing 4 ensures that the file is (re)generated during the export process.

**Listing 6**: Mathematica code to export an image (solutions for a hyperbolic equation)

```
ClearAll[u, x, t, p];
len = 1;
pde = D[u[x, t], t, t] == \
16*D[u[x, t], x, x];
(* initial conditions *)
ic = {u[x, 0] == Sin[Pi*x], \
Derivative[0, 1][u][x, 0] == 0};
(* boundary conditions *)
bc = {u[0, t] == 0, u[len, t] == 0};
sol = u[x, t] /. \
First@DSolve[{pde, ic, bc}, \
 u[x, t], {x, t}];
p = Plot3D[sol, {x, 0, 1}, \
{t, 0, 1}, PlotPoints -> 30];
Export["images/q2aM.png",p];
Print["images/q2aM.png"]
```

### 4.3 Data driven development

The previous sections used Lisp to configure the Org and LATEX configuration in a reproducible manner. The `org-babel` tangling works for arbitrary languages as well, with minor workflow modifications. Consider the Mathematica code in Listing 6.

The key element in Listing 6 from the perspective of the end user is that the image must exist on disk, and the filename must be returned from the code block for it to be picked up in the `orgmode` workflow. This has the added benefit of being more WYSIWYG (what you see is what you get) compared to Jupyter Notebooks or RStudio which often render images which must be saved to disk manually with additional configurational changes.

### 4.4 Workflow samples

For full examples of a report generated with a CI-based collaborative workflow, interested readers are referred to:
`github.com/HaoZeke/ipam21_tqc_wg_report`
or slides for TUG 2021 at:
`github.com/HaoZeke/haozeke.github.io/blob/src/presentations/TUG2021/tug21rgpres.org`
and for a more involved example that showcases `org-mode` and its integration with R, see:
`github.com/HaoZeke/haozeke.github.io/blob/src/content-org/solutions/SR2/sol03.Rorg`

## 5 Discussion

The workflow outlined in this short article is not without its rough edges. The content presented in this article are aimed at groups of collaborators with at least one intermediate user of both the Unix shell and `git` to set up the initial workflow. It is expected, with the adoption of Wizards on Windows [4] along with template repositories [2] and GitHub Codespaces, that this workflow will have an even lower barrier of entry. In its current form, it is expected to be of interest to asynchronously communicating authors who are unwilling or unable to leverage solutions in commercial domains or those invested in the `org-mode` ecosystem.

The CI aspects are independent of the underlying method used to generate the TeX, and plain TeX workflows are also supported with ease, along with alternative markup methods like the R ecosystem's `bookdown` [10], or even other TeX generation techniques using `pandoc` are viable approaches.

However, the utility of using `org-mode` cannot be discounted, given that variables are retained between code blocks during execution. This means that programming languages may be freely mixed in an `org` document and tangled and executed subsequently by `org-babel` while retaining desired TeX layouts. Additionally, `org-mode` is "closest to bare metal" in that there are far fewer defaults compared to other approaches.

## 6 Conclusions

We have outlined a mechanism by which task-specific ad hoc TeX templates can be generated from `org-mode` files, thus enhancing collaboration with non-TeXnical colleagues. This allows for a transparent workflow for data analysis and programming via `org-mode` extensions such as the `babel` ecosystem.

Additionally, the use and abuse of build servers for collaborative TeX editing has been explored. Avenues for further development of such cloud-based distributed collaboration mechanisms have also been identified. The CI discussion can also be extended to encompass the automated testing of CTAN packages themselves, which would aid package developers.

### 6.1 Acknowledgments

The author would like to thank the attendees of TUG 2021 for fruitful discussions which enhanced the article.

## References

[1] C. Davidson-Pilon. *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference.* Addison-Wesley, 2016.

[2] GitHub Docs. Creating a repository from a template. `https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-repository-on-github/creating-a-repository-from-a-template`

[3] Island of TeX. The Island of TeX: Developing abroad, your next destination. *TUGboat* 41(2):182–184, 2020. `https://tug.org/TUGboat/tb41-2/tb128island.pdf`

[4] Microsoft. Wizards - Win32 apps. `https://docs.microsoft.com/en-us/windows/win32/uxguide/win-wizards`

[5] T.E. Oliphant. Python for Scientific Computing. *Computing in Science Engineering* 9(3):10–20, May 2007. `10/fjzzc8`

[6] R. Peverati. Fitting elephants in the density functionals zoo: Statistical criteria for the evaluation of density functional theory methods as a suitable replacement for counting parameters. *International Journal of Quantum Chemistry* 121(1):e26379, 2020. `10.1002/qua.26379`

[7] Tectonic typesetting system, Aug. 2021. `https://tectonic-typesetting.github.io`

[8] B. Veytsman. Using Overleaf for collaborative projects: First impressions and lessons learned. *TUGboat* 41(2):179–181, July 2020. `https://tug.org/TUGboat/tb41-2/tb128veytsman-overleaf.pdf`

[9] Y. Xie. *Dynamic Documents with R and Knitr.* CRC Press, Boca Raton, FL, second edition ed., 2015.

[10] Y. Xie. *Bookdown: Authoring Books and Technical Publications with R Markdown.* CRC Press, Boca Raton, FL, 2017.

[11] Y. Xie. TinyTeX: A lightweight, cross-platform, and easy-to-maintain LaTeX distribution based on TeX Live. *TUGboat* 40(1):30–32, 2019. `https://tug.org/TUGboat/tb40-1/tb124xie-tinytex.pdf`

[12] Y. Xie, J.J. Allaire, G. Grolemund. *R Markdown: The Definitive Guide.* CRC Press, Boca Raton, FL, 2019.

◇ Rohit Goswami
Science Institute
& Faculty of Physical Sciences
University of Iceland VR-III
107 Reykjavík, Iceland
& Quansight Labs
`rog32 (at) hi dot is`
`https://rgoswami.me`
ORCID 0000-0002-2393-8056

# The WEB to CWEB conversion of TEX

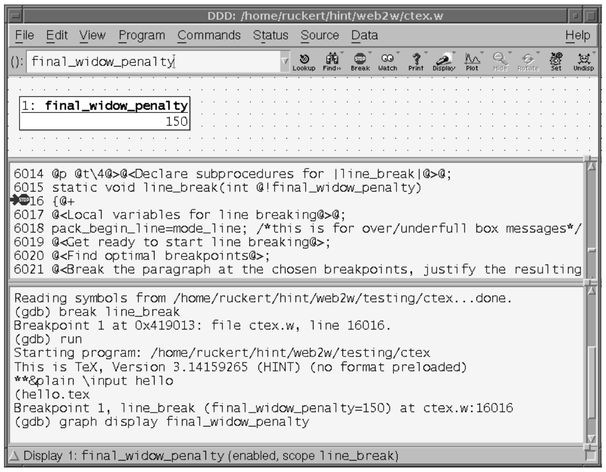
Martin Ruckert

## Abstract

This paper describes several aspects of the conversion of TEX's source code from WEB, based on Pascal, to CWEB based on C with web2w. It emphasizes those aspects that are relevant for obtaining a translation that can truly be regarded as source code and lends itself to modifications.

## 1 The advantages of CWEB

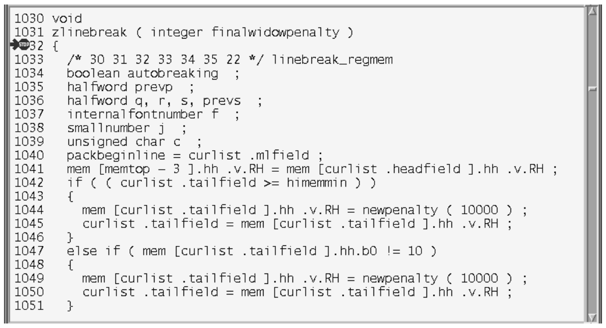
In the realm of programming environments, the C language enjoys very good support. Since the CWEB system of structured documentation [1] generates `#line` directives, this support extends also to programs written in CWEB. Figure 1 shows a debug session with the CWEB version of TEX where a breakpoint was set on the `line_break` function. After the run command, TEX asks for input and then stops in the CWEB file at the start of the `line_break` function. After a right click on "`final_widow_penalty`" in the source window, the debugger displays a menu that offers to display the variable in the data window. The traditional `web2c` translator expands `WEB` code to Pascal before translating to C. The debugger then shows fully expanded code in the source window as shown in figure 2.



```
@p @t\4@>@<Declare subprocedures for |line_break|@>@;
static void line_break(int @!final_widow_penalty)
{@+
@<Local variables for line breaking@>@;
pack_begin_line=mode_line; /*this is for over/underfull box messages*/
@<Get ready to start line breaking@>;
@<Find optimal breakpoints@>;
@<Break the paragraph at the chosen breakpoints, justify the resulting
```

```
Reading symbols from /home/ruckert/hint/web2w/testing/ctex...done.
(gdb) break line_break
Breakpoint 1 at 0x419013: file ctex.w, line 16016.
(gdb) run
Starting program: /home/ruckert/hint/web2w/testing/ctex
This is TeX, Version 3.14159265 (HINT) (no format preloaded)
**&plain \input hello
(hello.tex
Breakpoint 1, line_break (final_widow_penalty=150) at ctex.w:16016
(gdb) graph display final_widow_penalty
```

**Figure 1**: Debugging CWEB code with GNU **`ddd`**: true source code.



```
void
zlinebreak ( integer finalwidowpenalty )
{
  /* 30 31 32 33 34 35 22 */ linebreak_regmem
  boolean autobreaking  ;
  halfword prevp  ;
  halfword q, r, s, prevs  ;
  internalfontnumber f  ;
  smallnumber j  ;
  unsigned char c  ;
  packbeginline = curlist .mlfield ;
  mem [memtop - 3 ].hh .v.RH = mem [curlist .headfield ].hh .v.RH ;
  if ( ( curlist .tailfield >= himemmin ) )
  {
    mem [curlist .tailfield ].hh .v.RH = newpenalty ( 10000 ) ;
    curlist .tailfield = mem [curlist .tailfield ].hh .v.RH ;
  }
  else if ( mem [curlist .tailfield ].hh.b0 != 10 )
  {
    mem [curlist .tailfield ].hh .v.RH = newpenalty ( 10000 ) ;
    curlist .tailfield = mem [curlist .tailfield ].hh .v.RH ;
  }
```

**Figure 2**: Debugging **`web2c`** code with GNU **`ddd`**: fully-expanded code, difficult to read.

Another advantage of the CWEB version of TEX is the simple tool chain. From a `ctex.w` source file, `ctangle ctex.w` produces `ctex.c`, and `gcc -o ctex ctex.c` produces the `ctex` executable; no additional conversion commands are needed. For experimenting with your own TEX, `ctex.w` can be modified — preferably with a change file — and the debugger has no difficulties switching the source window between `ctex.w` and the change file as needed.

## 2 From web2w version 0.4 to 1.0

In the last three years, the development of the HINT project would not have been possible without the CWEB version of TEX's source code produced with `web2w` version 0.4 [3]. During development, some shortcomings have become apparent, which I address in the new version 1.0 as described below.

### 2.1 Creating a header file

Combining TEX's source code with C code residing in separate source files frequently requires access to TEX's functions and variables through "extern" declarations. Further, because of TEX's extensive use of macros, any substantial reuse of TEX's code also requires reusing TEX's macros. For this purpose, the new converter can optionally create a header file containing TEX's macros, TEX's ⟨Constants in the outer block⟩, its ⟨Types in the outer block⟩, and finally a selection of "extern" declarations. The names of the selected extern variables and functions are taken from a text file given on the command line. As a side effect, all the remaining functions and global variables in TEX's sources are then declared "static".

As an extreme example, assume that you want to reuse TEX's `x_over_n` function: you can create a text file with a single line just containing the name `x_over_n` and use it to convert `tex.web` to `ctex.w`. Then run `ctangle` on `ctex.w` to get `ctex.c` and `ctex.h`. The file `ctex.h` will then end with the line "`extern scaled x_over_n(scaled x,int n);`".

Because all the other functions and variables of TEX are declared static, any decent C compiler can figure out that almost all of these are dead variables and dead functions. Running "`gcc -O3 -c ctex.c -o ctex.o`" will therefore produce a surprisingly small object file of just 1520 bytes. Analyzing the object file with GNU `nm` reveals:

doi.org/10.47397/tb/42-2/tb131ruckert-cweb

| Size | Type | Name |
|---|---|---|
| 0000000000000001 | b | `arith_error` |
| 0000000000000004 | b | `rem` |
| 0000000000000089 | T | `x_over_n` |

The object file contains two variables, one byte for the boolean `arith_error`, four bytes for the integer `rem`, and 89 bytes for the function `x_over_n`. All the rest has been removed by the compiler. You can link `ctex.o` to your `main` program without incurring any unnecessary overhead. If you want access to `arith_error` and `rem`, just add two lines with these names to your text file.

## 2.2 Eliminating the static string pool

Another spot that needed further attention is TEX's string pool. Strings enclosed in C-like double quotes receive special treatment by `tangle`: the strings are collected in a string pool file and replaced by string numbers in the Pascal source. In `ctangle` no such mechanism is available.

The previous converter replaced literal strings by section names, which were then expanded to an index in the `str_start` array. So

```
primitive("par",par_end,256);
```

became

```
primitive(<"par">, par_end, 256);
```

together with "`<"par">=444`" following in the appendix. With a suitable static initialization of the `str_start` and `str_pool` array, this had the desired effect and it was quite readable. For a limited subset of the literal strings — those used exclusively for printing — the converter took extra action to keep them in the code as literal C strings. This had the advantage that these strings were easier to change when modifying the generated `.w` files.

When implementing the HINT viewer, however, it turned out that TEX's entire string pool was still necessary for compilation; the code inherited from TEX still contained a few references to the string pool. This seemed unnecessary because the HINT viewer does not use TEX's control sequences or string handling functions. When implementing version 1.0, I started to further reduce the number of strings entering the string pool until only the names of control sequences remained in the string pool. With these names, the first argument of the `primitive` function remained a string number.

Defining new TEX primitives in change files is, however, common for extensions of TEX, and with `ctex.w` this had proved to be a bit cumbersome. You could not just write

```
primitive(<"newname">, ...);
```

but also needed to define the section name such that it would expand to a string number, which depended on the initialization of the `str_start` array, which in turn depended on the initialization of the `str_pool` array. A cumbersome and error-prone process.

So in the end, I decided to eliminate the static initialization of the string pool entirely. In version 1.0 the string pool is initialized at runtime with the first 256 single character strings and the empty string. All other strings are added during TEX's initialization, for example by calling `primitive("par", par_end, 256)`. The advantage is simplicity and readability; the disadvantage is the overhead in time and space because names of control sequences will now exist twice: the static string that is the argument of "`primitive`" and its copy in the string pool.

Here is some data on the incurred overhead to justify the decision: Version 0.4 already reduced the initial number of strings in the string pool from 1044 to 730 and the initial size of the string pool from 22742 bytes to 4700 bytes. Further reductions in version 1.0 left 611 strings with a total of 3701 bytes in the string pool — still without simplifying the addition of new primitive control sequences. The next logical step was abandoning the static initialization of the string pool altogether. Two alternatives came to my mind: removing the string pool completely or switching to a dynamic initialization of the string pool. I decided on the second alternative for the following reasons:

- Dynamic initialization adds overhead in time and space. The space overhead, however, is small (about 1% of the executable's size) and the time overhead is incurred only in the `INITEX` version of TEX.
- Dynamic initialization makes addition of new string literals as simple as possible.
- Dynamic initialization simplifies the implementation of `web2w`.
- Keeping the string pool avoids changing TEX's data structures and algorithms.

## 2.3 64-bit memory words

The biggest problem with the previous `ctex.w` was the limitation of a 16-bit pointer type, allowing access to at most $2^{16}$ of TEX's memory words. To run a typical LATEX job, loading only a few of the most common packages, this is usually insufficient. And for many people, TEX is just an abbreviation for LATEX. Hence, an implementation of TEX that is restricted to 16-bit pointers is a nice research project but is not suitable for processing practical LATEX workloads.

Allocating bigger arrays is not the problem; the problem is the storage space needed for the array indices. Indices are stored in a `halfword` which in version 0.4 was a `uint16_t`. Two `halfword`s make up a `memory_word`. These data structures needed redefinition and it was unclear how the packing and unpacking of memory words would be affected by the change. Still, I expected that changing `max_halfword` should be feasible without creating too many complications, because I remember having seen in the 1970s a TeX implementation using 48-bit words. Because TeX tests "`if 2 * max_halfword < mem_top - mem_min then bad:=41;`", the value of `2 * max_halfword` must not produce an overflow. So I changed the value of `max_halfword` to `0x3FFF FFFF`.

After that, `web2w` ran without complaint. My patch file needed a few changes, such as replacing a `uint8_t` by a `uint16_t` at one place and a `uint16_t` by a `uint32_t` at another place, until, to my own surprise, the GNU C compiler would again compile `ctex.c` without further errors or warnings and even the infamous TRIP test had no objections.

### 2.4 Minor changes

#### 2.4.1 C-style macros

Avoiding name conflicts with the long list of TeX's macros — including common names like "x0", "day", "pop", "link", "next", "name" — was a constant concern when working on the HINT project. In the C language, there is no separate name space for macros, but it is common practice to avoid conflicts between macros and variables, functions, or other identifiers by using all uppercase names for macros. So version 1.0 implements a command line option that makes macro names use uppercase letters.

Changing all macro names to upper case does, however, impact the visual appearance of the TeX program considerably. Therefore this replacement is optional.

The `WEB` system allows defining parameterized macros using a single `#` sign to mark the insertion point(s) for the parameter text. This does not prevent you from passing multiple parameters because commas are allowed in the parameter text. But at every insertion point, the complete parameter text—with all its commas—is inserted. To overcome this restriction, TeX resorts to "tail calls" in it macro definitions. An example is TeX's definition of `char_info`:

```
@d char_info_end(#)==#].qqqq
@d char_info(#)==
    font_info[char_base[#]+char_info_end
```

which is used as

```
char_info(f)(c)
```

The `char_info` macro ends with `char_info_end`, the tail call, without specifying the parameter for it.

`web2w` is a source code converter. It strives to produce readable source code as its output. While Pascal does not know about macros, they are a common feature of C and there is a well-established way of using them. So translating the above literally to C does not yield code that has the right "look and feel". So, version 1.0 of `web2w` now implements the unrolling of the tail calls and generates true C-style macro definitions:

```
@d char_info(A, B) \
    font_info[char_base[A]+B].qqq
```

which is used as

```
char_info(f, c)
```

As another improvement, `web2w` now counts the number of uses and eliminates macro definitions that are not or — as in the case of `char_info_end` — are no longer used.

#### 2.4.2 Placing the `case` keyword

Respecting the common coding style of C also demanded an improvement in the placement of C's `case` keyword. In the code for processing a ligature or kern command, TeX uses case labels of the form "`qi(1),qi(5): ...`" where the macro `qi(A)` is defined as `A+min_quarterword`. From this, `web2w` version 0.4 produce:

```
    qi(case 1):  qi(case 5):  ...
```

which is correct C code but looks odd. Finding the right place for the `case` keyword is not trivial; for example, it would be an error to place the keyword before the "`A`" in the expansion text of macro `qi`. Version 1.0 now produces the better-looking:

```
    case qi(1):  case qi(5):  ...
```

#### 2.4.3 Mixed arithmetic with signed and unsigned integers

In Pascal, comparing of two integer expressions will always return a correct value, because Pascal maps both operands onto a common ordinal type, "large enough" for both operands, before comparing them.

This is different in C, where the rules for implicit type casting of operands are complicated and comparing a signed and an unsigned integer might not produce the expected outcome. Fortunately, the C compiler will, with the right warnings enabled, emit complaints about signed/unsigned comparisons. Similar problems occur when signed and unsigned integers are mixed in other arithmetic operations. Version 0.4 tried to replace a Pascal subrange type with the smallest standard C integer type that is large enough. So

```
    beta:1..16;
```
became `uint8_t beta;` and
```
    shown_mode:-mmode..mmode;
```
was converted to `int16_t shown_mode;`

The information about the true range of values is unavoidably lost by this translation. Considering the problems caused by the constant mix of different integer types, it seems preferable to forgo the approximation with the smallest possible C integer type and choose a plain `int` wherever this is sufficient. This is the approach taken by version 1.0 which eliminated all (if I can trust the compiler) problems with mixed integer expressions.

## 3 `ctex` and TEX Live

While `ctex.w` is a complete and functional implementation of TEX, it does not offer the functionality and amenities that users expect from a modern TEX engine.

When running `ctex`, probably the first thing you observe is that `ctex` will not search the "usual" directories for TEX's font metric files. As described in *The TEXbook*, only the current directory and the subdirectory `TeXfonts` are searched. The directories `TeXformats` and `TeXinputs` are searched for formats and input files. You are probably also accustomed to specifying input files and various options on the command line, but `ctex` will ignore your command line and present you, after displaying the banner, with a plain "`**`" prompt.

For a modern TEX engine, the `kpathsearch` library has become the standard to find all sorts of TEX-related files, and the TEX Live distribution has promulgated de facto standards on command line functionality.

When you try to run LATEX, sooner or later you will also find out that quite a few LATEX packages will assume that some primitive control sequences are present that are not specified in *The TEXbook* but are part of $\varepsilon$-TEX.

The extensions of $\varepsilon$-TEX are happily straightforward to obtain. Using the program `tie`, you can apply `etex.ch` to `tex.web` and obtain `etex.web`. Then apply `web2w` to get `etex.w`. To provide file searching with the `kpathsearch` library and a usable command line, another change file, `ktex.ch`, is part of the `web2w` distribution. It can be applied to `etex.w` to obtain `ktex.w`.

To fully support LATEX even more primitive control sequences are necessary [2]. At the time of writing, another change file is in preparation to add these control sequences to `ktex.w`. It is planned that `ktex.w` will be part of the next TEX Live distribution.

## 4 Conclusion

With `web2w` I have tried to achieve a source code to source code translation of TEX that strives to provide TEX source code in a form that is as close as possible to Donald Knuth's original source code while at the same time is supported by modern programming environments using the C programming language. While there are still many improvements to be made, `ctex.w` in its present form is well-suited to compile, run, and interactively study TEX. It also provides a good basis for conducting experiments with TEX, trying new extensions of TEX, or using parts of TEX in other software projects.

With `ktex.w` an extended TEX will be available that can cope with serious workloads. I have developed `ktex.w` primarily for using it as a basis for HiTEX, a new specialized TEX engine [4]. At the outset, I had no plans for making `ktex.w` available to the public, but while `ktex` is not intended for the average TEX or LATEX user, it may serve others as a basis for their development of specialized versions of TEX.

## References

[1] Donald E. Knuth and Silvio Levy. *The CWEB System of Structured Documentation.* Addison Wesley, 1994. `https://ctan.org/pkg/cweb`.

[2] LATEX Project Team. LATEX news, issue 31, February 2020. *TUGboat*, 41(1):34–38, 2020. `https://tug.org/TUGboat/tb41-1/tb127ltnews31.pdf`. Section "LATEX requirements on engine primitives". See also: `latex-l` thread of April 20, 2021, "LaTeX required primitives: some questions", at `https://listserv.uni-heidelberg.de/cgi-bin/wa?A0=latex-l`.

[3] Martin Ruckert. Converting TEX from WEB to cweb. *TUGboat*, 38(3):353–358, 2017. `https://tug.org/TUGboat/tb38-3/tb120ruckert.pdf`.

[4] Martin Ruckert. HINT: Reflowing TEX output. *TUGboat*, 39(3):217–223, 2018. `https://tug.org/TUGboat/tb39-3/tb123ruckert-hint.pdf`.

◇ Martin Ruckert
Hochschule München
Lothstrasse 64
80336 München
Germany
`martin.ruckert (at) hm dot edu`

# Plane and simple: Exploration of machine interaction with text type for visual-based navigation systems

Oliver Austin

## Abstract

Air travel provided the zoom for society before we had to 'Zoom'. However, at the most critical stage of flight, when the pilot and plane are coming to land, readability of runway designators is of utmost importance. While the methods of marking have traditionally been white paint on blacktop written in the standardized font used by the International Civil Aviation Organization (ICAO), with the aviation industry beginning to focus on the integration of fully autonomous systems into a variety of vehicles, it raises the question of whether the current font is suitable for visual-based navigational systems. This paper consequently examines the ability of machine learning software to read, learn, and recognize digits 0–9 and letters L, C, and R across a variety of fonts.

## 1 Introduction

On airport runways, there are a large variety of markers. In this paper, one group of these markers (i.e., Runway Designators) is examined to test their compatibility with autonomous aerial vehicles using Visual-Based Navigational Systems.

## 2 Runway designators

Runway Designators are a critical part of runway infrastructure. These markers, comprised of two numbers and in some cases, a letter, represent the three-digit magnetic heading of a runway and its relation to other runways at an airport. In order to simplify the three digits to two, the three digit magnetic heading is rounded to the nearest ten degrees of heading, and then the third digit is omitted. In the case that there is more than one runway at an airport all with the same heading, the characters L, C, and R are used to designate the Left, Center, and Right runways respectively.

## 3 Application of machine learning

In machine learning, there are three major steps that need to be taken to get an output, which can then be used by a system that has an integrated machine learning application. In the case of this project, the main focus is the use of a machine learning application in Visual-Based Navigation System for aircraft. Following are the major steps that need to be taken to achieve some form of output.

### 3.1 Collecting data sets

In this project, two different data sets were used, one comprised of images of the characters 0–9, L, C, and R in various fonts, referred to as the Multi-Font Data Set, and a second comprised completely of real-world pictures of runway designators, which are the same characters as the Multi-Font Data Set. Both of these sets were separated by character, which subsequently created the different classes used by the machine learning algorithm throughout the next steps.

### 3.2 Training the models

The next step in the process is to train the models using the previously obtained data sets. Each data set gets its own model file, which stores the patterns that a machine learning algorithm finds when sifting through the data. Each model is trained as a result of the machine learning algorithm passing through a data set a predetermined number of times (the quantitative unit for one pass being an epoch), looking for similarities between the individual pieces of data. Once training has been completed, or an increase of epochs fails to yield any more improvement, the findings are exported to the model file.

### 3.3 Interacting with new data

Once the model file has been exported, another machine learning algorithm can use the model to interact with new data. When applied, the algorithm is able to make predictions as to which class it thinks the new pieces of data should be put into, along with a confidence percentage specifying how similar the new data is to the patterns stored in the model. This information can then be used in many different ways to make informed decisions, such as whether or not an aircraft is lined up on the right runway.

## 4 Upcoming work

A second phase of this project will closely analyze the results obtained from the current data sets and discuss ways to improve them from both the design and computing standpoints.

## 5 Acknowledgements

Special thanks to Jennifer Claudio for helping see that the project stayed on track and seeing it through to where it is today, and Anthony Tieu, Kirsten Clemente, Aditi Jha, Arvin Singh, and Thu Thao Nguyen for helping with the preparation of data for the initial data set.

◇ Oliver Austin
San Jose, California, USA
`hagridshome04 (at) gmail dot com`

doi.org/10.47397/tb/42-2/tb131austin-plane

# My personal journey into creating word search puzzles in Cyrillic and Arabic using multilingual support in LaTeX

Jennifer Claudio

## Abstract

Word search puzzles are a fun pastime and can be a helpful learning tool for spelling and letter recognition. After a brief history and educational rationale, I present my personal exploration of the *babel* and *polyglossia* language packages for LaTeX with the production of puzzles in Russian and Arabic.

## 1 Introductory history

Word search puzzles are fun pastimes, and their existence tends to be attributed to one of two claims. Normal E. Gibat, an American, was documented in 1968 as having generated a word search game in order to entice readers to interact with the publication known as the Selenby Digest. The intention was to create customer loyalty by publishing answers and new mixes in the subsequent issues. Due to the simplicity of the puzzles, Oklahoma teachers who were readers of the Selenby Digest were drawn to them for their own classroom use, and requested personalized puzzles to share with their students.

Pedro Ocón de Oro of Spain was also credited with producing word soup puzzles in the 1960s. In his puzzles, words were hidden in randomized letters, and solvers had to find the words. Other word puzzle inventions attributed to Pedro Ocón de Oro include the cuadrograma and the oconogram, of which there is evidence of publications in 1968 and 1976.

## 2 Educational and socio-emotional value in schools

Today, educators often use such puzzles as vocabulary recognition practice for students of all ages. In early learning, letter and word pattern skills can be built using word searches with a published list, and later used in conjunction with students performing activities such as fill-in-the-blanks to generate their own word lists to solve within a hidden puzzle to reinforce the word. Students who are working on early language acquisition, students with special needs, and students who are learning jargon-heavy content are often assigned word search puzzles. It is important to note that the value of the word search puzzle lies in the ability to develop word recognition.

In educational psychology, word search puzzles are also acknowledged as helpful for building student confidence. While word search puzzles are not well-received as a standalone activity for high-achieving, older students done within their dominant language, the word search puzzle does offer a stepping-stone opportunity to encourage completion of other tasks. The positive reinforcement of “success” with word search puzzles encourages students who traditionally struggle in school to continue participating in other written tasks or worksheets.

Non-educational uses of word search puzzles include incorporation by student mental health services in helping students with focusing and coping exercises. Word search puzzles used in these situations often contain words with positive or otherwise uplifting meanings, and tend to have large text. The puzzles themselves are considered non-intimidating to students, but require sufficient attention to distract a student from immediate anxieties until a social worker, school psychologist, or intern is available to help a student.

## 3 World languages

Most school systems world-wide endorse, and often require, the study of a foreign language. In California (United States), typical language offerings at the high school level are Spanish and French. Depending on the region and relative language prevalence, languages such as Vietnamese might also be offered. Students who seek an opportunity to earn college credits during high school may also take the annual Advanced Placement (AP) language and culture exams for Chinese (Mandarin), French, German, Italian, Japanese, Spanish, and Latin. (Tangentially, a growing number of high schools are also offering American Sign Language to qualify for state requirements for world languages.)

To provide language authenticity, teachers of these languages often rely on their personal experiences with the language for instruction. These strategies could include using their own native childhood resources, integration of cultural practices innate to the language, recitation of prose and poetry, and examination of art that may reflect elements of the language itself.

Public information on the World Wide Web, language apps, and AP course materials complement textbooks and support teachers, but there are still gaps in the ease of control of customized worksheets and puzzles for students. In multilingual vocabulary building tasks, especially those that are associated with non-Latin alphabets, teachers are more likely to work with resources that are readily available rather than create something personalized for their style or teaching objectives.

With the exception of Chinese and Japanese, the AP exam thus far only offers exams and curriculum

doi.org/10.47397/tb/42-2/tb131claudio-puzzles

for content that uses the Latin alphabet. This is in contrast to the top ten most spoken languages of the world, typically ordered as English, Mandarin, Hindi, Spanish, French, Arabic, Bengali, Russian, Portuguese, and Indonesian.

## 4 Exploring multilingual word searches

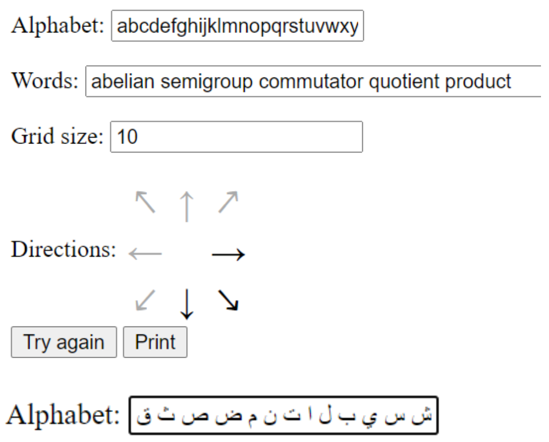
As a hobbyist of learning different alphabets and, ideally and with continued persistence, languages, I came across a book that taught Arabic writing. The first word search puzzle within the book, however, had errors and I could not find all of the words, a flaw that was verified by two native readers of Arabic. I enjoyed the puzzle and its intentions, especially since learning the Arabic alphabet is very different experience from English, particularly the difference of letter forms in initial, medial, and terminal positions and letters in their singular forms. I hoped to generate my own word searches in order to practice becoming familiar with the alphabet in this manner.

As a teacher, I am familiar with finding "quick-and-easy"[1] resources on the web, but I found that web-based word search generation is often limited to characters of the Latin alphabet.

Anecdotally describing a typical teacher, it is common to generate a word search puzzle using resources from the first-hit retrieved from a search engine, taking a screenshot of the puzzle that is generated, and pasting that image into a word processor document. Unfortunately this method not only results in a low-resolution capture, but it renders the teacher unable to edit the puzzle except by hand, especially if they are adding diacritics such as for Vietnamese, Spanish, or German.

Alternatively, there are teachers who cut and paste the letters into a text editor, but they, too, run into formatting and tabbing issues, which makes this method less convenient. The most time-consuming method would be for a teacher to manually create a grid, place letters of a word, then use filler letters to create the word soup.

The word search generator made available by Christian Lawson-Perfect on GitHub (`github.com/christianp/wordsearch-generator`) provides an excellent solution to this problem, and to my personal delight, accommodates non-Latin characters. A user enters words, independent of language or alphabet, chooses a grid size, selects the direction options of the hidden words, and rolls the puzzle (Figure 1). An option to input custom filler letters was recently implemented, consequently enabling the filler letters to easily integrate non-Latin alphabets.


Alphabet: abcdefghijklmnopqrstuvwxy

Words: abelian semigroup commutator quotient product

Grid size: 10

Directions: ↖ ↑ ↗ ← → ↙ ↓ ↘

Alphabet: ش س ي ب ل ا ت ن م ض ص ث ق

**Figure 1**: Input for Lawson-Perfect's puzzle generator, including (last line) the alphabet used for filler letters.

```
\documentclass{standalone}

\usepackage[thinlines]{easytable}

\begin{document}

\begin{TAB}(e,15pt,15pt){|c|c|c|c|c|c|c|c|c|c|
A & A & N & H & T & C & G & E & Y & L \\
S & B & E & C & S & O & A & D & N & Q \\
E & E & P & L & E & M & O & H & T & U \\
M & L & R & B & O & M & I & L & I & O \\
I & I & O & E & E & U & E & E & N & T \\
G & A & D & O & B & T & L & T & R & I \\
R & N & U & C & E & A & C & A & E & E \\
O & I & C & R & A & T & G & O & D & N \\
U & T & T & L & A & O & D & F & F & T \\
P & S & N & T & H & R & A & L & E & T \\
\end{TAB}

\end{document}
```

**Figure 2**: Example LaTeX output from puzzle generator (template line abridged).

## 5 Word searches, TUG, and LaTeX

Happily, Christian's word search generator also produces LaTeX markup (Figure 2). I was introduced to the TeX Users Group community in 2008, but I still have not quite truly used TeX or LaTeX. This project seemed to be a natural and low-risk starting point, and had the additional value of being something that I could share with my co-workers. The

[1] While it might seem natural to question why teachers cannot typically make the effort to find better solutions for things such as word search puzzles, or to create their own solutions, it must be considered that a teacher's time is truly not their own. During school hours, time is spent teaching or interacting with students. Preparatory periods during the school day are spent on follow-ups with families, supporting students with special needs, safety management, and more. Teachers often spend time well past work hours to ensure that all legal rights are met for students and to run extracurricular programs for students. Time pressure inevitably results in some teachers choosing "good enough" options for a task.

markup made available by Christian's word search generator allowed me the opportunity to tweak something existing rather than beginning from scratch. Although it is most certainly true that plenty of tutorials and learning opportunities exist, practice by manipulating pre-generated content offered an entry point with no excuse to avoid it.

I chose the Overleaf environment as my editor. Being able to compose while frequently recompiling to see my output made it conveniently possible to try and undo small changes while easily seeing any associated errors or warnings. Additionally, device independence removed the need for me to question working versions spread across machines. I imagine, though I do not yet have experience working collaboratively in Overleaf, that it will confer convenience if I am ultimately working with other authors or contributors on a project.

My first step involved looking at the automatically generated markup. I cut and pasted the default content into a new Overleaf document and compiled. As expected, a word search puzzle was generated without errors. My second step was to try generating an equivalent word search puzzle in another language that used a different alphabet. I created a word list in Russian using Google Translate, which I had proofed by two native speakers to avoid mismatches in pluralization and capitalization.

To accommodate the non-Latin alphabets, I initially used the *babel* language package. After communication with Boris Veytsman regarding the font options, I switched to the *polyglossia* package.

The second language I explored for this project was Arabic, and here I was faced with handling the right-to-left writing system and letters that have different forms depending on their position within a word. By setting up the preamble to support *arabtex*, the word search puzzle correctly generated an image in right-to-left as intended. Regrettably, due to insufficient skill and knowledge of language packages available at the time that I began this project, I deliberately omitted generating word search puzzles using the Arabic diacritics for short voweling.

As an impractical but interesting exploration, I created a puzzle blending words and filler letters in both Russian and Arabic (Figure 3). The *polyglossia* package handles this with a default main language and additional alphabets used within the document.

Ultimately, I created a title and a word list table within LaTeX as these could be used as worksheets that include a word list, as seen in Figure 4. Improvements to this project that have been considered include word search puzzle generation built internally within LaTeX, which would include automatic sheet generation with all components such as title, word list, and answer key.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ع | ر | ع | ل | К | ت | ت | Т | ش | И |
| И | و | ب | С | Р | О | Я | С | ل | С |
| ر | ب | ن | م | Е | Б | ا | И | ل | ت |
| م | ي | ه | ط | В | Л | س | Ч | ل | ا |
| ل | ا | ر | ر | Е | А | К | Е | Р | Т |
| س | ن | М | س | Т | К | Н | Р | С | م |
| م | ب | ن | ن | К | О | Е | Я | س | م |
| ا | С | م | Ь | А | К | Б | م | ي | غ |
| ء | ر | ه | ن | А | م | О | ي | С | س |
| Ь | Д | Ж | О | Д | ا | ع | ب | ش | ع |

**Figure 3**: Puzzle mixing Russian and Arabic.

ПРИРОДА

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| В | Е | Т | Е | Р | П | Г | Р | Л | Е |
| А | С | Р | Е | К | А | Е | О | Ш | Ы |
| М | Д | Е | Р | Е | В | О | С | Р | Ш |
| О | С | М | Ы | О | Р | П | Г | О | А |
| Б | С | Л | Б | Т | С | Н | Ы | Р | К |
| Л | Л | Е | У | О | Р | Д | А | Б | Н |
| А | Н | Д | Л | Н | З | А | С | Ш | Ы |
| К | В | Н | Л | Е | А | Н | В | М | Р |
| О | Ц | Т | В | Л | Ы | А | Ф | А | Н |
| Е | К | З | Ф | Д | М | Л | И | Б | Т |

| | | |
|---|---|---|
| ветер | гора | дерево |
| дождь | звезды | луна |
| небо | облако | песок |
| река | солнце | трава |

طبيعة

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ت | ؤ | ر | غ | ب | ث | ش | ش | ص | ن |
| ؤ | م | ة | ر | م | ق | ل | ا | ض | ح |
| ل | ز | ة | ة | ف | ر | ن | ح | ي | ر |
| ج | ن | ع | ئ | ئ | س | ء | ا | م | س |
| ب | ه | ص | ش | س | م | ش | ل | ا | ز |
| ل | ر | م | خ | ط | ب | ش | خ | ي | ح |
| ن | ق | ت | ر | ب | م | ى | ب | ض | ق |
| ج | ة | ر | م | ي | غ | ؤ | غ | ء | ئ |
| ي | ى | ا | ئ | ة | خ | ي | و | ة | ي |
| ل | ر | ث | م | و | ج | ن | ل | ا | ؤ |

| | | |
|---|---|---|
| النجوم | القمر | الشمس |
| رمل | خشب | جبل |
| غيم | سماء | ريح |
| نهر | نجيل | مطر |

**Figure 4**: Full puzzles in Russian and Arabic, with titles and word lists.

## 6 Acknowledgments

For support in this project, I acknowledge my father, Dennis Claudio for his early attempts at generating a word search puzzle in LaTeX. I am additionally thankful to Christian Lawson-Perfect for his responses regarding the puzzle generator, to Boris Veytsman for his input on the Cyrillic alphabet for both Russian and Ukrainian, and to Mohamed Mehrez for the instructional tutorial on IguanaTeX as a PowerPoint add-on. I am also grateful to my native and fluent word search reviewers: Rachel Rudyak, Dana Zhytnik, Samiya Ali, AbdulBaqi Matrook, Anita Taft, and Yusuf Alam.

◇ Jennifer Claudio
Santa Clara, California, USA
`claudioj (at) esuhsd dot org`

# On typesetting an English–Japanese book

Antoine Bossard

## Abstract

Improving the English skills of non-native undergraduate students has important implications and can often be directly linked to students' futures, particularly in the IT field. The edition of a bilingual lecture textbook is thus meaningful, notably by considering English as the "major" book language and students' mother tongue the "minor", supporting one. Yet, from a TEXnical point of view, this is far from being trivial. In this article, LATEX methods are given together with guidelines to support the realisation of a bilingual textbook. The especially technically demanding English–Japanese scenario is considered.

## 1 Introduction

Proficiency in English by undergraduate students is, in numerous countries, wishful thinking. Nevertheless, this competence is often a condition for successful graduate studies and thus ought not be overlooked — the earlier the better for students to acknowledge this reality. It is thus pedagogically relevant to provide students with bilingual textbooks.

Herein, we consider the technically demanding English–Japanese scenario, with English being the language of the main text, and Japanese being the "supporting" language (for the sake of generality, we mostly refer to these two languages as the major and minor ones). When it comes to technical considerations for the realisation of such a textbook with LATEX, several challenges arise, starting with the manipulation of distinct writing systems and thus typography.

The objective of this paper is to give LATEX techniques and guidelines to support an edition of such bilingual material while remaining as flexible as possible: the standard `book` class is used and the redefinition of standard commands is avoided to retain as much compatibility as possible.

Beginning with type (textual) considerations, we then address page layout issues before going on with the complex customisation topic, as for page headers, and a PDF-specific discussion. Finally, some statistics are given regarding the usage of such bilingual material as a lecture textbook.

Although bilingual editions are not rare in literature, they are seldom produced for technical texts. For instance in Japanese, authors sometimes rely on marginal notes to provide original English terms when the Japanese transliteration remains unclear [7].

## 2 Type considerations

Especially to facilitate the English and Japanese font manipulation, we rely on a Unicode-capable TEX system. While XƎTEX is an option, we favour LuaTEX for its wider coverage of micro-typographic features with the `microtype` package [10].

The `fontspec` package [9] is used to load the English and Japanese fonts. The language of the main text being English, the `\setmainfont` command sets the English font. For the Japanese font, a new font switch is defined with the `\newfontfamily` command. Because it is often the case that the heights of the glyphs of the fonts do not match, the Japanese font is scaled as follows:

```
\newfontfamily\yu[
    BoldFont = {YuGothB.ttc},
    Scale = MatchLowercase
  ]{YuGothR.ttc}
```

In this example, `\yu` is defined to switch to the Yu Gothic Regular ("游ゴシック Regular") typeface, scaled down to match the main text (i.e., English text) lower case font height. And, the corresponding bold type face is manually set.

Finally, text written in the minor language, here Japanese, is printed in `\small` size, and the `polyglossia` package [6] is used to set the major language, here English, as the default language.

## 3 Page layout issues

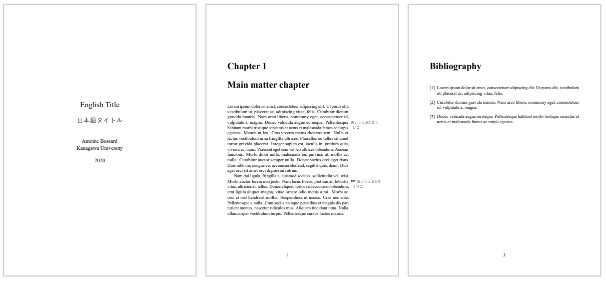
When editing bilingual material, it is likely that a special page layout be required. For instance, in the case of English–Japanese edition, the JIS B5 paper format is a strong candidate. Hence, we rely on the `geometry` package [11]. Furthermore, one key aspect of bilingual editions is the usage of margin notes: it is indeed very valuable to the reader to have margin notes written in the minor language to provide supporting information to the main text written in the major language (see for instance the statistics given in Section 6). A line width of eight Japanese characters in the margin is acceptable. Because several margin notes can be displayed consecutively, it is desirable to use indentation to help the reader split them. Recalling that minor language text is printed in `\small`, we rely on the macro

```
\newcommand\marginnote[1]{\marginpar{%
  \small\hangindent=5pt\hangafter=1 #1}}
```

to insert margin notes. Moreover, when the margin note consists of multiline Japanese text, it can be easily inserted with

```
\marginnote{\yu\cjkscanstart Japanese text\par}
```

doi.org/10.47397/tb/42-2/tb131bossard-engjpn



English Title

日本語タイトル

Antoine Bossard
Kanagawa University

2020

Chapter 1

Main matter chapter

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

あいうえおかきくけこ

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

☞ あいうえおかきくけこ

1

Bibliography

[1] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis.

[2] Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

[3] Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

3

**Figure 1**: Switching between the no-margin and margin layouts: the title page and bibliography are horizontally centred on the page since they need no margin.

where `\yu` is the Japanese font switch as declared above and the macro `\cjkscanstart` (used together with `\par`) has been defined in [4]. This method to typeset a Japanese paragraph is also used in the main text when needed, for instance to include an author biography in both languages.

So, the page layout needs, in addition, to take into account this usage of the margin. And, since header text is hardly avoidable, the page layout needs to be set up accordingly. Such a particular layout can be obtained with the following `geometry` settings:

```
\usepackage[
    paper=b5j, includemp, marginparwidth=70pt
  ]{geometry}
```

With `includemp`, the margin is considered as part of the text body, which is important for headers as detailed in Section 4, and also for horizontal centering of the text on the page if needed. The margin width is adjusted for eight Japanese characters.

Although margin notes are critical in such a scenario, they would feel out of place on certain pages such as the title page. The layout switching feature of `geometry` is used to make the document margin visible or not. Thus, we rely on the

```
\newgeometry{includemp, nomarginpar}
```

command to temporarily disable the margin, as before the title page, and use the `\restoregeometry` command to restore the margin layout.[1]

This is illustrated in Figure 1, with three sample pages: the title page, using the no-margin layout, the first page of a main matter chapter, using the margin layout and showing sample margin notes in the minor language and the first page of the bibliography, using the no-margin layout.

[1] We specify the additional geometry options `vcentering`, `asymmetric` and `hcentering` as well as setting `width` and `textheight`. Since these are design-specific, they are omitted here for the sake of clarity and conciseness.

## 4 Customisation

The customisation of the page headers, chapter pages, table of contents and index are described in this section.

### 4.1 Page headers

We rely on the `titleps` package [1] to customise headers (as opposed to, say, `fancyhdr` [12]) since we use the `titlesec` package [2] for chapter page customisation. The page header includes the margin, so the header rule needs to do the same. We calculate how much the header needs to be extended as follows:

```
\newlength\hdrextension
\setlength\hdrextension{\marginparwidth}
\addtolength\hdrextension{\marginparsep}
```

Then, two layout switching commands are declared accordingly:

```
\newcommand\geonomglayout{% no-margin layout
  \newgeometry{includemp, nomarginpar}
  \widenhead[0pt][0pt]{0pt}{0pt}}% no extension
\newcommand\geomgrestore{% margin layout
  \restoregeometry
  \widenhead[0pt][\hdrextension]
    {0pt}{\hdrextension}}% rule extension
```

Because we use an asymmetric layout as per the `geometry` package definition, the margin is always on the right side of the page. The detailed page style information, including header typesetting, is given in Section 4.2.

### 4.2 Chapter pages and marks

We rely on the `titlesec` package, loaded as follows:

```
\usepackage[pagestyles, extramarks]{titlesec}
```

Chapter titles are given a subtitle in the minor language (i.e., translation) via the command `\mychaptersubtitle`. They are typeset as follows:

```
\newcommand\mychaptersubtitle{}
\titleformat{name=\chapter}[display]
  {\normalfont}
  {\LARGE\sffamily
    \chaptertitlename\ \thechapter}
  {24pt}{\Huge\bfseries}
  [\large\mdseries\mychaptersubtitle]
```

The chapter page style stays simple:

```
\renewpagestyle{plain}[\small]{
  \setfoot{}{\thepage}{}}
```

The page style of the other pages includes a ruled header. Marks are used to retrieve the title in the major language. Headers conventionally alternate between the chapter and section titles. A new page style is declared and set as follows. Note the usage of the commands `\ifthechapter` and `\ifthesection`, provided by `titlesec`, to handle unnumbered chapters and sections.

```
\newpagestyle{hdrgenstyle}[\small]{
  \headrule
  \sethead[\thepage][]% even page left, centre
    [\ifthechapter{\chaptertitlename
      \ \toptitlemarks\thechapter .
      \ \topshortmark\mychaptertitle}
      {\topshortmark\mychaptertitle}]%ev. right
    {\ifthesection{\bottitlemarks\thesection .
      \ \botshortmark\mysectiontitle}
      {\botshortmark\mychaptertitle}}% odd left
    {}{\thepage}}% odd page centre, right
\pagestyle{hdrgenstyle} % general page style
```

A chapter could be created without relying on `titleps`'s extramarks feature simply with

```
\chapter{title\\subtitle}\chaptermark{title}
```

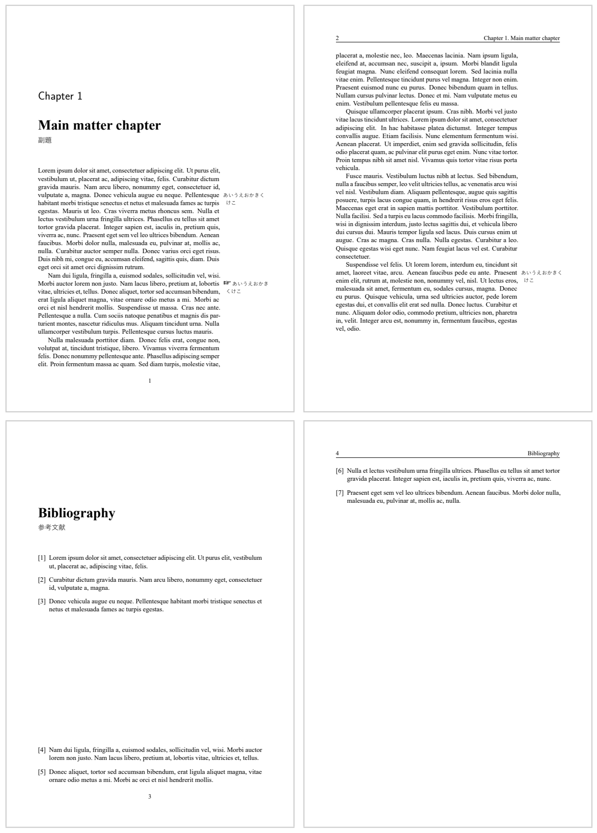
to similarly set the chapter subtitle and the header text in the major language only. However, this simpler method would only work in the case of chapters directly created with the `\chapter` command and thus not with the bibliography, table of contents, and so on. So, it is wiser to customise the chapter page with `\titleformat` whose "after-code" is a command like `\chapsubtitle` which is redefined before `\chapter` for regular chapters and before `\tableofcontents`, `\bibliography`, etc., for such special chapters. Note that this technique is similar to that of `titleps`' extramarks.

Hence, the command `\newchapter` to create a new chapter is defined as follows:

```
\newcommand\mychaptertitle{}
\newshortmark\mychaptertitle
\newcommand\newchapter[2]{
  \renewcommand\mychaptertitle{#1}
  \renewcommand\mychaptersubtitle{#2}
  \chapter{#1}\shortmark\mychaptertitle}
```

its two arguments being the chapter title and subtitle, typically in the major and minor language, respectively. Commands to create sections are defined similarly (`\newsection`, `\mysectiontitle`). Sample output with both a numbered and an unnumbered chapter is given in Figure 2.

Finally, because automatically generated chapters like the table of contents do not call our command `\newchapter` to create the chapter, the subtitle given to the chapter title in the minor language and the extra mark for the header text need to be



Chapter 1

**Main matter chapter**

副題

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. あいうえおかきくけこ

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. ☞ あいうえおかきくけこ

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae,

1

2 Chapter 1. Main matter chapter

placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio. あいうえおかきくけこ

**Bibliography**

参考文献

[1] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis.

[2] Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

[3] Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

[4] Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus.

[5] Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis.

3

4 Bibliography

[6] Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc.

[7] Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla.

**Figure 2**: Ruled header including the margin, chapter marks and chapter page customisation (e.g., subtitle).

additionally specified. Such special chapters are thus created as follows, the table of contents in this example:

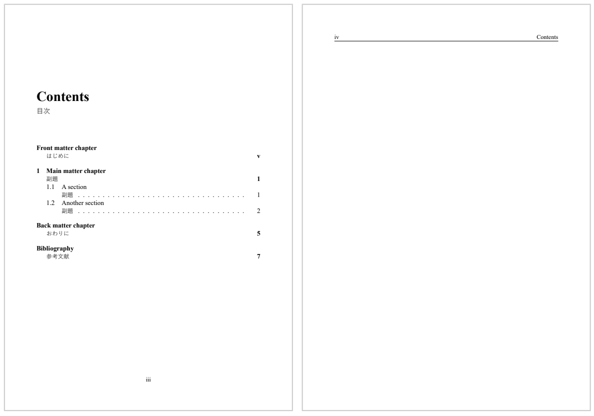
```
\cleardoublepage
\renewcommand\mychaptertitle{\contentsname}
\renewcommand\mychaptersubtitle{{\yu 目次}}
\shortmark\mychaptertitle % set the header mark
\tableofcontents
```

## 4.3 Table of contents and index

It is useful for readers to have TOC entries in both the major and minor languages. Hence, we create each TOC entry from the section (e.g., chapter, section, subsection) title (major language information) and subtitle (minor language information). To this end, the `\chapter` call inside the previously-defined command `\newchapter` is updated as follows:

```
\chapter[#1\\\textmd{#2}]{#1}
  \shortmark\mychaptertitle
```

Again, the same approach is used for lower levels (e.g., sections, subsections). Sample output which illustrates both numbered and unnumbered chapters is given in Figure 3.



Contents

目次

Front matter chapter
はじめに v

1 Main matter chapter
副題 1
1.1 A section
副題 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Another section
副題 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

Back matter chapter
おわりに 5

Bibliography
参考文献 7

iii

iv Contents

**Figure 3**: A sample table of contents with bilingual entries. The effect of header marks can be noticed on the second page.

Regarding the generation of the index, several multilingual index systems exist; since we are considering here the English–Japanese scenario, it may be wise to rely on kameindex (see [3] for more information on kameindex and multilingual indexes in general). kameindex can be combined with imakeidx to obtain, for example, a three-column index:

```
\makeindex[noautomatic, columns=3]
```

Just as with the table of contents, the index chapter title, subtitle and header mark are set before calling `\printindex`.

## 5 PDF specifics

As nowadays electronic publishing often complements traditional, paper-based publishing, it is worth paying some attention to specific features of the former. Although the `hyperref` package can generate PDF bookmarks [8], it is arguably desirable to also have bilingual PDF bookmarks.

The bookmark generated by `hyperref` is labelled with the title specified by the sectioning commands such as `\chapter` and `\section`, with the short title being favoured when set. Hence, there is little to do for the chapters (and other levels) that are created with the `\newchapter` command since a bilingual short title is set for the table of contents as explained previously. However, while an entry of the table of contents relies on the new line command `\\` to separate its title and subtitle, this is not applicable for PDF bookmarks. This can be handled with the conditional command `\texorpdfstring` provided by `hyperref`. The call to `\chapter` in `\newchapter` is refined as follows:

```
\chapter[#1\texorpdfstring{\\\textmd}{ - }{#2}]
  {#1}
```

In this case, the title and subtitle are separated with a dash in the bookmark label and with a line break in the table of contents.

`hyperref` issues a warning when a TeX command — typically a font selection command for the minor language — is included in `#2`, the subtitle. In such a case, `hyperref` (happily) skips the command, so it does not appear in the bookmark label. So, while the warning can be safely ignored, it can also be avoided by using once more the `\texorpdfstring` command inside the subtitle string.

Next, we consider the chapters that are automatically generated and absent from the table of contents (thus not put in the PDF table of contents by `hyperref`), typically the table of contents itself, and the bookmarked pages that do not correspond to a new chapter, such as the title page. These two cases are handled with the appropriate `hyperref` command as exemplified below with a bookmark for the table of contents:

```
\currentpdfbookmark{\contentsname\ - 目次}{lnk1}
```

This is typically called after `\cleardoublepage`.

Next, for the chapters that are manually added to the table of contents — typically the bibliography and index — we combine the `\addcontentsline` and `\texorpdfstring` commands, as exemplified below with a table of contents entry and the corresponding bookmark for the bibliography:

```
\newcommand\myformat[1]{{\mdseries\yu #1}}
\addcontentsline{toc}{chapter}{\bibname
  \texorpdfstring{\\\myformat}{ - }{参考文献}}
```

Here, we have also shown another solution to completely avoid TeX commands inside a PDF string.

Finally, to avoid garbled text in the PDF strings such as bookmark labels, it is wise to set

```
\hypersetup{pdfencoding=auto}
```

in addition to any other `hyperref` options.

## 6 Some statistics

The techniques and guidelines described in this article have been accumulated over the years to typeset the author's own textbook *A Gentle Introduction to Functional Programming in English* [5], whose major language is English and minor one is Japanese.

Those students of the course who used that textbook (not all did) were asked how useful they think the supporting text (e.g., margin notes, section titles translations) written in the minor language is. An online survey was prepared in Japanese — the mother tongue of most students — which consisted of these four statements:

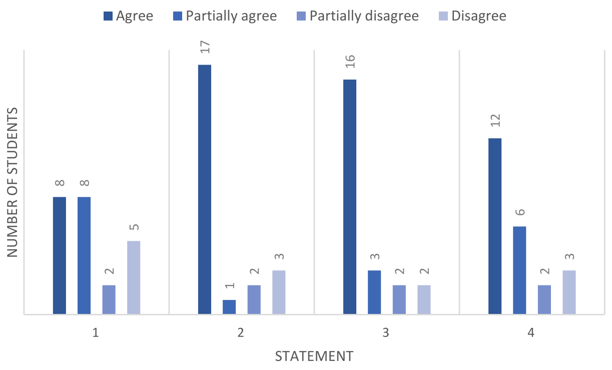


**Figure 4**: Students' answers to the survey regarding the bilingual lecture textbook.

1. Supporting text in Japanese (table of contents, margin notes, index, etc.) is useful.
2. The number of margin notes given in Japanese should be increased.
3. More keywords should be translated in Japanese.
4. More Japanese short commentaries should be given.

The students rated each with "agree", "partially agree", "partially disagree" or "disagree". Twenty-three students submitted the survey whose results are given in Figure 4. The results show the importance and the corresponding student expectations of the supporting text given in the minor language. Note that it is reasonable to assume that the students who disagree at the first question most likely mean "not useful because not enough: I would like more". This is corroborated by the results of the subsequent questions.

## 7 Conclusions

Bilingual publishing has various important applications, for instance educational textbooks. With particular attention to the demanding English–Japanese scenario, we have reviewed herein general LaTeX guidelines and techniques to produce a bilingual book. Considering a major (e.g. English) and a minor (e.g. Japanese) language for the book, issues of the type, page layout and customisation of page headers, chapter pages and special pages have been addressed. For electronic publishing, the adaptation of PDF features, particularly bookmarks, to support bilingual content has been discussed.

We have also obtained readers' feedback: after using such a bilingual textbook (major language: English, minor language: Japanese) for an undergraduate lecture (functional programming), we have asked students to rate this bilingual feature. The results show that students expect a great deal from it, likely because the prevalent mother tongue of the class was the minor language of the book.

## References

[1] J. Bezos. *Headers and footers with titleps*, July 2021. Version 2.14. `ctan.org/pkg/titleps`.

[2] J. Bezos. *The titlesec, titleps and titletoc Packages*, July 2021. Version 2.14. `ctan.org/pkg/titlesec`.

[3] A. Bossard. Experimenting with *makeindex* and Unicode, and deriving *kameindex*. *ArsTeXnica* 26:55–61, 2018. `www.guitex.org/home/it/numero-26-ottobre-2018/351-06-kameindex`.

[4] A. Bossard. A glance at CJK support with XeTeX and LuaTeX. *TUGboat* 40(2):196–201, 2019. `tug.org/TUGboat/tb40-2/tb125bossard-cjk.pdf`.

[5] A. Bossard. *A Gentle Introduction to Functional Programming in English*. Ohmsha, Kanda Jinbocho, Tokyo, Japan, third ed., 2020.

[6] F. Charette, A. Reutenauer, et al. *Polyglossia: Modern multilingual typesetting with XeLaTeX and LuaLaTeX*, Apr. 2021. Version 1.53. `ctan.org/pkg/polyglossia`.

[7] S.L. Harris, D.M. Harris. *ディジタル回路設計とコンピュータアーキテクチャ (Digital Design and Computer Architecture)*. Shoeisha, Funamachi, Shinjuku, Tokyo, Japan, second ed., Sept. 2017. Translated to Japanese by H. Amano, H. Nakajo, M. Suzuki, L. Nagamatsu.

[8] S. Rahtz, H. Oberdiek, The LaTeX3 Project. *Hypertext marks in LaTeX: a manual for hyperref*, June 2021. Version 7.00m. `ctan.org/pkg/hyperref`.

[9] W. Robertson. *The fontspec package — Font selection for XeLaTeX and LuaLaTeX*, Feb. 2021. Version 2.7i. `ctan.org/pkg/fontspec`.

[10] R. Schlicht. *The microtype package*, Mar. 2021. Version 2.8c. `ctan.org/pkg/microtype`.

[11] H. Umeki. *The geometry package*, Jan. 2020. Version 5.9. `ctan.org/pkg/geometry`.

[12] P. van Oostrum. *The fancyhdr and extramarks packages*, Jan. 2021. Version 4.0.1. `ctan.org/pkg/fancyhdr`.

◇ Antoine Bossard
Kanagawa University
Tsuchiya 2946
Hiratsuka, Kanagawa 259-1293
Japan
`abossard (at) kanagawa-u dot ac dot jp`

# New Czechoslovak hyphenation patterns, word lists, and workflow[*]

Petr Sojka, Ondřej Sojka

## Abstract

Space- and time-effective segmentation and hyphenation of natural languages remain at the core of every document preparation system, web browser, or mobile rendering system. We use the unreasonable effectiveness of pattern generation with `patgen`. It is possible to use hyphenation patterns to solve the dictionary problem also for closely related languages, without compromise. In this article, we show how we applied the marvelous effectiveness of `patgen` for the generation of the new Czechoslovak hyphenation patterns that cover both Czech and Slovak languages.

We show that developing universal, up-to-date, high-coverage and highly generalized hyphenation patterns is feasible, generated from semi-automatically prepared word lists from actual language usage. We evaluate the new approach and argue that the new Czechoslovak hyphenation patterns bring significant coverage and generalization improvements, and space savings. We share all the data, word lists, and workflow for reproducibility and usage.

“Any respectable word processing package includes a hyphenation facility. Those based on an algorithm, also called logic systems, often break words incorrectly.” Major Keary in [11]

## 1 Introduction

Space- and time-effective segmentation and hyphenation of natural languages remain at the core of every document preparation system, be it TEX, a modern web browser, or mobile rendering system.

The Unicode Standard supports 5,000 languages that are still in use today. Each of these languages is on the move. A digital typographic system that supports Unicode and its languages in full should support hyphenation in the form of algorithms, rules, or patterns.

However, languages are “moving targets”. Vocabulary changes (e.g., a language adopts new words). Meanings of individual words change in time (e.g., gay in English). The importance of word etymology and segmentation changes as well. The word `roz-um` (understanding) hyphenated in 1956 [7] according to prefix `roz`, signaling separation, and suffix `um`, signaling knowledge is now perceived as a single stem `rozum` (intelligence, mind). Thus also word hyphenation algorithms should adapt accordingly from time to time to match language usage.

There are essentially two quite different approaches to hyphenation:

**etymology-based** The rule is to cut a word on the border of a compound word or the boundary of stem and affix, prefix, or negation. A typical example is the British hyphenation rules from the Oxford University Press [1].

**phonology-based** Hyphenation follows the pronunciation of syllables, allowing for much more fluent reading. Syllabification is not followed near word borders (in the same languages) — hyphenation is forbidden when close to word borders. American publishers [6] and the *Chicago Manual of Style* [4] users prefer this pragmatic approach.

There is a trade-off between the two: one prefers visual highlighting of the word meaning etymology as British do, or likes phonology — convenient reading across the lines.

There is high diversity among languages, but what is the same is that the meaning is conveyed by syllables of the language [15]. There is also high diversity among languages’ spelling, but what is the same is that the mapping from phonology to spelling is almost lossless. And there is high diversity in the language hyphenation rules, but when phonology-based hyphenation is preferred, the syllable definition based on consonant and vowel segments is the same for all languages, providing a chance to develop one universal syllable-based segmentation algorithm.

Czech and Slovak are very close languages. Citizens of Czechoslovakia understood both before the states split in 1993. The syllabification and pronunciation rules are the same. We spotted a clear trend towards phonology-based hyphenation. The differences in spelling are rule-based. These observations led us to the idea of common Czechoslovak hyphenation patterns usable for both languages.

This paper evaluates the feasibility of the development of *universal* phonology-based (syllabic) hyphenation patterns. As a case study, we describe the development of Czechoslovak hyphenation patterns from word lists of Czech [20, 21, 28] and Slovak [23]. We generated new patterns from word lists captured from actual language use during the last decade. We rigorously evaluated new patterns as superior to the current specific Czech and Slovak patterns. We document our reproducible workflow and all resources in a public repository. We conclude by outlining further possible hyphenation pattern developments to meet today’s demands.

[*] This is a significantly updated and enriched version of the paper published in CSTUG’s *Zpravodaj* [27].

"Hyphenation does not lend itself to any set of unequivocal rules. Indeed, the many exceptions and disagreements suggest it is all something dreamed up at an anarchists' convention." Major Keary in [11]

## 2 Syllable segmentation methods

The core idea is to develop shared hyphenation patterns for phonology-based languages. If these languages share pronunciation rules, homographs from different languages typically do not cause problems, as they are hyphenated the same [3, 7, 9, 31]. There are occasional cases where the break in a compound word dictates a hyphenation point contrary to phonology (`roz-um` vs. `ro-zum`). These could be solved by not allowing the hyphenation of this particular word around this specific break.

Marchand et al. [16] showed that data-driven approaches to syllabification algorithms outperform rule-based ones, reaching accuracy levels around 95% per single language. Bartlett et al. [2] developed a machine learning approach for automatic syllabification, motivated by the needs of letter-to-phoneme conversion. Trogkanis et al. [29] used conditional random fields for word hyphenation and compared the accuracy and other metrics with the original technique of Liang [14]. Their results abstracted heuristics to optimize generated patterns by `patgen` [8], diminishing achievable performance by Liang's technique. A recent study on syllabification [13] shows that even in comparison with the latest "deep" neural approaches, fine-tuned `patgen` performance beats them in both accuracy and performance.

Recently, there have been attempts to tackle the word segmentation problem in different languages by Shao et al. [18]. The algorithm is error-prone, but it was developed primarily for speech recognition and language representation tasks. Due to the nonzero error rate, its applicability to the hyphenation task is limited. In a typesetting system, the hyphenation algorithm must cover all exceptions and not tolerate any errors.

We recently showed that the `patgen` approach of pattern generation from word list is unreasonably effective [26]. One can set the parameters of the generation process so that the patterns cover 100% of hyphenation points, and their size remains reasonably tiny. We compressed the word list with 3,000,000 hyphenated words into 30,000 bytes of the packed trie data structure for the Czech language. That means achieving a compression ratio of several orders of magnitude with 100% coverage and nearly zero errors [26]. For a similar language such as Slovak, the pronunciation is very similar, syllable-forming principles are the same, and compositional rules and prefixes are pretty close, if not identical.

We have decided to verify the approach by developing hyphenation patterns that will hyphenate *both* Czech and Slovak words without errors, with only a few missed hyphens. The missed hyphen will appear *only* in words like `oblít` where *meaning* of the term is needed for the decision: `o-blít` or `ob-lít`.

The clear trend, at least in the Czech hyphenation codification books from Haller [7] via [28] used so far in TeX and Word [3], to currently-maintained word lists in [9], reflect gradual movement from etymology to phonology for better syllabic pronunciation when reading hyphenated words. The context-dependent hyphenation decision to resolve such preferences and meaning ambiguities are needed only sporadically.

We needed to create lists of correctly hyphenated Czech and Slovak words to generate these hyphenation patterns.

## 3 Data preparation

For our work, Lexical Computing CZ donated word lists with frequencies for Czech and Slovak from the TenTen family of corpora [10, 12]. These corpora were drawn from the Internet within the last decade. They contain words used in both languages.

The Czech word list was cleaned up and extended as described by us [24, 25, 26], using the Czech morphological analyzer `majka`. Contrary to the German database, we opted for the simplest format possible, allowing easy enrichment and editing of word lists.

For the generalization of hyphenation rules by `patgen`, we do not need the word list to be as complete as possible, so we used only those words that appeared more than ten times. The final word list file `cs-all-cstenten.wls` contained 606,494 words.

For Slovak, we obtained 1,048,860 Slovak words with a frequency higher than ten from 2011 SkTenTen corpora [10]. We only used words with a frequency higher than thirty comprised of only ISO Latin 2 characters, obtaining a file `sktenten.wls` with 544,609 words.

By joining both language files, we got 967,058 Czech and Slovak words in `cssk-all-join.wls`, of which 106,016 were contained in the intersection of both word lists: `cssk-all-intersect.wls`.

## 4 Pattern development

Figure 1 illustrates the workflow of the Czechoslovak pattern development. We have used recent, accurate Czech patterns [26] for the hyphenation of the joint Czech and Slovak word list. We had to fix incorrect hyphenation points manually, typically near the prefix and stem boundary when phoneme-based hyphenation point was one character away from the seam of the prefix or compound word: `neja-traktivnější`, `neja-teističtější`, `neje-kologičtější`.

We then hyphenated words used in both languages also by the current Slovak patterns. There were only a few word hyphenations that needed to be corrected — we created the file `sk-corrections.wlh` that contained the fixed hyphenated words. Finally, we used them as input to `patgen` with a higher weight during the generation of the final Czechoslovak hyphenated patterns.

We did not pursue 100% coverage at all costs because the source data is noisy, and we do not want the patterns to learn all the typos and inconsistencies. We expand on this in the Jupyter notebook [19]. Gentle readers may also find the scripts used there.

## 5 Evaluation

We evaluated the quality of developed patterns by two metrics. *Coverage* of hyphenation points in the training word list tells how the patterns correctly predicted hyphenation points used in training. *Generalization* means how the patterns behave on unseen data, on words not available in the data used during `patgen` training.

We see the coverage and generalization as a *classification* task, i.e., how the patterns classify hyphenation points in the training and testing word lists, respectively.

### 5.1 Classification

To evaluate the classification results, there are four numbers in the contingency matrix that compare hyphenation point prediction by patterns with the ground truth expressed in the wordlist: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In tables 1–4 on page 156, we report:

**Good** sum or percentage of found hyphenation points (TP),

**Bad** sum or percentage of badly suggested hyphenation points (FP, type 1 error),

**Missed** sum or percentage of missed hyphenation points (FN, type 2 error).

Type 1 errors are more severe than type 2 errors in our hyphenation points setup. Nonzero **bad** results do not necessarily mean that the patterns performed poorly. Just the opposite holds — the patterns have found a rule that the ground truth wordlist does not obey. In other words, the inconsistency needs fixing in the underlying word list rather than emitting the pattern for a valid exception. We practiced manual inspection of bad hyphenation points during the development of the word list.

### 5.2 Generalization

We used tenfold cross-validation to assess the generalization properties, that is, leaving out one-tenth of the training set to evaluate the patterns' effectiveness on unseen words. We show the results in Table 5. The evaluation metrics differ slightly with different `patgen` parameters, with the best results achieved when we maximize the coverage of the training set.

The achieved results show that both evaluation metrics are close to perfection. We can either opt for perfect coverage and reach it or push to maximize generalization qualities and performance on unseen words. In the first case, we achieve essentially lossless compression of wordlist hyphenation points by the developed pattern. In the second, we miss only less than 1% of valid hyphenation points. Achieving that for two languages in parallel seems like a good result. It is feasible to continue merging additional word lists to develop generic patterns for syllabically hyphenated languages.

We do not know pattern performance for most of the other available patterns as there are no word lists to use for the evaluation and comparison.

> "Esoteric Nonsense? Hyphenation is neither anarchy nor the sole province of pedants and pedagogues.... Used in moderation, it can make a printed page more visually pleasing. If used indiscriminately, it can have the opposite effect, either putting the reader off or causing unnecessary distraction. If the intended audience is bound to read the work (a user manual, for example), poor hyphenation practice may not matter. If the author wants to attract and hold an audience, then hyphenation needs just as careful attention as any other aspect of presentation." Major Keary in [11]

## 6 Conclusion and summary

We have shown that the development of common hyphenation patterns for several languages with similar pronunciations is feasible. `Patgen` was able to generalize hyphenation rules for both languages with only a negligible increase in the size of the generated patterns.

The resulting Czechoslovak patterns hyphenate Czech and Slovak *much better* than the former single-language patterns, with much higher coverage, zero error rate, and evaluated generalization. The whole process is reproducible, is documented, and available as a Jupyter demo notebook with source code [19].

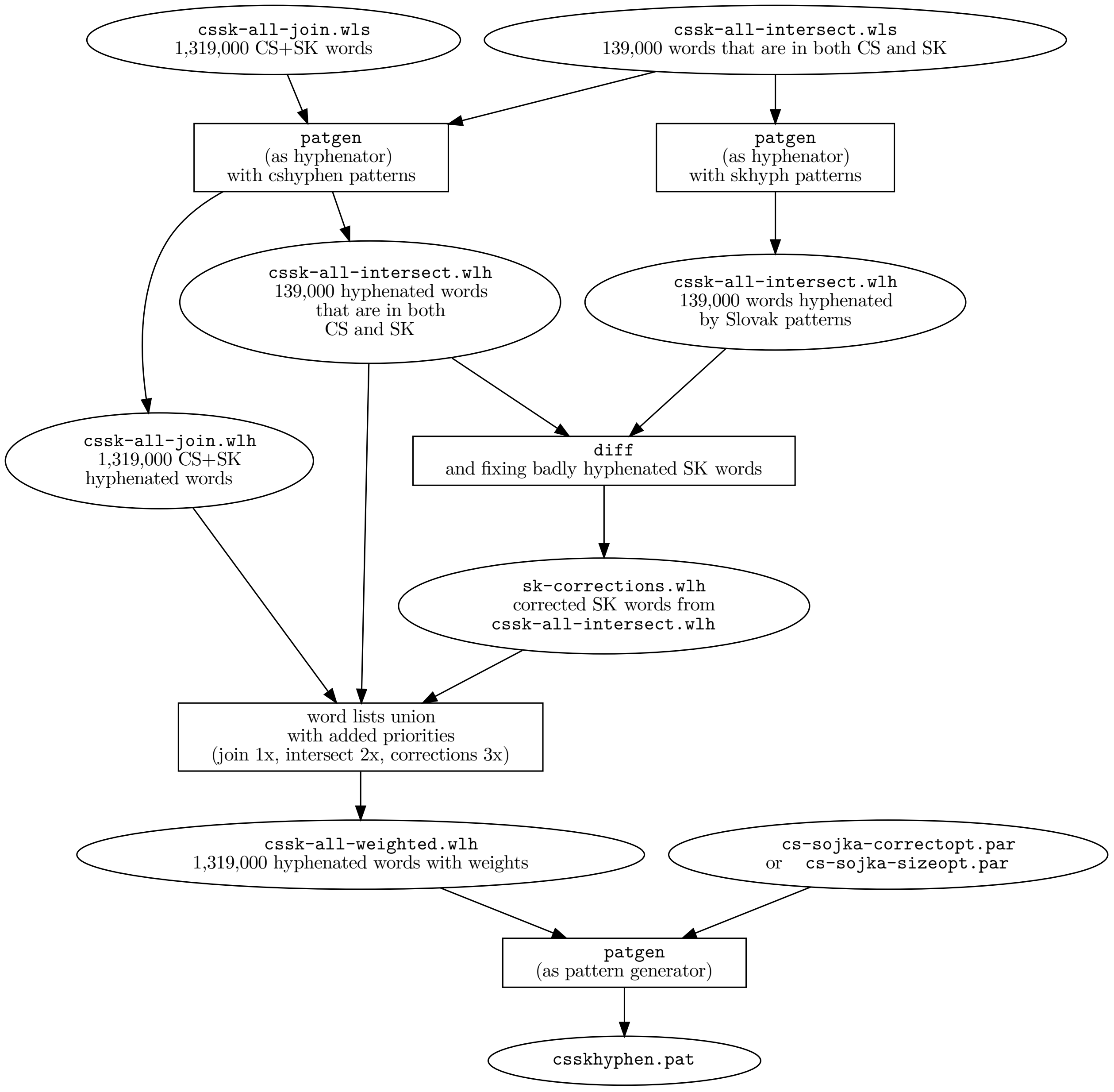


**Figure 1**: The whole pattern development workflow is showed above from top:
a) Czech and Slovak word lists collection, [26] and intersection;
b) bootstrapping hyphenated word lists with syllabic Czech patterns;
c) checking and fixing by deploying the rarely-used `patgen` weighting
for Slovak words common with Czech ones;
d) generation of final patterns.

The whole workflow and scripts are available in the public repository [19].

**Table 1**: Statistics from the generation of Czechoslovak hyphenation patterns with *custom* parameters.

| Level | Patterns | Good | Bad | Missed | Lengths | Params |
|---|---|---|---|---|---|---|
| 1 | 830 | 2,819,833 | 470,649 | 35,908 | 1 3 | 1 3 12 |
| 2 | 1,590 | 2,748,581 | 3,207 | 107,160 | 2 4 | 1 1 5 |
| 3 | 2,766 | 2,852,334 | 12,197 | 3,407 | 3 6 | 1 2 4 |
| 4 | 1,285 | 2,851,931 | 986 | 3,810 | 3 7 | 1 4 2 |

**Table 2**: Statistics from the generation of Czechoslovak hyphenation patterns with *correct optimized* parameters.

| Level | Patterns | Good | Bad | Missed | Lengths | Params |
|---|---|---|---|---|---|---|
| 1 | 2,032 | 2,800,136 | 242,962 | 55,605 | 1 3 | 1 5 1 |
| 2 | 2,009 | 2,791,326 | 10,343 | 64,415 | 1 3 | 1 5 1 |
| 3 | 3,704 | 2,855,554 | 11,970 | 187 | 2 6 | 1 3 1 |
| 4 | 1,206 | 2,854,794 | 33 | 947 | 2 7 | 1 3 1 |

**Table 3**: Statistics from the generation of Czechoslovak hyphenation patterns with *size optimized* parameters.

| Level | Patterns | Good | Bad | Missed | Lengths | Params |
|---|---|---|---|---|---|---|
| 1 | 419 | 2,833,402 | 667,031 | 22,339 | 1 3 | 1 2 20 |
| 2 | 1,506 | 2,430,120 | 1,188 | 425,621 | 2 4 | 2 1 8 |
| 3 | 3,579 | 2,846,112 | 15,881 | 9,629 | 3 5 | 1 4 7 |
| 4 | 2,401 | 2,843,657 | 4 | 12,084 | 4 7 | 3 2 1 |

**Table 4**: Comparison of the efficiency of different approaches to hyphenating Czech and Slovak. Note that the Czechoslovak patterns are comparable in size and quality to single-language ones — there is only a negligible difference compared to, e.g., purely Czech patterns.

| Word list | Parameters | Good | Bad | Missed | Size | Patterns |
|---|---|---|---|---|---|---|
| Slovak | [5, by hand] | N/A | N/A | N/A | 20 kB | 2,467 |
| Czech | correctopt [26] | 99.76% | 2.94% | 0.24% | 30 kB | 5,593 |
| Czech | sizeopt [26] | 98.95% | 2.80% | 1.05% | 19 kB | 3,816 |
| Slovak | [22, Table 1] | 99.94% | 0.01% | 0.06% | 56 kB | 2,347 |
| Czechoslovak | sizeopt | 99.67% | 0.00% | 0.33% | 40 kB | 7,417 |
| Czechoslovak | correctopt | 99.99% | 0.00% | 0.01% | 45 kB | 8,231 |
| Czechoslovak | custom | 99.87% | 0.03% | 0.13% | 32 kB | 5,907 |

**Table 5**: Results of 10-fold cross-validation with evaluated parameters shows very good generalization properties (learning on 90%, and testing on remaining 10%)

| Parameters | Good | Bad | Missed |
|---|---|---|---|
| correctopt | 99.81% | 0.15% | 0.04% |
| custom | 99.64% | 0.22% | 0.14% |
| sizeopt | 99.41% | 0.18% | 0.40% |

## Dissemination

Current hyphenation support based on hyphenation patterns is collected in the `hyph-utf8` [17] project. The project uses ISO standards, notably Unicode and IETF language tags BCP 47. BCP 47 defines a `Scope` property to identify subtags for language collections. `hyph-utf8` currently contains hyphenation patterns for 65 different languages with an additional 9 dialect or transliteration variants.

Our new patterns for "the Czechoslovak language" were accepted for inclusion to the `hyph-utf8` repository [17], and will be supported in the next revisions of `hyph-utf8` and `polyglossia` in the TEX Live distribution. LuaTEX allows loading patterns at runtime, for only the languages used in a document. For other engines, *all* the patterns have to be loaded in precomputed, compact form into TEX's memory from the format file at the start of every document compilation.

As suggested by TEX experts, we prefer Czech and Slovak `\language`s being internally synonyms, with patterns only loaded once.

Using the patterns via available libraries in many programming languages (JavaScript, Perl, Python, C, and more) is straightforward and makes the patterns' usage versatile. Most typesetting systems and browsers, including OpenOffice and Chrome, could hyphenate in narrow columns of mobile devices. Most of them, if not all systems, use pattern technology and practices from the TEX community anyway.

We will support pattern dissemination in TEX distributions and multilingual support packages. We will tidy up available language resources with the community of Czech and Slovak users.

## Future work

We think of developing language-agnostic patterns for syllabically hyphenated languages, based on available data from CELEX [13] with our workflow and evaluation measures. Wordpiece segmentation algorithm [30] gives superb results in the NLP domain for language translation, indicating that information is conveyed via character *n*-grams. With universal, syllable-based patterns, it will be possible to hyphenate text for most syllabically hyphenated languages even without knowing the language markup.

Another direction of research attention will be machine-learned heuristics for setting of `patgen` generation parameters, with the objective of metrics optimization used in the evaluation. When applied to the languages with available word lists, it would lead to pattern improvements for most supported languages.

## Acknowledgment

We are indebted to Don Knuth for questioning the common properties of Czech and Slovak hyphenation during our presentation of [26] at TUG 2019, which has led us in this research direction. We also thank everyone on whose shoulders we build our work, and to all who commented on our workflow, patterns, and this paper, and discussed it at TUG 2021.

## References

[1] R.E. Allen, ed. *The Oxford Spelling Dictionary*, vol. II of *The Oxford Library of English Usage*. Oxford University Press, 1990.

[2] S. Bartlett, G. Kondrak, C. Cherry. Automatic Syllabification with Structured SVMs for Letter-to-Phoneme Conversion. In *Proceedings of ACL-08: HLT*, pp. 568–576, Columbus, Ohio, June 2008. ACL. `https://www.aclweb.org/anthology/P08-1065`

[3] A. Bauer. *Dělení slov / slovotvorba v praxi /* [Word hyphenation / practical morphology / ]. Nakladatelství Olomouc, Olomouc, 1997.

[4] *The Chicago Manual of Style*. University of Chicago Press, Chicago, 17th ed., Sept. 2017.

[5] J. Chlebíková. Ako rozděliť (slovo) Československo [How to hyphenate the word Czechoslovakia]. *Zpravodaj CSTUG* 1(4):10–13, Apr. 1991. `10.5300/1991-4/10`

[6] P.B. Gove, M. Webster. *Webster's Third New International Dictionary of the English language Unabridged*. Merriam-Webster Inc., Springfield, Massachusetts, U.S.A, Jan. 2002.

[7] J. Haller. *Jak se dělí slova* [How Words Get Hyphenated]. Státní pedagogické nakladatelství Praha, 1956.

[8] Y. Haralambous. A Revisited Small Tutorial on Patgen, 28 Years After, Mar. 2021. `https://ctan.org/pkg/patgen2-tutorial`

[9] Internetová jazyková příručka [Internet Language Reference Book]. `https://prirucka.ujc.cas.cz/?id=135`

[10] M. Jakubíček, A. Kilgarriff, et al. The TenTen Corpus Family. In *Proc. of the 7th International Corpus Linguistics Conference (CL)*, pp. 125–127, Lancaster, UK, July 2013.

[11] M. Keary. On hyphenation—anarchy of pedantry. *PC Update, The magazine of the Melbourne PC User Group*, 2005. `https://web.archive.org/web/20050310054738/http://www.melbpc.org.au/pcupdate/9100/9112article4.htm`

[12] A. Kilgarriff, P. Rychlý, et al. The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pp. 105–116, Lorient, France, 2004.

[13] J. Krantz, M. Dulin, P.D. Palma. Language-Agnostic Syllabification with Neural Sequence Labeling. In *18th IEEE International Conference On Machine Learning And Applications, ICMLA 2019, Boca Raton, FL, USA, December 16–19, 2019*, M.A. Wani, T.M. Khoshgoftaar, et al., eds., pp. 804–810. IEEE, 2019. `10.1109/ICMLA.2019.00141`

[14] F.M. Liang. *Word Hy-phen-a-tion by Com-put-er.* Ph.D. thesis, Department of Computer Science, Stanford University, Aug. 1983. `https://tug.org/docs/liang/liang-thesis.pdf`

[15] I. Maddieson. Syllable Structure. In *The World Atlas of Language Structures Online*, M.S. Dryer, M. Haspelmath, eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013. `https://wals.info/chapter/12`

[16] Y. Marchand, C.R. Adsett, R.I. Damper. Automatic Syllabification in English: A Comparison of Different Algorithms. *Language and Speech* 52(1):1–27, 2009. `10.1177/0023830908099881`

[17] A. Rosendahl, M. Miklavec. TEX hyphenation patterns. `http://hyphenation.org/tex`

[18] Y. Shao, C. Hardmeier, J. Nivre. Universal Word Segmentation: Implementation and Interpretation. *Transactions of the Association for Computational Linguistics* 6:421–435, 2018. `10.1162/tacl_a_00033`

[19] O. Sojka, P. Sojka. cshyphen repository. `https://github.com/tensojka/cshyphen`

[20] P. Sojka. Notes on Compound Word Hyphenation in TEX. *TUGboat* 16(3):290–297, 1995. `https://tug.org/TUGboat/tb16-3/tb48soj2.pdf`

[21] P. Sojka. Hyphenation on Demand. *TUGboat* 20(3):241–247, 1999. `https://tug.org/TUGboat/tb20-3/tb64sojka.pdf`

[22] P. Sojka. Slovenské vzory dělení: čas pro změnu? In *Proc. of SLT 2004, 4th seminar on Linux and TEX*, pp. 67–72, Znojmo, 2004. Konvoj. `https://fi.muni.cz/usr/sojka/papers/skhyp.pdf`

[23] P. Sojka. Slovenské vzory dělení: čas pro změnu? [Slovak Hyphenation Patterns: A Time for Change?]. *CSTUG Bulletin* 14(3–4):183–189, 2004. `10.5300/2004-3-4/183`

[24] P. Sojka, O. Sojka. The Unreasonable Effectiveness of Pattern Generation. *Zpravodaj CSTUG* 29(1–4):73–86, 2019. `10.5300/2019-1-4/73`

[25] P. Sojka, O. Sojka. Towards Universal Hyphenation Patterns. In *Proc. of Recent Advances in Slavonic Natural Language Processing—RASLAN 2019*, A. Horák, P. Rychlý, A. Rambousek, eds., pp. 63–68, Karlova Studánka, Czech Republic, 2019. Tribun EU. `https://nlp.fi.muni.cz/raslan/2019/paper13-sojka.pdf`

[26] P. Sojka, O. Sojka. The unreasonable effectiveness of pattern generation. *TUGboat* 40(2):187–193, 2019. `https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf`

[27] P. Sojka, O. Sojka. Towards New Czechoslovak Hyphenation Patterns. *Zpravodaj CSTUG* 30(3–4):118–126, 2020. `https://cstug.cz/bulletin/pdf/2020-3-4.pdf#page=16`

[28] P. Sojka, P. Ševeček. Hyphenation in TEX — Quo Vadis? *TUGboat* 16(3):280–289, 1995. `https://tug.org/TUGboat/tb16-3/tb48soj1.pdf`

[29] N. Trogkanis, C. Elkan. Conditional Random Fields for Word Hyphenation. In *Proc. of the 48th Annual Meeting of the ACL*, pp. 366–374, Uppsala, Sweden, July 2010. ACL. `https://www.aclweb.org/anthology/P10-1038`

[30] Y. Wu, M. Schuster, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016. `https://paperswithcode.com/method/wordpiece`

[31] Ľ. Štúr Institute of Linguistics of the Slovak Academy of Sciences (SAS), ed. *Pravidlá slovenského pravopisu* [Rules of Slovak Grammar]. Veda, publisher of SAS, Bratislava, 3rd (updated printing) ed., 2000. `https://www.juls.savba.sk/ediela/psp2000/psp.pdf`

◇ Petr Sojka
Faculty of Informatics, Masaryk Univ., Brno, Czech Republic
`sojka (at) fi dot muni dot cz`
`https://www.fi.muni.cz/usr/sojka/`
ORCID 0000-0002-5768-4007

◇ Ondřej Sojka
Faculty of Informatics, Masaryk Univ., Brno, Czech Republic
`454904 (at) mail dot muni dot cz`
ORCID 0000-0003-2048-9977

# Automatically generate personalized tasks and sample solutions for the fundamentals of electrical engineering with PGFPlots and CircuiTi*k*Z

Mathias Magdowski

**Abstract**

All students in our class "Fundamentals of Electrical Engineering" get a personalized task via e-mail, solve it and submit a digitized version of their handwritten solution. Then, students peer review each other (lowering the review effort for the lecturer) with the help of a similarly personalized sample solution.

The procedure is automated via MATLAB and therefore scalable for large numbers of students. In contrast to multiple choice questions or pure numerical solutions, here the approach and the detailed calculations can also be assessed. This article describes generating the tasks and sample solutions in LaTeX with the help of PGFPlots and CircuiTi*k*Z.

## 1 Motivation

Just like riding a bicycle or playing the piano, you learn the basics of electrical engineering not by watching and listening, but by participating, trying and practicing. In order to prepare our students as well as possible for the corresponding examination after two semesters, we have been offering personalized tasks for handwritten solutions at the Otto von Guericke University in Magdeburg for several years. Such a solution is exam-related and, compared to "online" input in text fields or formula editors, also allows the use of sketches, diagrams and circuit diagrams in the presentation of the approach and calculation path. Personalization of the tasks reduces the danger of copying from an external solution. Nevertheless, the students can, may and should help and advise each other in solving the tasks.

To keep the correction effort for the teachers reasonable, students then correct each other in peer review. This also requires personalized sample solutions, which must be so good and detailed that even someone who has not been able to solve the task correctly can still correct it correctly. Thinking into a foreign solution path also promotes a deeper understanding of the topic of the task. As an extrinsic motivation and incentive for solving the tasks and for mutual assessment, students receive additional points for admission to the exam.

## 2 Procedure

To prevent the procedure from degenerating into "red tape" for larger groups of students and to ensure that it can be easily automated, the generation of tasks and sample solutions, the submission of solutions, the mutual correction and the allocation of additional points is carried out completely digitally; see Figure 1. The students register in our learning management system Moodle (recognizable by the symbol ). The list of students is fed into a MATLAB program, which automatically generates all personalized tasks and sample solutions using LaTeX. The tasks are then sent to the students by e-mail; the sample solutions are saved into a local folder first.

Next, the students work out their solutions within one week and submit them back to Moodle. Then all solutions can be downloaded in a `zip` file. Another MATLAB program sends all students two foreign solutions as well as the appropriate sample solutions by e-mail for peer review.

The students then must submit their corrections within another week. Again, all corrections and the additional points can be downloaded collectively. A last MATLAB program sends all students their two corrected solutions as well as their own sample solution and notes the achieved points in a list.

MATLAB as a programming environment was chosen by the author because it is quite common in electrical engineering. However, there is already a similar implementation in Python by Olivier Cleynen, who was also working in Magdeburg [1].

## 3 Implementation in LaTeX

The choice of LaTeX as a document description language was of course not solely made for this special kind of task. Rather, the complete exercise book and the corresponding sample solutions, the catalog of exam questions and the lecture script as a book [3], were already typeset in LaTeX. Especially advantageous are the excellent mathematical notation via the `amsmath` package, the perfect set of physical units via `siunitx`, and the possibility to use PGFPlots or CircuiTi*k*Z based on the Ti*k*Z package to create diagrams and circuit diagrams directly in the LaTeX source code.

When repeatedly creating new exam tasks, the author finally came up with the idea of not changing the numerical values in a diagram by hand, but to use a random number generator for this purpose. If the (pseudo-)random number generator is then started using a personal but known number, e. g. the student number, another number creates a different diagram, but the same number creates the same diagram again. This way, tasks and solutions can be generated multiple times, if something does not work correctly or if small changes have to be made in the text.

doi.org/10.47397/tb/42-2/tb131magdowski-autoee

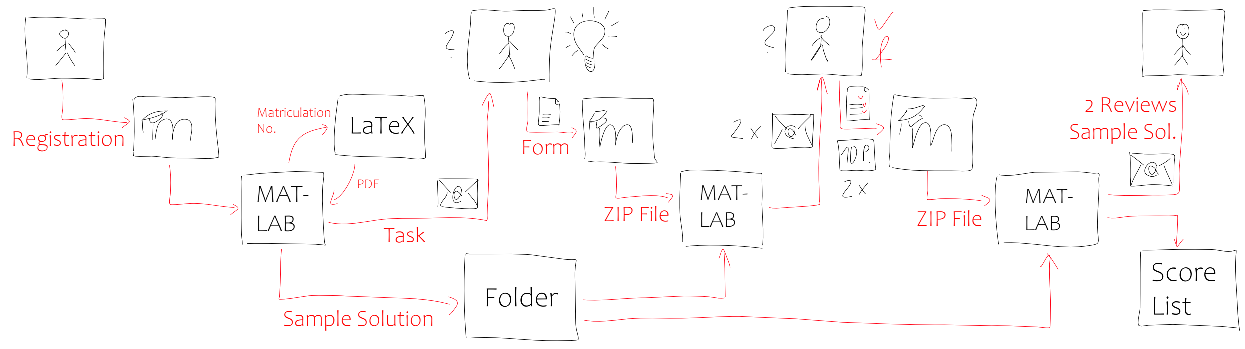


**Figure 1**: Procedure for the realization of personalizable tasks with anonymous peer review

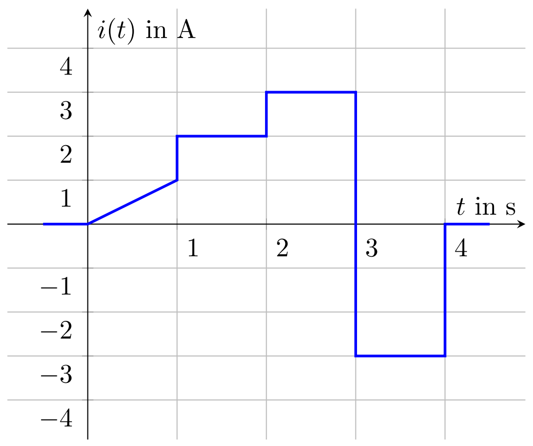


**Figure 2**: Randomized generated plot in PGFPlots; output of Listing 1

Current in the 1. section (1 point):

$$i(t) = 1\,\mathrm{A\,s^{-1}} \cdot t \tag{1}$$

Charge in the 1. section (1 point):

$$Q(t) = \int\limits_0^t 1\,\mathrm{A\,s^{-1}} t'\,\mathrm{d}t' + 0 \tag{2a}$$

$$= 1\,\mathrm{A\,s^{-1}} \cdot \left[\frac{t'^2}{2}\right]_0^t \tag{2b}$$

$$= 0.5\,\mathrm{A\,s^{-1}} t^2 \tag{2c}$$

Charge at the end of the 1. section (1 point):

$$Q(1\,\mathrm{s}) = 0.5\,\mathrm{A\,s} \tag{3}$$

**Figure 3**: Algorithmically generated sample solution to the diagram in Figure 2; output of Listing 2

### 3.1 Random number generation and plot creation directly in LaTeX

A typical task (the same for all students) is the calculation of the time-dependent charge from a graphically given current curve by integrating it section by section over time. All students then receive a personalized time diagram of the current. Listing 1 shows example source code for the generation of the random numbers and the generation of this diagram with the PGFPlots package based on Ti*k*Z. (All listings have been edited slightly for presentation.)

The corresponding output is shown in Figure 2. Each diagram always contains a linear increase (or decrease) of the current in the first time period as well as three further time periods in which the current remains constant (and can also be zero). The greatest difficulty in creating these plots is to use case distinctions to ensure that the current is not the same in two successive time periods. In this case, the number of time segments and thus the computational effort for the students would be reduced.

The LaTeX source for generating the corresponding sample solution is shown in Listing 2. For brevity, it has been shortened to the first time period. The corresponding output is shown in Figure 3. The actual version for students is much more detailed and also deals with typical errors.

One can note that the solution is the same for all students and differs only in the numerical values. This is indeed the case. Nevertheless, the difficulty of this introductory task should not be underestimated, especially for students in the first weeks of the semester when this task is given.

In the case of the independent, handwritten solution, there are many things to pay attention to, in addition to the actual arithmetic solution. These include the reading of the current function from the diagram, the correct mathematical notation, the correct calculation of the units and the drawing of the time-dependent charge. Students must also learn to break down complex tasks into simpler subtasks. The same scheme for generating the diagrams and the appropriate sample solution is also used in a

**Listing 1**: LaTeX source code to create randomized PGFPlots diagrams; output is shown in Figure 2

```
\documentclass{standalone}
\usepackage{pgfplots,siunitx}
\begin{document}
% Set random number generator to matriculation number
\pgfmathsetseed{123456}
% Current at time 1 s (in A, can also still be zero, but should not be)
\pgfmathrandominteger{\stromeinsrandom}{-4}{4}
% if current is zero, set to 1 A
\pgfmathsetmacro{\stromeins}{ifthenelse(\stromeinsrandom==0,1,\stromeinsrandom)}
% Current in the period from 1 s to 2 s (in A, can also be zero)
\pgfmathrandominteger{\stromzwei}{-4}{4}
% Current in the period from 2 s to 3 s (in A, can also be zero)
\pgfmathrandominteger{\stromdreirandom}{-4}{4}
% if the current is equal to the value from the previous period, invert the sign
\pgfmathsetmacro{\stromdrei}{ifthenelse(\stromzwei==\stromdreirandom,
                                        -\stromdreirandom,\stromdreirandom)}
% if both currents are zero, set new current to 1 A
\pgfmathsetmacro{\stromdrei}{ifthenelse(abs(\stromzwei)+abs(\stromdrei)==0,1,\stromdrei)}
% Current in the period from 3 s to 4 s (in A, can also be zero)
\pgfmathrandominteger{\stromvierrandom}{-4}{4}
% if both currents are zero, set new current to 1 A
\pgfmathsetmacro{\stromvier}{ifthenelse(abs(\stromzwei)+abs(\stromvierrandom)==0,
                                        1,\stromvierrandom)}
% if both currents are zero, set new current to 1 A
\pgfmathsetmacro{\stromvier}{ifthenelse(abs(\stromdrei)+abs(\stromvier)==0,1,\stromvier)}
% if the current is equal to the value from the previous period, invert the sign
\pgfmathsetmacro{\stromvier}{ifthenelse(\stromdrei==\stromvier,-\stromvier,\stromvier)}
\begin{tikzpicture}
        \begin{axis}[
                xlabel={$t$ in \si{\second}}, ylabel={$i(t)$ in \si{\ampere}},
                xmin=-0.9,xmax=4.9,ymin=-4.9,ymax=4.9,
                xtick={1,2,3,4},ytick={-4,-3,-2,-1,1,2,3,4},
                xticklabel style={below right},
                yticklabel style={below left},
                axis x line=middle,axis y line=center,
                xmajorgrids,ymajorgrids,]
                \addplot+[mark=none,line width=1pt] coordinates {
                        (-0.5,0)
                        (0,0)
                        (1,\stromeins) (1,\stromzwei)
                        (2,\stromzwei) (2,\stromdrei)
                        (3,\stromdrei) (3,\stromvier)
                        (4,\stromvier) (4,0)
                        (4.5,0)};
        \end{axis}
\end{tikzpicture}\end{document}
```

task where students have to determine the mean and effective value of periodic time functions.

## 3.2 Random number generation and LaTeX source code creation for circuit diagrams in MATLAB

Further typical tasks relate to the core area of the fundamentals of electrical engineering, circuit calculation or linear network analysis. The CircuiTi*k*Z package, which is also based on Ti*k*Z, can be used to represent electrical networks. With this package, basic elements such as resistors or voltage and current sources can be placed along paths. In addition, the elements and nodes can be labeled, current and voltage arrows can be drawn in, etc. The elements are placed according to a Ti*k*Z coordinate system.

The generation of different circuit diagrams for practicing the nodal analysis, which all have approximately the same complexity and the same computational effort, can for example be based on a fixed

**Listing 2**: Source code for the algorithmic generation of the matching sample solution to the diagram in Figure 2; for output see Figure 3

```
% Truncate decimal places for an integer number
\pgfmathdeclarefunction{trimzero}{1}
  {\pgfmathparse{ifthenelse(#1==round(#1),
                            int(#1),#1)}}
% Convert current (in A) to integers
\pgfmathsetmacro{\stromeins}{int(\stromeins)}
% Charge at the end of the 1st section (in As)
\pgfmathsetmacro{\ladungeins}{trimzero(\stromeins/2)}
% Differential operator (small upright d)
\newcommand*{\diff}{\mathop{}\!\mathrm{d}}

Current in the 1. section (1 point):
\begin{equation}
  i(t) = \SI{\stromeins}{\ampere\per\second} \cdot t
\end{equation}

Charge in the 1. section (1 point):
\begin{subequations}\begin{align}
  Q(t) &= \int \limits_0^t
                 \SI{\stromeins}{\ampere\per\second}
                 t' \diff t' + 0 \\
  &= \SI{\stromeins}{\ampere\per\second}
     \cdot \left[ \frac{t'^2}{2} \right]_0^t \\
  &= \SI{\ladungeins}{\ampere\per\second} t^2
\end{align}\end{subequations}

Charge at the end of the 1. section (1 point):
\begin{equation}
  Q(\SI{1}{\second}) = \SI{\ladungeins}
                          {\ampere\second}
\end{equation}
```

**Listing 3**: Source code (created automatically from MATLAB) for a circuit diagram using CircuiTi*k*Z for practicing nodal analysis; for output see Figure 4

```
\documentclass{standalone}
\usepackage{amsmath}
\newcommand{\ind}[1]{\mathrm{#1}}
\usepackage{circuitikz}
\begin{document}\begin{tikzpicture}[scale=1.3]
\draw (2,0) to[short] (0,0);
\draw (0,0) to[I, i^>=$I_{\ind{q}1}$] (0,2);
\draw (0,0) to[short, *-] (-2,0)
  to[R, l=$R_{1}$] (-2,2) to[short, -*] (0,2);
\draw (2,0) to[short] (4,0);
\draw (4,0) to[I, i_>=$I_{\ind{q}2}$] (4,2);
\draw (2,0) to[R, l^=$R_{2}$] (2,2);
\draw (0,2) to[short] (0,4);
\draw (0,4) to[I, i^<=$I_{\ind{q}3}$] (4,4);
\draw (4,4) to[short] (4,2);
\draw (0,2) to[I, i<^=$I_{\ind{q}4}$] (2,2);
\draw (4,2) to[R, l_=$R_{3}$] (2,2);
\node[circ] at(2,0) {};\node[below] at (2,0) {0};
\node[circ] at(0,2) {};\node[above left] at(0,2) {1};
\node[circ] at(4,2) {};\node[above right] at(4,2) {2};
\node[circ] at(2,2) {};\node[above] at(2,2) {3};
\end{tikzpicture}\end{document}
```

number of three nodes in the network, which are then connected to each other with different branches. With three nodes 1, 2 and 3 (plus the reference node 0), there are six possible branches ($0 \leftrightarrow 1$, $0 \leftrightarrow 2$, $0 \leftrightarrow 3$, $1 \leftrightarrow 2$, $1 \leftrightarrow 3$ and $2 \leftrightarrow 3$). Possible options for the branches are an open circuit, a resistor, a current source or a parallel connection of a resistor and a current source when calculating with DC currents and voltages. So there are $4^6 = 4096$ different possibilities.

Like the random values in the previous task for charge and current, the branches cannot be chosen completely arbitrarily, because it must be ensured that the generated network is actually computable. Therefore it has to be checked if there is at least one source in the network. Furthermore it has to be checked if all node sets can be fulfilled. For example, the network could be overdetermined by too many current sources or too many open-circuited branches. In this case, another parallel resistor must be added to a current source, or a current source or an open circuit must be removed and replaced by a resistor, respectively. To check whether a network is computable, the algorithm can calculate, for instance, the determinant of the nodal conductance matrix and check for inequality with zero.

In principle, it would be possible to perform all these queries and calculations directly in LaTeX. Instead, the author has decided to outsource the randomization of the tasks for nodal analysis to MATLAB due to the programming effort involved and to generate the corresponding LaTeX source code for the tasks as well as sample solutions from there. To export the LaTeX source from MATLAB, the simple `fprintf` command is used. The `dos` command can also be used to call `pdflatex` directly from MATLAB in order to create PDF files and send them to the students using the `sendmail` function [2].

Listing 3 (corresponding output in Figure 4) shows an example of LaTeX source code automatically generated in MATLAB for a circuit to practice node voltage analysis. The MATLAB program generating the LaTeX source runs in a loop over all branches of the network and outputs the corresponding source code depending on the circuit element. The $0 \leftrightarrow 1$, $0 \leftrightarrow 2$ and $1 \leftrightarrow 2$ branches require a pre-path or post-path from the node to the circuit element and/or back. Afterwards, a loop runs over all nodes, checks if more than two branches with circuit elements are

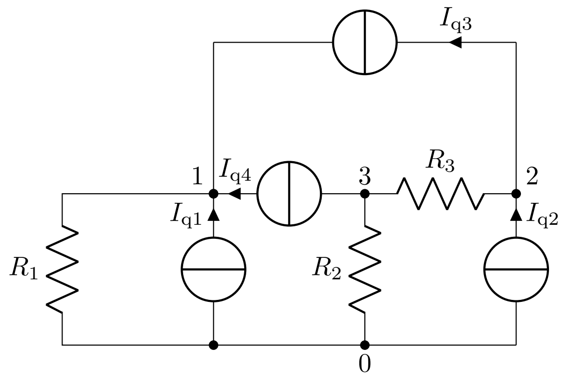


**Figure 4**: Randomized generated circuit diagram in CircuiTi*k*Z; output of Listing 3

connected to the node and, in this case, draws a solder point at the node.

The LaTeX source code for the corresponding sample solution is at least partially generated in MATLAB. An excerpt of the system of equations for the calculation of the network is shown in Listing 4 (corresponding output in Figure 5).

Further task types for nodal analysis with complex admittances in the frequency domain, for the equivalent resistance and impedance of a circuit, for the voltage divider rule and the phasor diagram can be found at `https://bit.ly/12PeerTasks`.

## 4 Conclusion & outlook

With a little conceptualization and programming effort (the author previously needed about 15 h to 20 h per task), randomized or personalized tasks can be created directly in LaTeX, or via source code generation in MATLAB, to practice basic computing procedures in the fundamentals of electrical engineering. With some creativity, the procedure could be transferred to other disciplines in the engineering sciences.

The concept is already tested in practice and well accepted by students. As an extension, additional task types are conceivable; for example, the current divider rule or stepwise calculation of current and voltage in a branched network. In addition, tasks with varying degrees of difficulty for internal differentiation in heterogeneous student groups would also be possible. Up to now, all students have received a task with similar difficulty and comparable calculation effort. In the example task on charge and current, the number of time steps could be increased or more linear, or even quadratic or exponential time functions could be inserted instead of constant charge functions. In the example task for the nodal analysis, networks with more nodes and thus more potential branches could be generated, whose computation requires a larger system of equations with more equations and more unknowns.

From a programming point of view, it would be interesting to implement more of the algorithms for task and solution generation in LaTeX instead of generating the source code externally in MATLAB. For the overall concept, it would be desirable if the generation and provision of the tasks for the students, the assignment and processing of the mutual assessments and the accounting of the achieved points would completely work in a learning management system like Moodle.

**Listing 4**: Source code (created automatically from MATLAB) for the system of equations to calculate the circuit from Figure 4 using nodal analysis; for output see Figure 5

```
\begin{equation*}
\begin{bmatrix}
G_{1} & 0 & 0 \\
0 & G_{3} & -G_{3} \\ 0 & -G_{3} & G_{2} + G_{3}
\end{bmatrix} \cdot \begin{bmatrix}
U_{\ind{Kn}1} \\ U_{\ind{Kn}2} \\ U_{\ind{Kn}3}
\end{bmatrix} = \begin{bmatrix}
I_{\ind{q}1} + I_{\ind{q}3} + I_{\ind{q}4} \\
I_{\ind{q}2} - I_{\ind{q}3} \\ - I_{\ind{q}4}
\end{bmatrix}
\end{equation*}
```

$$\begin{bmatrix} G_1 & 0 & 0 \\ 0 & G_3 & -G_3 \\ 0 & -G_3 & G_2 + G_3 \end{bmatrix} \cdot \begin{bmatrix} U_{\mathrm{Kn}1} \\ U_{\mathrm{Kn}2} \\ U_{\mathrm{Kn}3} \end{bmatrix} = \begin{bmatrix} I_{\mathrm{q}1} + I_{\mathrm{q}3} + I_{\mathrm{q}4} \\ I_{\mathrm{q}2} - I_{\mathrm{q}3} \\ -I_{\mathrm{q}4} \end{bmatrix}$$

**Figure 5**: Algorithmically generated sample solution to calculate the network in Figure 4 using nodal analysis; output of Listing 4

## References

[1] O. Cleynen, G. Santa-Maria, et al. Peer-graded individualized student homework in a single-instructor undergraduate engineering course. *Research in Learning Technology* 28, May 2020. `10.25304/rlt.v28.2339`

[2] D. Gleich. Get Matlab to email you when it's done running!, Feb. 2014. `https://dgleich.wordpress.com/2014/02/27/`

[3] J.B. Nitsch, U. Knauff, M. Magdowski. *Einführung in die Elektrotechnik*. Shaker-Verlag, Aachen, 2nd ed., July 2011.

◇ Mathias Magdowski
Otto von Guericke University,
Magdeburg, Germany
`mathias.magdowski (at) ovgu dot de`

# Creating a VPAT statement for TEX Live

Boris Veytsman, Keiran Harcombe

## Abstract

Governments around the world are enforcing accessibility standards. Vendors of software used by government agencies are required to file formal statements of accessibility for their products. This presents a special challenge for open source products, if they are not sponsored by a corporation.

In this paper we discuss our experience in creation of such a formal statement for TEX Live. While command-line tools are usually more accessible than GUI interfaces, the work turned out to be more difficult than we thought in the beginning.

## 1 Introduction

If we want to prove that the current scientific and technical progress is accompanied by moral progress, then the changing consensus about the people with disabilities may provide a solid argument. We now more and more think that the full participation of disadvantaged people in the life of society is not a generous gift from the society to them, but rather their right. Governments now often act to protect this right. In particular, according to current regulations, software vendors must make reasonable efforts to make their products accessible, and to formally report these efforts. In many countries, including the US and EU, these reports are becoming a condition for the software to be used by government agencies and contractors.

This presents a challenge for free software. Commercial vendors have resources to hire lawyers and experts to create formal compliance reports. Free software like TEX is often created by volunteers across the world. Volunteers are motivated to do "interesting" things rather than wade through pages of regulations and templates. On the other hand, the absence of compliance reports can undermine the usage of free software by the public sector and anybody dealing with the government agencies — which, in modern society, means everybody. Organizations such as TUG might be useful in this situation, since "uninteresting" work necessary for free software to thrive is clearly in their remit.

In this paper we discuss a pilot project in this vein: a VPAT statement for TEX Live.

## 2 Rules, regulations, templates

A template for a compliance report was developed by the Information Technology Industry Council (ITI).[1] It is available as the Voluntary Product Accessibility Template (VPAT)[2] and is free to use. The template is based on several sources:

1. Web Content Accessibility Guidelines developed by the W3 Consortium (WCAG).[3]
2. Revised Section 508 Standards by the US government.[4]
3. EN 301 549 Accessibility requirements suitable for public procurement of ICT products and services in Europe.[5]

WCAG is not, strictly speaking, a government standard. However, both US and European standards require compliance with WCAG, or, in lawyers' lingo, incorporate it.

ITI's VPAT template has several variants, or "editions". Since TEX is used throughout the world, we chose the largest one, the International edition, that reports compliance with both US and EU standards.

The standards consider several categories of software, as follows:

*Web documents* are documents (in any format) published on, as you might surmise, the World Wide Web. This is what WCAG specifies.

*Electronic documents* are documents published in any other manner. Even if a product is not itself a document, the government standards say, in essence, that its documentation must be accessible and thus comply with WCAG standards, even when it is not published on the web.

*Software* is a generic term for software.

*Closed system* does not mean what we in the Free software community mean by these words: the standards define closed systems as those which do not allow an easy interaction with assistive technologies.

*Authoring tool* is software for the creation of documents.

The compliance report may discuss compliance with any or all of these categories, with one exception: documentation is usually a part of any category.

## 3 Our decisions and the lessons learned

Initially we thought that our task would be easy. TEX and friends are command line tools. We know

[1] `https://www.itic.org/`

[2] `https://www.itic.org/policy/accessibility/vpat`

[3] `http://www.w3.org/TR/2008/REC-WCAG20-20081211` and `https://www.w3.org/TR/WCAG21`

[4] `https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-ict-refresh/final-rule/text-of-the-standards-and-guidelines`

[5] `https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.01.01_60/en_301549v030101p.pdf`

that command line tools are very accessible: people with low/no hearing or low/no vision can use command line tools with the existing assistive technology. However, when we got better acquainted with the standards and the requirements, we understood this was not as easy as seemed.

First, the compliance report cannot be generic. We cannot say that any implementation of TeX and its ecosystem is accessible. Therefore we decided to choose at first a concrete implementation, which could be installed and operated with assistive technologies. We chose TeX Live 2021: our flagship implementation of TeX and friends. We plan to add other implementations to the list in the future.

Second, we must report whether the documentation for TeX is compliant. This immediately causes a question, what is TeX documentation? At the time we counted, TeX Live 2021 had 36 769 files in the `doc` tree in various formats: mostly text, PostScript, PDF and HTML. Obviously many of them are not accessible. Fortunately, it is not needed to read them *all* to successfully use TeX Live. Also fortunately, in practice the situation is ameliorated by the long-standing policy of TeX Live to require source code for all its documentation: text source code can be read by voice software.

We decided to discuss only one piece of documentation: the TeX Live manual in HTML format at `https://tug.org/texlive/doc/texlive-en`. The WCAG guidelines define three levels of compliance, denoted as A, AA, and AAA. Only the first two are required by the US and EU standards. The third one contains rather difficult requirements such as automatic explanation of abbreviations and special terminology.

However, we found problems even at the first two levels. Namely, the guidelines require alternative text for all images. The TeX Live manual has images for GUI installation mode. One can argue that these images are not especially relevant for low vision users, who are probably going to use text mode installation anyway. However, this is a weak argument, and we need to make the manual fully accessible.

Further, the category *closed systems* is evidently not applicable for TeX Live: it is open not only as open source software, but also as software which allows the user to integrate it with the assistive technologies.

We hit a snag, however, when looking into the requirements for *authoring tools*. Simply saying, these requirements say that the software is not just accessible by itself, but also the documents it creates are accessible. Now the TeX community and TUG have and continue to put considerable efforts into creating the tools for accessible output, notably tagged PDF. However, at present the creation of accessible documents with TeX, while possible, still requires significant effort. Thus we made a decision to not yet make a statement about TeX Live as an authoring tool. That is, we state that, for example, a blind person can efficiently learn and use TeX to create a document. However we do *not* state that this document will be accessible even to its author. This is a sad situation, but hopefully it will be ameliorated soon.

The resulting draft statement is available at `https://github.com/TeXUsersGroup/TeX-VPAT`.

## 4 Next steps

Our work showed that there is much to do with TeX compliance documentation.

First, we must make the TeX Live manual fully compliant by adding alt text to all images.

Second, we need to make the statement itself fully accessible, either in PDF or HTML format, or both.

Third, we need to add separate statements for other distributions: MacTeX, MikTeX, proTeXt... We would like to invite volunteers best acquainted with these distributions to help with them; we hope it might be easy enough to follow our example.

Fourth, it is important to add the statement about TeX as an authoring tool. Perhaps we may put in statements about ConTeXt, which currently has good capabilities for creation of tagged PDF, or the usage of TeX4ht for creation of accessible HTML files. This requires further research and decisions.

The task of making all TeX Live documentation accessible seems to be very difficult, but we might start with the guidelines for package authors.

Last but not least, we plan to release the statement as a CTAN package.

Making TeX accessible is an ongoing process. We are glad to be a part of it.

◇ Boris Veytsman
George Mason University (US),
TeX Users Group
`borisv (at) lk (dot) net`
`http://borisv.lk.net`

◇ Keiran Harcombe
Open University (UK)
`kjh (at) harcombe (dot) net`
`https://harcombe.net/~keiran/contact`

# `latex2nemeth`: A direct LaTeX-to-Braille transcribing tool

Andreas Papasalouros, Antonis Tsolomitis

## Abstract

The Braille system allows the tactile representation of characters in various alphabets, giving access to reading texts to visually impaired persons. The Nemeth code allows the representation of mathematics symbols and expressions into the Braille system.

We have developed a tool, named `latex2nemeth`, for the reliable transcription of LaTeX documents to Nemeth Braille, thus facilitating the access of visually impaired students to studying science. In order to support the extensive set of mathematics symbols covered by TeX, we have proposed some new symbols based on the extension mechanisms of the Nemeth code. With the aid of `latex2nemeth`, we have created a repository of learning material in Braille/Nemeth code aiming to support studies in mathematics for visually impaired students. While most of the material available in the repository is in the Greek language, the tool supports other languages as well. `latex2nemeth` is currently available in both the TeX Live and MiKTeX distributions.

## 1 Introduction

In 2014 we came to know about a visually impaired student of mathematics at the University of Athens. The student could read and write in Braille notation and in the Nemeth representation of mathematics. Although the lecture notes and course textbooks were available in LaTeX source format, there was no reliable way to translate these sources into tactile representations, accessible by visually impaired persons. The student had to manually transcribe learning content with the help of a seeing person. We set out to create a tool for transcribing LaTeX texts to tactile representations accessible to blind persons. The above is an instance of a wider problem: Visually impaired students of related fields did not have access to the bulk of study material available in LaTeX format.

## 2 The Braille and Nemeth codes

### 2.1 The Braille code

The Braille system allows the tactile representation of characters in various alphabets, giving access to reading texts to visually impaired persons. The six-dot Braille system supports the representation of $2^6 = 64$ different characters. An assignment of Braille symbols to letters defines a certain Braille alphabet (e.g., English, Greek, etc.).

**Figure 1**: Reading of Braille/Nemeth code

### 2.2 The Nemeth code

The Nemeth code allows the representation of mathematics symbols and expressions in the Braille system. It defines a set of *rules* that designate the combinations of Braille symbols that describe various types of mathematical structures.

Nemeth's original document [4] specifies 25 rules covering various aspects of mathematical structures such as: numeric signs and symbols, alphabets, fractions, superscripts and subscripts, radicals, symbols of grouping, spatial arrangements, etc.

## 3 The `latex2nemeth` tool

The aim of the `latex2nemeth` program is twofold:

- The *reliable* transcription of books and electronic notes from LaTeX to Nemeth/Braille.
- The creation of a repository of mathematical texts available to visually impaired students and researchers.

At least at the beginning of this project (end of 2014), to the best of our knowledge, no tool or method met the above requirements.

### 3.1 Existing solutions

At the time of the design of this project, few solutions existed for transcribing LaTeX documents into Braille.

An available commercial solution implied the following steps: 1) conversion from LaTeX to MS Word with MathType; 2) conversion from MS Word to Braille with the use of the Duxbury software.

The above method supports only a small subset of LaTeX commands, thus, the process produced unreadable Braille code with many mistakes (through extensive tests during 2014).

An open source solution with the use of the `liblouis` library [3] was also extensively tested in the initial phases of this project. This solution had the following steps: 1) conversion from TeX into MathML (e.g., with TeX4ht [6]); 2) conversion from MathML into Braille with the `liblouis` library. This process also created Braille code with many errors, given that the `liblouis` library did not aim at supporting TeX at the time of the creation of the `latex2nemeth` program.

### 3.2 Program features

The main features of the `latex2nemeth` software are the following:

- LaTeX files with text in Greek, English and Ancient Greek are converted to Braille.
- More than 850 different mathematical symbols and expressions are supported.
- All TeX AMS mathematical symbols are covered, among others.
- Different Braille alphabets can be supported with the use of different *symbol tables*.
  - A symbol table is a JSON file that maps individual characters or TeX commands to Braille characters or sequences of characters.

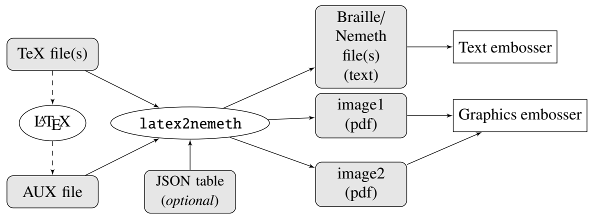


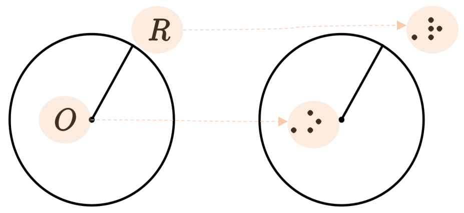
**Figure 2**: Flow of translation to Braille/Nemeth

The process of generation of Braille text is illustrated in figure 2. The `latex2nemeth` tool takes as input a source TeX file, or set of files, as well as the corresponding `.aux` file. The latter is used for expressing in Braille numbered elements such as figures, sections/subsections, mathematical equations, etc. The output of the tool is a set of Unicode (UTF-16) text files with Braille characters. These files are first converted to UTF-8 by an appropriate tool (such as `iconv`) and then imported to LibreOffice using a font that supports 6-dot Braille (such as `NewComputerModern`) and printed in tactile six-dot Braille form in a specialized text embosser with the use of `odt2braille` LibreOffice extension. The above process is automated by an appropriate script, available in the tool distribution.

In its default configuration, the tool transcribes texts that are written in English or in modern Greek. As seen in figure 2, the tool takes as an optional input a JSON table file. This file can be provided by the user in order to support a Braille alphabet different than the above such as Hebrew, Cyrillic, etc. As a demonstration of this extensibility feature, we provide in the standard distribution of the tool a JSON file named `polytonic.json` with the definition of the Ancient Greek Braille alphabet.

#### 3.2.1 Image label filtering

The `latex2nemeth` tool provides limited support for converting text inside images into Nemeth format. More specifically, images defined with the PSTricks library may contain labels with mathematics expressions delimited with the symbol $. These expressions are filtered by the tool and they are translated in Nemeth, as illustrated in figure 3. The generated images are exported separately in PDF and can be printed separately in a specialized graphics embosser, as shown in figure 2.



| Original code | Transcribed code |
|---|---|
| `\begin{pspicture}(-1,-1)(1.2,1.2)` | `\begin{pspicture}(-1,-1)(1.2,1.2)` |
| `\pscircle(0,0){1}` | `\pscircle(0,0){1}` |
| `\qdisk(0,0){1pt}` | `\qdisk(0,0){1pt}` |
| `\uput[180](0,0){$O$}` | `\uput[180](0,0){⠠⠕}` |
| `\psline(0,0)(.5,.866)` | `\psline(0,0)(.5,.866)` |
| `\uput[30](.5,.866){$R$}` | `\uput[30](.5,.866){⠠⠗}` |
| `\end{pspicture}` | `\end{pspicture}` |

**Figure 3**: Transcribing image labels

## 4 Implementation

The transcriber is based on a parser for the LaTeX language, created specifically for this project. The parser recognizes the most common LaTeX commands and environments in text and mathematics modes and also covers most structures and mathematical symbols. The program is developed in Java. The lexical analyzer and the parser were developed using the JavaCC compiler generation tool [2]. The tool is available as a JAR (Java archive) package.

### 4.1 LaTeX to Braille transcription

Each paragraph and each environment in the input LaTeX sources is processed separately. In text mode, each lexical token is recognized and transcribed into its corresponding Braille symbol by using a certain *symbol table* given in the form of a JSON file, as mentioned earlier. Numbers are lexically scanned as atomic entities, e.g., 13.455, since, according to Braille code, a certain numeric indicator (⠼) must precede the whole number instead of each numerical symbol in the number.

## 4.2 Processing of mathematical expressions

Mathematical expressions are parsed into appropriate syntactical trees in memory. The abstract syntax trees for mathematics expressions are independent of the target language (Nemeth). Depending on the type of expression (fraction, superscript/subscript, etc.) an appropriate procedure (semantic routine) generates corresponding Nemeth code.

An object-oriented representation of mathematical expressions was adopted, based on the Composite and Interpreter design patterns [1].

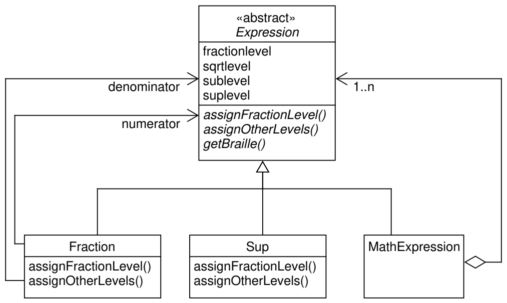


**Figure 4**: Class diagram of basic mathematical expression classes

## 4.3 Expression indicators and depth

In order to assist a blind person in conceiving the structure of a mathematical expression, in Nemeth code, an *expression indicator* signifies not only the kind but also the *depth* of certain expressions that can be nested such as fractions, subscripts and superscripts, root expressions, etc.

$$e^{x^{b+1}} + \frac{c}{d+\frac{k}{x+3}}$$

The above expression is rendered in Nemeth Braille code as shown in figure 5. In the example, the fraction and the exponent are defined by appropriate *expression indicators*. Thus,

```
e<exp>x<exp-2>b+1<base>
+
<openfrac-2>
c
<fractionbar>
d
+
<openfrac>k<fractionbar>
x+3
<closefrac>
<closefrac>
```

**Figure 5**: Expression indicators in nested expressions

- `<exp>` ( ⠘) signifies a simple superscript
- `<exp-2>` ( ⠘ ⠘) signifies a superscript within a superscript.
- `<openfrac>` (⠹) signifies the opening of a single fraction.
- `<openfrac-2>` ( ⠠⠹) signifies the opening of a complex fraction, etc.

Depth indicators for various expressions is calculated by appropriate methods of the instances of the *Expression* Java class, as illustrated in figure 4: the depth for complex fractions is calculated *bottom-up* by the `assignFractionLevel()` method, while the depth for other nested expressions (superscripts, subscripts, roots, etc.) is calculated *top-down* by the `assignOtherLevels()` Java method.

## 4.4 Spatial arrangements

Certain mathematical structures such as arrays, case and align expressions, etc., are specially aligned, in the sense that their elements are positioned in a certain horizontal or vertical arrangement. Since the Nemeth code supports spatially aligned structures, the `latex2nemeth` tool generates spatially aligned Nemeth code. The following example defines an `array` structure:

```
$$\chi_F (x)=\left\{\begin{array}{ll}
  1, & \textrm{if } x\in F\\
  0, & \textrm{if } x\notin F
  \end{array}\right.$$
```

This structure corresponds to the definition:

$$\chi_F(x) = \begin{cases} 1, & \text{if } x \in F \\ 0, & \text{if } x \notin F \end{cases}$$

For the above, the tool generates the following spatially aligned Braille code:

```
  ⠨⠯⠰⠠⠋⠐⠷⠭⠾⠨⠅
⠠⠨⠷⠼⠂⠠  ⠊⠋⠭⠈⠑⠠⠋⠠
⠠⠨⠷⠼⠴⠠  ⠊⠋⠭⠌⠈⠑⠠⠋⠠
```

## 5 Extension of the Nemeth code: Proposing new symbols

The Nemeth code prescribes certain mechanisms for defining new symbols. As an example, the symbol $\hookrightarrow$ (`\hookrightarrow`) is defined based on the plain arrow, expressed as ⠒⠒⠕ in Nemeth. The new, modified, Braille symbol is ⠫ ⠈⠷⠒⠒⠕ This new symbol:

- Starts with character ⠫ that signifies grouping of characters in a new symbol.
- Contains the combination ⠈⠷ that signifies the characteristic hook.

To support the extensive set of symbols used in university-level mathematical texts, by using extension mechanisms such as the above, we have proposed other new Braille mathematical symbols as well.

## 6 Evaluation — Reliability of transcription

A formal evaluation has been conducted [5] using mathematical documents from the AMS. The quality of the transcription was evaluated by comparing the transcribed documents with back translations generated by two visually impaired persons with good knowledge of the Nemeth code. The average error of the two back translations was 2.4% while the inter-rater agreement was very high (Cohen's kappa $\approx 0.98$).

The tool fails to discriminate among symbols *period* and *decimal point* in the TeX sources, which results in a small number of errors, given that these symbols have different representations in Braille/Nemeth, but the same representation in TeX (`.`). Also, the tool adds a numeric indicator before every number, while, according to the Nemeth rules, some numeric indicators should be omitted in certain numeric expressions. Although this output does not strictly conform to the Nemeth rules [4], the redundancy of some generated numeric indicators was found not to confuse the reader, and result in only a slight increase in the Braille text size.

As an informal evaluation, much of the generated content has been used as study material by the blind student mentioned before. The student has graduated on schedule and now she is pursuing graduate studies with the help of `latex2nemeth`.

## 7 Distribution and documentation

The website of the `latex2nemeth` project is: `myria.math.aegean.gr/braille/index-en.html`. The project can also be accessed via the CTAN repository at `ctan.org/pkg/latex2nemeth`, and its source code can be accessed at `sourceforge.net/projects/latex2nemeth`. The `latex2nemeth` program is currently available in the TeX Live, MiKTeX and MacTeX distributions.

The following documentation is available at the project website:

- Dictionary of mathematical terms (from symbol name to Nemeth).
- Reverse dictionary of mathematical terms (from Nemeth to symbol name).

The above dictionaries are available in both printed and Braille forms, so they are accessible by both seeing and visually impaired persons.

### 7.1 Towards a mathematical/scientific repository for the visually impaired

With the use of the tool we are creating a repository of publicly available mathematical/scientific content in Braille code. So far, the following complete texts have been transcribed: 1 book in English, 23 books and course notes in Greek, and 2 books in ancient Greek: Homer's Odyssey and The Orphic Hymns. All books are available via the project's website.

## 8 Limitations and future work

The tool has limited support for macros. Currently a simple parameter substitution mechanism is implemented. Thus, we plan a full re-implementation with full macro expansion support.

The discrimination among symbols *period* and *decimal point* is planned to be resolved in a forthcoming version. Also, we are working on avoiding redundant numeric indicators, as described above, by applying the appropriate Nemeth rules based on deep parsing of numerical expressions.

We are also working on more extensive support for images, beyond the simple filtering of image labels for PSTricks figures mentioned above.

**Acknowledgments**

We would like to thank Mrs. Olga Maleza for the support and the information that she has provided from the beginning of this project.

The development of the tool was partially supported by the University of the Aegean Research Unit (grant no. 2625).

## References

[1] E. Gamma, R. Helm, et al. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley, Boston, MA, USA, 1995.

[2] The JavaCC compiler compiler. `javacc.github.io/javacc`.

[3] The `liblouis` project. `liblouis.org`.

[4] A. Nemeth. The Nemeth Code 1972. `brailleauthority.org/mathscience/nemeth1972.pdf`.

[5] A. Papasalouros, A. Tsolomitis. A direct TeX-to-Braille transcribing method. *Journal of Science Education for Students with Disabilities* 20(1):36–49, 2017. RIT Scholar works.

[6] The `tex4ht` project. `tug.org/tex4ht`.

◇ Andreas Papasalouros,
Antonis Tsolomitis
Department of Mathematics,
University of the Aegean
832 00 Karlovassi, Greece
`andpapas (at) aegean dot gr`,
`atsol (at) aegean dot gr`

# On the road to Tagged PDF: About StructElem, Marked Content, PDF/A and Squeezed Bärs

Ulrike Fischer

## Abstract

In this article I present two packages as part of the LaTeX Project's "Tagged PDF" effort:

`tagpdf` which contains the core code to create a tagged PDF and is used by the LaTeX team to test new code.

`pdfmanagement-testphase` which contains a large number of PDF-related commands and tools and installs a new management command for central PDF dictionaries.

I will show how to use these packages and the benefits they will bring for the average user, while also mentioning resulting incompatibilities and required changes in documents.

## 1 Introduction

At its core, PDF is a page and graphic-related format: it allows describing, very accurately, a page layout, the font sizes, font types, colors, placement of characters and graphical elements, but essentially it simply says where to put ink on the page. It is possible to extract text but one doesn't get the semantic meaning of such a text, the reading order can be unclear, and — with PDF produced by TeX — it can even be unclear where words begin and end.

However, PDF is also quite an extensible format. In the same way you can enhance HTML with Cascading Style Sheets (CSS) and JavaScript, you can enhance PDF in various ways: you can embed files with source code or other material, you can add alternative text to objects. And, most importantly for our purposes, you can *tag* it: with this you are adding a structural layer over the content (figure 1).

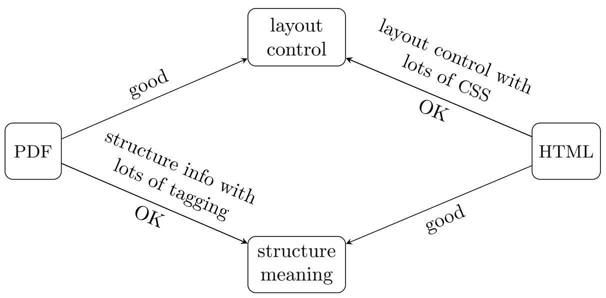

layout control
good
layout control with lots of CSS
OK
PDF
HTML
structure info with lots of tagging
OK
good
structure meaning

**Figure 1**: PDF vs. HTML, layout vs. structure

## 2 Some technical details

In the PDF format, every page is stored in a *page stream.* This stream contains various operators which select fonts, move around on the page, and insert images and text.

As described in a previous article [1], tagging requires first the writing of markers called *Marked Content* (MC) markers into the page stream which label and number all chunks of meaningful content.

The next step is to add various objects to the PDF. These objects contain references to each other as parent and kid objects and so form a tree. The numbered *Marked Content* chunks are leaf nodes in this tree. Figure 2 shows a simple tree created with such objects. The last step is to create various objects for cross-referencing and housekeeping, and to register them in the catalog so that a PDF viewer can find and use the structure.

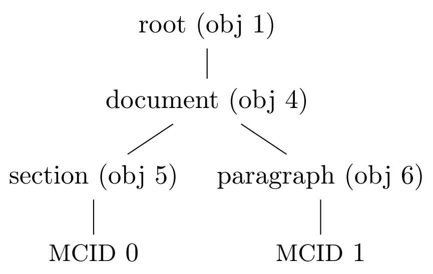


**Figure 2**: Simple structure tree

## 3 Inspecting a tagged PDF

Checking the tags of a PDF can be done with Adobe Acrobat Pro; some free options are also slowly appearing. One of them is the online editor at `ngpdf.com`. It allows you to upload a tagged PDF, view the tag structure and view and also export an HTML representation of the PDF. If the math has an embedded associated file with the MathML representation, then the HTML will use MathJax to render it.

Unlike the copy & paste heuristic of PDF viewers, the site doesn't try to guess the reading order but shows you what you get from the tags and the structure tree. The HTML export demonstrates that tagging isn't only for screen readers. It gives you more options to *reuse* the PDF. The old saying that PDF is like scrambled eggs and that you can't get back an egg (a structured LaTeX document) from it is no longer true: You still don't get all the details back, but depending on the amount of tagging that's been added, you can restore quite a lot.

## 4 Amount of tagging: About (im-)perfection and standards

A document must be tagged in its entirety. You can't tag only a specific section or some table or some math

Ulrike Fischer
doi.org/10.47397/tb/42-2/tb131fischer-tagpdf

to improve its export or copy & paste, even if the rest of the document already works satisfactorily.

But this still leaves you with quite a lot of freedom about the amount of tagging. If a paragraph contains some emphasized text you can tag it as an `Em`-structure, but you don't have to. This naturally means that some information will be lost, but quite apart from the fact that is often impossible anyway to mirror the intention of every detail of a layout into such a structure tree, tagging doesn't have to be perfect to be useful; it improves accessibility significantly even if merely headings are tagged for better navigation and paragraphs are shown in the correct reading order.

The same goes for standards: There exist various guides and standards which describe requirements for accessible and well-tagged PDF documents. They and the validators checking such standards are useful to set goals which we should aim to reach, but that it is often not (yet) possible to fulfill all aspects of such a standard should not stop you from creating more usable documents.

The end users of the documents we produce are not validators but intelligent humans, and intelligent humans can often handle imperfect documents surprisingly well, and in the end, their feedback is what will really matter.

## 5 Tagging with the `tagpdf` package

I wrote the `tagpdf` package (`ctan.org/pkg/tagpdf`) around four years ago. At that time I was already convinced that tagging at a reasonable scale can't be done in an external package, but that changes would be required in LaTeX itself. If we want a large number of documents to be tagged, "normal" tagging shouldn't need expert skills and fragile patches but should be supported by LaTeX directly.

So the goals of `tagpdf` were

- to develop the basic tagging code for the kernel,
- to provide examples and tests,
- to identify LaTeX problems, and
- to develop stable solutions which can be eventually integrated into LaTeX.

This means that `tagpdf` wasn't written as a standard user package, but rather primarily as a research and development tool and will eventually vanish again. As such

- it requires the newest LaTeX code, sometimes even LaTeX-dev,
- it can still change,
- it concentrates on basic commands, and
- it doesn't patch other packages.

**Listing 1**: Tagging commands

```
\tagstructbegin{tag=P}%
 \tagmcbegin{tag=P}...text...\tagmcend
 \tagstructbegin{tag=Link}
  \tagmcbegin{tag=Link}...text...\tagmcend
 \tagstructend
  \tagmcbegin{tag=P}...text...\tagmcend
  %pagebreak: new mc-chunk
  \tagmcbegin{tag=P}...text...\tagmcend
\tagstructend
```

`tagpdf` defines a number of commands, but the two core command pairs are these: First, the commands to mark the MC-chunks. As they split the text into small labeled chunks these commands should be used linearly: `\tagmcbegin ... \tagmcend`.

Secondly, there are the commands to mark the structure. They are typically nested: `\tagstructbegin ... \tagstructend`.

Listing 1 shows a simple example of their use. It is important to note that if there is a page break in the middle of a text chunk, the MC-marker must be closed before the break and reopened after the break. This is easy to do manually but rather challenging if it has to be done automatically, at least with other engines than LuaLaTeX. Frank Mittelbach's companion talk discusses this.

Listing 2 shows the source code of a small but complete document using `tagpdf`. In general, tagging works best with LuaLaTeX. To use other engines, you need the code mentioned in Frank's talk, or mark up all page breaks manually. Neither LaTeX & dvips nor XeLaTeX support real space characters.

There are a few important points to note here:

- The document should be compiled twice, sometimes more. Be aware that you don't necessarily get a rerun message yet, even if one is needed.
- `tagpdf` requires the `pdfmanagement-testphase` package. I will say more about this below.
- `tagpdf` is still a normal package and can be loaded with `\usepackage`, but as a first step towards full integration into LaTeX itself it can also be loaded with the `testphase` key.
- Tagging must be activated explicitly. This can be done as in the example with the `activate` key, and also with the `\tagpdfsetup` command from the package (see the documentation).
- The example shows lots of tagging commands around the section, but none at all around the paragraph or the link.

This last point reflects the state of the project: With the help of the work done over the last months

**Listing 2**: A full example

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{pdfversion=2.0,
  testphase={tagpdf}, activate=tagging}
\documentclass{article}
\usepackage{hyperref}
\pagestyle{empty}
\begin{document}
\tagpdfparaOff
\tagstructbegin{tag=H1}\tagmcbegin{tag=H1}
\section{A section}
\tagmcend\tagstructend\tagpdfparaOn

A paragraph with some text and a link to
\url{https://www.latex-project.org}

Another paragraph \ldots
\end{document}
```

on the new paragraph hooks and PDF management, paragraphs and links can be automatically tagged, but sectioning commands still need either manual tagging or patches. It will be the next task to automate this too, which will require some months' work. You may wonder why it takes so long to insert what amounts to two lines of code. The first reason is that the code must be added to commands which are used, patched and redefined by many classes and packages. If we simply change them to support tagging, this would break these classes and packages. So we need a proper change strategy here. The second reason is that as we are obliged to change the sectioning commands, we want to use the opportunity to modernize and improve them.

## 6 PDF management

Tagging writes many objects and other code into the PDF. This requires proper tools. "Proper tools" means, above all, abstracted, backend-independent tools provided directly by LaTeX. When I started with `tagpdf` there was nearly nothing available. You either had to use some external package or a primitive command. By now the situation has changed dramatically, and for the better: The LaTeX format now includes the L3 programming layer (`expl3`) code and loads backend files from `l3backend`. This means that even in small documents, commands to write PDF objects, to set the PDF version, and to use colors are available.

The new PDF management support, currently in the external package `pdfmanagement-testphase`, extends this set of tools. It offers commands to write the MC-markers, to embed files (needed to add associated files to a structure), to create links which have suitable hooks for the automatic tagging, commands to create form fields with built-in tagging support, and more.

But the PDF management does more: In a PDF there are a number of central dictionaries to which packages shouldn't write directly, to avoid clashes with other packages. Until now, packages had no way to avoid such problems apart from testing for potentially problematic packages, but the new PDF management will resolve this problem: It offers a command which allows to write to the dictionaries in a managed way. The command exists in two versions, for LaTeX $2_\varepsilon$ code and for `expl3` code: `\PDFManagementAdd` and `\pdfmanagement_add:nnn`, respectively, which together we'll call `pdfmanagement` for short. Details on how to use the command can be found in the documentation `l3pdfmanagement`.

### 6.1 Incompatibilities

This new `pdfmanagement` support replaces the five primitives `\pdfinfo`, `\pdfcatalog`, `\pdfpageattr`, `\pdfpagesattr` and `\pdfpageresources` and the analogous commands of the other engines and backends. *Replace* truly means *replace* here: every package which uses those primitives is incompatible with the new `pdfmanagement`. For this reason we built a safety net around its use:

- The code has not been added directly to the kernel, but is offered now as an external testphase package.
- For the central management command, there is an explicit activation trigger command: `\DeclareDocumentMetadata`.
- We already wrote a number of replacements for incompatible packages and drivers, for example `hyperref`, `transparent`, `pdflscape`.
- We wrote a number of "firstaid" patches for packages which will stay until the packages adapt their code, e.g., `pgf` and `hyperxmp`.
- We notified various package and class maintainers and asked them to adjust their code.

But we can see problems only with packages that are on CTAN and in the TeX distributions. For the various house classes and packages of universities and journals we need the help of users to report problems. *So please test and check!*

### 6.2 Benefits

The new `pdfmanagement` is not only good for tagging. The rewriting and checking of PDF related code and packages was also used to modernize, correct and extend various other features. I would like to mention a few improvements.

**Listing 3**: New hyperref options

```
\hypersetup{href/protocol=https://,
            href/urlencode}
\hrefurl{www.köln.de}{Köln}
\url[format=\textsc]{www.köln.de}
```

**Listing 4**: Rotating of float pages

```
\begin{figure}[p]
\PDFManagementAdd{ThisPage}{Rotate}{90}
... landscape picture ...
\end{figure}
```

### 6.2.1 Colors and link options for hyperref

When the new pdfmanagement is used, hyperref (ctan.org/pkg/hyperref) uses a new, generic driver. This driver uses better default colors, has keys to change to other color schemes, and styles like linkcolors can now be changed at any time in the document. The support for non-ASCII links has been extended: links can be input in UTF-8 and hyperref will convert them internally into the needed "percent" encoding. It is possible to preset a protocol, most likely https://, which is then prefixed to all links. There is also a hook system which allows packages to add support to differentiate internal links by concepts such as citation, acronym, glossary and similar. See listing 3.

### 6.2.2 Rotations of float pages

With the pdfmanagement command it is possible to rotate float pages; depending on the engine this requires one or two compilations. See listing 4.

### 6.2.3 Support for PDF standards

The pdfmanagement bundle also contains a module for PDF standards. Currently it mostly handles A-standards. When such a standard is requested the code will, for example, include a color profile and register it in the PDF catalog. It will also enable verification tests that other packages can use in their code; for example, the new hyperref driver uses this to suppress JavaScript actions. This replaces the key pdfa which is no longer used by the new driver. XMP-metadata can be added by loading the hyperxmp package (ctan.org/pkg/hyperxmp). See listing 5.

### 6.2.4 Preserving links

As part of the work, a modernized version of the pax package (ctan.org/pkg/pax) from Heiko Oberdiek has been written. The new package is called newpax

**Listing 5**: Specifying a PDF standard

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{pdfstandard=A-2b}
\documentclass{article}
\usepackage{hyperxmp}
\usepackage{hyperref}
....
```

(ctan.org/pkg/newpax), and it supports including PDFs inside another PDF without losing the links. It works with more backends and engines and, unlike pax, it doesn't require an external Java application, instead using LuaTEX.

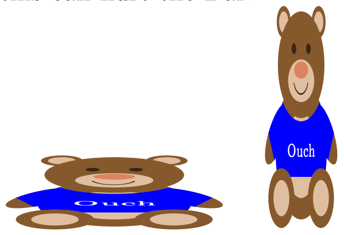
### 6.2.5 New form field code

Finally I would like to mention the code for form fields, which has been completely rewritten.

The new code is currently in a testphase package called l3pdffield-testphase, which is part of the pdfmanagement-testphase bundle. It is not yet decided if it will be part of hyperref like the old code or stay in a separate package.

One important change in this code is that it has been adapted to PDF 2.0. This means that it uses "appearances" in various places for the look of the fields (appearances are small graphics called "form XObjects", stored in the PDF). Using appearances means that radio buttons or checkbox fields are no longer restricted to characters but can contain arbitrary images, e.g., animals from the tikzlings package (ctan.org/pkg/tikzlings).

But be careful! Appearances are squeezed by the PDF viewer into the dimensions of the form field, and this can hurt the Bär!



### References

[1] U. Fischer. Creating accessible pdfs with LATEX. *TUGboat* 41(1):26–28, 2020. https://tug.org/TUGboat/tb41-1/tb127fischer-accessible.pdf

◇ Ulrike Fischer
LATEX project team
Mönchengladbach
Germany
ulrike.fischer (at) latex-project.org

# TEI-XML to LaTeX workflow: Issues and lessons

Nicolás Vaughan

## Abstract

In this paper I discuss some of the issues surrounding the workflow used in the production of the annotated Spanish translation of the medieval work, *Salomon et Marcolfus*. I explain the decisions taken regarding the XSLT transformation of the TEI-XML document, in order to produce a final print-ready PDF from the LuaLaTeX text.

## 1 Introduction

Several methods are available for converting XML documents into PDF format. Perhaps the most well-known is that of XSL Formatting Objects (XSL-FO) [2]. However, given that XSL-FO was originally designed to produce technical documents, it is quite difficult — if not impossible — to produce other kinds of more sophisticated publications with it. For example, while technically possible, it is impractical to create critical editions using XSL-FO. Moreover, the only available open source application for generating PDFs from XSL-FO sources — the Apache Formatting Objects Processor (FOP) — offers limited compliance with the W3C XSL-FO 1.1 Standard. For example, it does not support interesting features such as indexes and table properties [1], which are now covered by the standard.

Another, more recent, method of producing PDF from XML involves the use of a technology called CSS Paged Media, originally conceived by researchers at Mozilla [3]. Since XML cannot make direct use of CSS, this method requires that the XML sources be transformed — e.g., via XSLT — into HTML files, which, together with the appropriate CSS, can later be converted into PDF. An application is ultimately responsible for carrying out that conversion. Several free and/or open source (FOSS) applications such as wkhtmltopdf [20] and WeasyPrint [16] are available. As in the case of XSL-FO, however, the problem with such FOSS applications is that they provide subpar performance when compared with commercial ones. On the other hand, commercial applications such as PDFreactor [5] and PrinceXML [6] are expensive, often costing thousands of dollars.

Bearing this in mind, the approach we took for the present project was to use XSLT transformations to convert the original XML (more exactly, TEI-XML) into a LuaLaTeX document, which in turn would be used to generate a PDF file. This should allow for the highest possible quality in a print-ready PDF file, while at the same time affording the availability of useful LaTeX packages for generating bibliographies, indexes, nomenclatures, and the like.

Nonetheless, this workflow proved to be not as straightforward as originally thought. Leaving aside the problem of dealing with undesired whitespace (see §4), the greatest complication concerned the decision of where to carry out most of the granular processing (e.g., how to transform an XML `<cit>` element into a full-fledged LaTeX citation) — in the XSLT code or in the LuaLaTeX code. Ideally one wishes for such processing to occur mostly in one or the other of the stages, as this simplifies the code and facilitates debugging. In actuality, however, this was not possible. The idiosyncrasies of each language made it necessary that some parts of the processing be preformed in the first stage, while others in the second one.

## 2 Background

The project alluded to above is an original annotated translation into Spanish of the Late-Medieval, anonymous work, *Salomon et Marcolfus*, written in Medieval Latin. The objective was to produce a TEI edition of the Latin text (based upon the 1912 critical edition), with an accompanying, digitally-born Spanish translation, also in TEI. An accompanying Middle-English translation, as well as an original critical edition created from the collation of a score of surviving incunabula, are also planned.

All TEI texts were transformed using XSLT into XHTML documents destined for a web version, on the one hand; and into LuaLaTeX documents, which were thereafter compiled into print-ready PDF files destined for digital consumption or for print, on the other hand. The LuaLaTeX version contains indexes of mentioned names and of cited works, manuscripts, and incunabula, as well as a bibliography. It is still a work in progress, and this is the reason why it is not fully available to the public. When ready, it will be made available under the CC BY-NC-SA 4.0 license.

## 3 Workflow

We first designed a series of XSLT templates [15] corresponding to each TEI-XML element in the source documents.[1] In what follows we will not describe all transformations (the reader can skim the code), but

[1] The precise specification of the TEI documents was inspired by the guidelines described by the Scholastic Commentaries and Texts Archive (SCTA) for diplomatic transcriptions [18, 19]. Since the present document was an annotated translation of a Medieval Latin text, several modifications had to be made. Accordingly, we created ODD [13] and RELAX NG [14] schemas to validate the TEI code.

will refer only to some of the more interesting and challenging.

The templates were used to transform the TEI source documents into LuaLaTeX documents. It is worth mentioning that the decision to use LuaLaTeX rather than PDFLaTeX responded to the need to render Ancient Greek and Hebrew scripts, something not easily possible in the latter. Finally, we used the Java edition of Saxon-HE v.10 [7] to process the XSLT transformations to render LuaLaTeX code, which was thereafter compiled into PDF files.

The main template contains the core of the LuaLaTeX document:

```
<xsl:template match="/">
  \def\SMVersion{<xsl:value-of
    select="$combinedversionnumber"/>}
  \input{smpreamble.tex}
  \begin{document}
    \input{frontmatter.tex}
    \mainmatter
    \pagestyle{smtrad}
    \chapterstyle{smtrad}
    <xsl:apply-templates select="//body"/>
    \input{backmatter.tex}
  \end{document}
</xsl:template>
```

**Listing 1**: Main XSLT template

To keep it simple, we load the preamble and all macros as external documents: `smmacros.tex`, which in turn is included in `smpreamble.tex`. The whole document is built using the `memoir` class [17], with a few additional packages. One such is the `enotez` package [4], needed to customize the endnotes.

The TEI document contains several levels of logical division, determined by nested `<div>` elements:

```
<body>
  <div> <!--part 1-->
    <div> <!--chapter 1-->
      <p>...</p> <!--paragraph 1-->
      ...
    </div>
    ...
  </div>
  ...
</body>
```

**Listing 2**: Logical divisions in a TEI document

In XML the fact that one `<div>` element is the child of another `<div>` element is sufficient to establish their logical relationship. In LaTeX, by contrast, logical divisions are determined by the different partitioning commands: `\part`, `\chapter`, `\section`, and so on.

Nevertheless, to facilitate the transformation, we decided to make use of the TEI attribute `@ana` in the `<div>` elements. Accordingly, a `<div>` with `@ana="level1"` is transformed into a LaTeX `\part`; one with `@ana="level2"`, into a LaTeX `\chapter`; and one with no `@ana` into a `\section`. Hence, we created the following XSLT template to transform parts and chapters accordingly:

```
<xsl:template match="div[child::head]">                 1
  <xsl:text>&#10;&#10;</xsl:text>                        2
  <xsl:choose>                                           3
    <xsl:when test="head[@ana='level1']">                4
      <xsl:text>\part</xsl:text>                         5
    </xsl:when>                                          6
    <xsl:when test="head[@ana='level2']">                7
      <xsl:text>\chapter</xsl:text>                      8
    </xsl:when>                                          9
    <xsl:otherwise>                                      10
      <xsl:text>\section</xsl:text>                      11
    </xsl:otherwise>                                     12
  </xsl:choose>                                          13
  <xsl:text>{</xsl:text>                                 14
  <xsl:copy-of select="head"/>                           15
  <xsl:text>}</xsl:text>                                 16
  <xsl:if test="@xml:id">                                17
    <xsl:text>\label{</xsl:text>                         18
    <xsl:value-of select="@xml:id"/>                     19
    <xsl:text>}</xsl:text>                               20
  </xsl:if>                                              21
  <xsl:text>&#10;&#10;</xsl:text>                        22
  <xsl:apply-templates/>                                 23
</xsl:template>                                          24
```

**Listing 3**: `<div>` sectioning template

Line 15 copies the content of the `<head>` element which is a child of the `<div>`, and uses it as a title for the division. In addition, lines 2 and 22 introduce two line feed characters (Unicode U+0010) in the resulting LaTeX document. This is required both for readability and to create a LaTeX line break. In other templates (e.g., that in Listing 4 below) we simply left a blank line, mainly for readability of the XSLT code.

Paragraph elements (`<p>` in our TEI document) are transformed into LaTeX paragraphs seamlessly, always taking care to leave a line break before and after them. Some paragraphs, however, are treated differently. For some parts of the text in the appendixes it was necessary to create pseudo-headers which were horizontally centred. These were encoded using the TEI attributes `@ana="h1"` and `@ana="h2"` for a `<p>` element. In addition, some non-indented paragraphs were also needed, which were encoded using the attribute `@rend="indented"`.[2] The complete template for `<p>` elements is the following:

```
1  <xsl:template match="p">
2    <xsl:if test="@xml:id">
3      <xsl:text>\label{</xsl:text>
4      <xsl:value-of select="@xml:id"/>
5      <xsl:text>}</xsl:text>
6    </xsl:if>
7    <xsl:choose>
8      <xsl:when test="@ana='h1' or @ana='h2'">
9        <xsl:text>
10         \begin{adjustwidth}{.2\textwidth}%
11           {.2\textwidth}
12           \begin{center}</xsl:text>
13             <xsl:if test="@ana='h1'">
14             <xsl:text>\large{}</xsl:text>
15             </xsl:if>
16             <xsl:apply-templates/>
17             <xsl:text>\end{center}
18         \end{adjustwidth}
19
20         \bigskip
21       </xsl:text>
22     </xsl:when>
23     <!--indented paragraph-->
24     <xsl:when test="@rend='indented'">
25       <xsl:text>\begin{indentedpar}</xsl:text>
26         <xsl:apply-templates/>
27         <xsl:text>\end{indentedpar}</xsl:text>
28     </xsl:when>
29     <!-- -->
30     <xsl:otherwise>
31       <xsl:apply-templates/>
32     </xsl:otherwise>
33   </xsl:choose>
34 </xsl:template>
```

**Listing 4**: `<p>` template

As can be seen in lines 24–28, a TEI indented paragraph is transformed into a LaTeX `indentedpar` environment, defined like this in the `smmacros.tex` file:

```
\NewDocumentEnvironment{indentedpar}{}
{\begin{list}{}%
   {\setlength\rightmargin{28pt}%
     \setlength\leftmargin{28pt}}%
 \item[]\small\ignorespaces}
 {\end{list}}
```

**Listing 5**: `indentedpar` environment

[2] According to the TEI P5 Guidelines [11], `@ana` is used to provide an analysis of the element containing it, whereas `@rend` describes the way it is actually rendered.

This illustrates the issue mentioned earlier — where should the piecemeal processing occur. In this case, both the XSLT and the LaTeX code perform part of the processing necessary for producing and typesetting indented paragraphs. Such an ugly "language promiscuity" — so to speak — is exacerbated in the case of citations, quotes, and references. This is mainly because we found it necessary, in the TEI document, to distinguish quotes in several languages: Latin, Spanish, English, Middle English, Anglo Saxon, French, Middle French, Old French, German, Classical Greek, Welsh, Italian, and Hebrew. The XML attribute `@xml:lang`, common to all elements, makes this seemingly easy, for instance:

```
<q xml:lang="fra">
  Ceci n'est pas une pipe.
</q>
```

We follow the ISO 639-3 [8] standard to encode language names: `fra` for current French, `fro` for Old French, `enm` for Middle English, and so on. The problem is that LaTeX's `babel` package uses other language names.[3] So the transformation could not proceed as easily as merely taking the value of `@xml:lang` and using it as an argument to `babel`'s `otherlanguage*` environment or commands to that effect. First we had to translate the ISO codes into `babel` codes using a special function in XSLT:

```
<xsl:template name="my:lang">
  <xsl:param name="lname"/>
  <xsl:choose>
    <xsl:when test="$lname='ang'">english</xsl:when>
    <xsl:when test="$lname='enm'">english</xsl:when>
    <xsl:when test="$lname='eng'">english</xsl:when>
    <xsl:when test="$lname='lat'">latin</xsl:when>
    <xsl:when test="$lname='es'">spanish</xsl:when>
    <xsl:when test="$lname='fro'">french</xsl:when>
    <xsl:when test="$lname='frm'">french</xsl:when>
    <xsl:when test="$lname='fra'">french</xsl:when>
    <xsl:when test="$lname='grc'">greek</xsl:when>
    <xsl:when test="$lname='cym'">welsh</xsl:when>
    <xsl:when test="$lname='deu'">german</xsl:when>
    <xsl:otherwise>english</xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:function name="my:lang" as="xs:string">
  <xsl:param name="lname"/>
  <xsl:call-template name="my:lang">
    <xsl:with-param name="lname" select="$lname"/>
  </xsl:call-template>
</xsl:function>
```

**Listing 6**: Language code names conversion

[3] After my TUG 2021 presentation, I have been advised — by Frank Mittelbach and others, to all of whom I am grateful — that Babel now supports ISO language codes. This should simplify the XSLT processing here described.

Our original plan was to code this algorithm (and perhaps others) using either pure (LA)TEX commands or Lua functions within a LuaLATEX document. However, this proved to be more difficult than expected,[4] and thus we defaulted to performing the processing both in XSLT and in LATEX.

We next created the XSLT template to transform `<foreign>`, `<mentioned>`, and `<gloss>` elements, as well as standalone `<q>` elements:

```
 1 <xsl:template match="foreign | mentioned | gloss
 2     | q[not(parent::cit)]">
 3   <xsl:text>\MyQ</xsl:text>
 4   <!--insert lang-->
 5   <xsl:text>{</xsl:text>
 6   <xsl:choose>
 7     <xsl:when test="@xml:lang
 8            or name(.)='foreign'">
 9       <xsl:sequence
10         select="my:lang(@xml:lang)"/>
11     </xsl:when>
12     <xsl:otherwise>
13       <xsl:text>spanish</xsl:text>
14     </xsl:otherwise>
15   </xsl:choose>
16   <xsl:text>}</xsl:text>
17   <!--insert label-->
18   <xsl:text>{</xsl:text>
19   <xsl:if test="@xml:id">
20     <xsl:value-of select="@xml:id"/>
21   </xsl:if>
22   <xsl:text>}</xsl:text>
23   <!--insert text-->
24   <xsl:text>{</xsl:text>
25   <xsl:if test="@ana='lexeme'">
26     <xsl:text>\lexquote{</xsl:text>
27   </xsl:if>
28   <xsl:apply-templates/>
29   <xsl:if test="@ana='lexeme'">
30     <xsl:text>}</xsl:text>
31   </xsl:if>
32   <xsl:text>}</xsl:text>
33 </xsl:template>
```

**Listing 7**: Language code names conversion

Lines 7–12 call the previous function (Listing 6) and introduce the `babel`-compliant language code. If no language code is specified, the template defaults to Spanish. Line 3 produces a LATEX command, `\MyQ`, which sorts out the language of the quote, defined as follows:

[4] See the `tex.stackexchange.org` discussions [9, 10].

```
\NewDocumentCommand{\MyQ}{m m +m}
{%
  % check for label
  \ifstrempty{#2}{\relax}{\label{#2}}%
  % check for language
  {%
    \ifstrempty{#1}%
    {\begin{otherlanguage*}{spanish}}%
      % else
      {\begin{otherlanguage*}{#1}}%
        % if spanish
        \ifstrequal{#1}{spanish}%
        % then
        {\enquote{#3}}%
        % else
        {%
          \ifstrequal{#1}{greek}%
          {#3}%
          {\textit{#3}}%
        }%
      \end{otherlanguage*}%
    }%
  }
```

**Listing 8**: LATEX code for handling quotations

Handling of citations (which comprise a quote in the original language, an optional quote of the Spanish translation, and a bibliographic reference) is performed in a similar fashion by three different XSLT templates and a series of LATEX commands.

More interesting, however, is the code used to handle cross-references, both internal and external. We decided to take advantage of the `@type` attribute of TEI's `<ref>` element, which allows for twelve kinds of references, to wit:

| **Value** | **Meaning** |
|---|---|
| `a` | `\citeauthor` |
| `p` | `\parencite` |
| `t` | `\citetitle` |
| `y` | `\citeyear` |
| `py` | `\citeyear` in parenthesis |
| `abbr` | `\citeabbr` |
| `pabbr` | `\citeabbr` in parenthesis |
| `abbrpc` | `\citeabbr` (content) |
| `fulltext` | use `text/rend` without calling `\cite` |
| `nc` | `\nocite` |
| `pnc` | `\nocite` in parenthesis |
| `url` | `\href` |

**Table 1**: Values for `//ref/@type`

For instance, to cite a work from the bibliography using a predefined abbreviation and putting the "content" of the citation inside parentheses (in the `@rend` attribute), the TEI code will be the following:

```
<cit>
  <ref target="#DMLBS" type="abbrpc"
       rend="s.v. 2"/>
</cit>,
```

**Listing 9**: Example of citation with reference

Some of the twelve possible references are handled by different XSLT templates. For the case chosen above (`@type="abbrpc"`), the code is as follows:

```
<xsl:template match="ref[@type='abbrpc']"
              priority="2">
  <xsl:text>\citeabbr{</xsl:text>
  <xsl:value-of select="my:cleanref(@target)"/>
  <xsl:text>}</xsl:text>
  <!--insert rend/text-->
  <xsl:text>\space(</xsl:text>
  <xsl:choose>
    <!--select @rend if present-->
    <xsl:when test="@rend">
      <xsl:value-of select="@rend"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>)</xsl:text>
</xsl:template>
```

**Listing 10**: Template for `//ref[@type="abbrpc"]`

As can be seen, we have taken full advantage of `biblatex`'s citation commands, so all processing is performed by LaTeX behind the scenes. This feature, as well as automatic indexes and nomenclatures, justifies our choice of (Lua)LaTeX within the TEI → PDF workflow.

## 4 Whitespace

One of the tougher problems when dealing with XML and related languages concerns whitespace characters. XML always combines multiple whitespace characters, collapsing them into a single space character. This is sometimes useful, but can create undesirable results when processing TEI documents. Take, for instance, this sample TEI code:

```
<p>
  Some text here that will occupy a couple of
  lines in the main body of the final document.
</p>
<note>
  A textual note for the previous paragraph.
</note>
```

**Listing 11**: Example of TEI whitespace

Here, indentation and line breaks are needed only for human readability, not for TEI-XML validity. However, since our TEI document was digitally born — i.e., not converted from other sources — and all editorial work is done directly with it, it is imperative that the authors, editors, copy-editors, and other people working with the text can work with it with ease. In short, we did not want to sacrifice code legibility.

The consequent problem is that without further processing and cleaning, the former code will be rendered with an additional space (marked here as a visible space) between the paragraph and the endnote anchor:

> …in the main body text of the finished document.␣$^1$

**Figure 1**: Example of rendered text

For typographical reasons, such insertion of whitespace characters is highly undesirable. XSLT can deal with some of these issues, but only with limited success. We used the following code:

```
<xsl:template match="text()">
  <xsl:value-of
    select="replace(., '\s+', ' ')"/>
</xsl:template>
<xsl:function name="my:cleantext">
  <xsl:param name="input"/>
  <xsl:variable name="step1"
    select="replace($input, '\n+', ' ')"/>
  <xsl:variable name="step2"
    select="normalize-space($step1)"/>
  <xsl:sequence select="$step2"/>
</xsl:function>
```

**Listing 12**: XSLT code for dealing with whitespace

Nonetheless, many spaces still remained in the transformation. Thus, an additional processing step had to be introduced in our workflow, this time in Python [12]. The script we devised is a collection of regex search-and-replace commands such as the following:[5]

```
# Remove trailing whitespace at closing paren
(r" +\)", r")"),
# Remove redundant space before \endnote.
(r"\s*(\\endnote)", r"%\n\n\1"),
# Break line after colon (but skip citations)
(r": +(?!\d)", r":\n"),
```

**Listing 13**: Python code for dealing with whitespace

[5] More exactly, these are tuples in a list which is then processed with a `re.sub(cpattern, replacement, buffer)` function.

After running the script, unnecessary whitespace is successfully pruned from the output LaTeX document.

## 5 Conclusions

In the end, the workflow consists of the following steps:

TEI-XML → XSLT → Python → LuaLaTeX → PDF.

To be sure, some of the additional processing — concerned mainly with the transformation of TEI elements into LaTeX commands — might be realised in Lua (within LuaLaTeX). However, it is unlikely that the whitespace problems can be solved without the help of a regex solution (in Python, sed, etc.), as it concerns the passage from one stage to another in the workflow.

## References

[1] Apache FOP Compliance Page. `https://xmlgraphics.apache.org/fop/compliance.html`

[2] A. Berglund, ed. Extensible Stylesheet Language (XSL) Version 1.1. W3C Recommendation 05 December 2006. `https://www.w3.org/TR/xsl11/`

[3] E.J. Etemad, S. Sapin, eds. CSS Paged Media Module Level 3. W3C Working Draft, 18 October 2018. `https://www.w3.org/TR/css-page-3/`

[4] C. Niederberger. Enotez—Support for end-notes. `https://ctan.org/pkg/enotez`

[5] PDFreactor. `https://www.pdfreactor.com/`

[6] PrinceXML. `https://www.princexml.com/`

[7] Saxon XSLT and XQuery Processor. `https://sourceforge.net/projects/saxon/files/`

[8] SIL. ISO 639-3. `https://iso639-3.sil.org/`

[9] StackExchange - Problem using macros in directua. `https://tex.stackexchange.com/questions/556911/`

[10] StackExchange - problem with command expansion. `https://tex.stackexchange.com/questions/557079/`

[11] TEI Consortium. TEI P5: Guidelines for Electronic Text Encoding and Interchange. `https://tei-c.org/release/doc/tei-p5-doc/en/html`

[12] N. Vaughan. Cleantex - Python script to remove unwanted whitespace from LaTeX files produced from TEI. `https://github.com/nivaca/cleantex`

[13] N. Vaughan. SM-ODD. ODD schema for the Salomon & Marcolfus project. `https://github.com/nivaca/sm-odd`

[14] N. Vaughan. SM-RELAXNG schema for the Salomon & Marcolfus project. `https://github.com/nivaca/sm-relaxng`

[15] N. Vaughan. XSLT-LaTeX - templates to transform TEI-XML documents into LaTeX. `https://github.com/nivaca/xslt-latex`

[16] WeasyPrint. `https://weasyprint.org/`

[17] P.R. Wilson, L. Madsen. Memoir – Typeset fiction, non-fiction and mathematical books. `https://ctan.org/pkg/memoir`

[18] J.C. Witt, M. Stenskjær, N. Vaughan. Lombard Press Schema 1.0.0 - Diplomatic Transcription Guidelines. `https://community.scta.info/pages/lombardpress-schema-diplomatic.html`

[19] J.C. Witt, M. Stenskjær, N. Vaughan. LombardPress Schema. `https://github.com/lombardpress/lombardpress-schema`

[20] Wkhtmltopdf. `https://wkhtmltopdf.org`

◇ Nicolás Vaughan
Departamento de Literatura
Universidad de los Andes
Bogotá, Colombia
`n.vaughan (at) uniandes.edu.co`
`http://nicolasvaughan.org`
ORCID 0000-0002-2877-0539

# LaTeX News

Issue 33, June 2021

## *Contents*

## *Introduction*

The focus of the June 2021 release is to provide further important building blocks for the future production of reliable tagged PDF output (see [1]); these enhancements are discussed in the next two sections.

Subsequent sections describe quite a number of recent smaller enhancements and fixes. As usual, more detail on individual changes can be found in the `changes.txt` files in the distribution and, of course, in the documented sources [2].

## *Extending the hook concept to paragraphs*

Largely triggered by the need for better control of paragraph text processing, in particular when producing tagged PDF output, we have changed LaTeX so that the kernel gains control both at the start and at the end of each paragraph. This is done in a manner that is (or should be) transparent to both packages and documents.

Besides the addition of internal control points for the exclusive use of the LaTeX kernel, we also implemented four public hooks that can be used in packages or documents (via the normal hook management declarations) to achieve special effects, etc. Until now, such enhancements required redefinitions of `\everypar` or `\par`, which led to the usual issues since such changes can easily conflict with changes made by other packages.

The documentation of these new “paragraph hooks”, together with a few examples, is in `ltpara-doc.pdf` and, for those who want to study it, the (quite interesting) code can be found in `ltpara-code.pdf`. Additionally, both of these files are included as part of the full kernel documentation in `source2e.pdf`.

## *Extending the hook concept to commands*

Up to now, hook management covered hooks for only a few core areas, such as the hooks for the `\shipout` process or those in the `document` environment, as well as some "generic" hooks, both for file loading (helpful for patching such files) and for arbitrary environments (the hooks executed within `\begin` and `\end`). This concept of "generic hooks" has now been extended to provide `/before` and `/after` hooks for any (document-level) command—in theory at least.

In practice, these new generic `cmd` hooks, especially the `cmd/.../after`, hooks may fail with commands that are too complex to be automatically patched, breaking if the hook contains any code. These restrictions are documented in `ltcmdhooks-doc.pdf`. However, given that these hooks are mainly meant for developers who wish to provide better interoperability between different packages, and between packages and the LaTeX kernel, these restrictions are, we hope, of minor importance. Indeed, for commands where this mechanism can't be applied, one is in the same situation as before; and for all others there will be a noticeable improvement.

These hooks will be especially important for our current project to provide accessible and tagged PDF output [1] because we will eventually have to patch many third-party packages, and this must be done in controlled and standardized ways.

## Other hook business

### Shipping out a page while bypassing hooks

In the 2020 October release, several hooks were added to control the process of constructing and shipping out a page box: these support, for example, the addition of background or foreground material to some or all pages.

We have now added a command, called `\RawShipout`, which does not do any rebuilding of the page box and so does not run most of these hooks. When using this new command, essential internal book-keeping is still carried out, such as updating the `totalpages` counter and adding `shipout/firstpage` or `shipout/lastpage` material when appropriate.

### A new Lua callback in ltshipout, for custom attributes

For use just before shipping out a page, there is now a LuaTeX callback `pre_shipout_filter` to contain final adjustments to the box being shipped out. This is particularly useful for LuaTeX packages which flag (using, for example, attributes or properties) elements on a page in order to apply effects (such as the insertion of "color commands") to these elements at shipout.

## Improved handling of file names

### File names with spaces, multiple dots or UTF-8 characters

In one of the recent LaTeX releases we improved the interface for specifying file names so that they can now safely contain spaces (as is common these days), more than one dot character, and also UTF-8 characters outside the ASCII range. In the past this was only possible by applying a special syntax in the case of spaces, while file names with several dots often failed, as did most UTF-8 characters.

*Consequences for file names in `\include`:* TeX has a built-in rule saying that you can normally leave out the extension if it is `.tex`. Thus `\input{file}` and `\input{file.tex}` both load `file.tex` (if it exists). While this is convenient most of the time, it is a little awkward in some scenarios (for example, when both `file` and `file.tex` exist) and also when you manually try to implement the rule.

LaTeX therefore had one special syntax for `\include` and `\includeonly`: they always expected that their arguments contain a file name[1] with no extension given, so that it had to be `.tex`. Thus, when you mistakenly wrote `\include{mychap.tex}` (for example, because you changed from `\input` to `\include`), LaTeX went ahead and looked for the file `mychap.tex.tex` for inclusion and tried to use the file `mychap.tex.aux` for internal (auxiliary) information. The reason was that `\include` had to construct both of these file names from the given argument and it didn't bother to do anything special with the supplied extension `.tex`.

With the new implementation this has changed: the extension `.tex` now gets removed/ignored if it was supplied. Thus `\include{mychap.tex}` now no longer looks for `mychap.tex.tex` but loads `mychap.tex` and uses `mychap.aux`. *(github issue 486)*

### Normalization of robust commands in file names

The handling of file names has been modified so that `\string` is applied to normalize robust commands within the file name. Previously, for example, `\input{\sqrt{2}}` would cause LaTeX to loop indefinitely whereas with the new normalization it looks for the file named `sqrt {2}.tex` (and therefore very likely reports "file not found"). *(github issue 481)*

### Fix for `filecontents` with UTF-8 chars in the file name

Since a few releases back, the `filecontents` environment has allowed UTF-8 characters in the file name. There was, however, a bug that would not allow *over*writing a file with UTF-8 characters in its name. This has been fixed and now `filecontents` allows any characters in the file name. *(github issue 415)*

## Updates to the font selection scheme

### A new hook in `\selectfont`

After `\selectfont` has changed the font, we now run a hook (`selectfont`) so that packages can make final adjustments. This functionality was originally provided by the everysel package but our implementation is slightly different and uses the standard hook management. *(github issue 444)*

### Change of font series/shape delayed until `\selectfont`

With the NFSS extensions introduced in 2020, the font series and shape settings can be influenced by changes to the font family. The settings of these two are now therefore delayed until `\selectfont` is executed; this

[1] In the case of `\includeonly`, a comma-separated list of such names.

avoids unnecessary or incorrect substitutions that may otherwise happen due to the order of declarations.
*(github issue 444)*

## *Glyphs, characters & encodings*

### *Improved copy & paste for pdfTEX documents*

When compiling with pdfTEX, additional information (from the file `glyphtounicode.tex`) is now added automatically to the PDF file in order to improve copying from, and searching in, text.

In particular, this allows the most common ligatures to be copied as intended from all generated PDF files without the need to explicitly load the package `cmap`.
*(github issue 465)*

### *Support for more Unicode characters*

LATEX is quite capable of typesetting characters such as "ṃ", but until now it could not access some Unicode characters from the Latin Extended Additional block. This meant that, for example, there were no Unicode mappings for some characters that are used to write Sanskrit words in Latin transliteration (as seen in books about yoga, Buddhist philosophy, etc.). These characters have now been added so that they can be entered directly instead of using `\d{m}`, etc. *(github issue 484)*

### *More "dashes" in encodings OT1, T1 and TU*

When pasting in text from external sources, one can encounter these three Unicode characters `"2011` (non-breaking hyphen), `"2012` (figure dash) and `"2015` (horizontal bar), in addition to the more common `"2013` (en-dash) and `"2014` (em-dash). In the past, these first three produced an error message when used with pdfTEX (since they are not available in `OT1` or `T1` encoded fonts). They now typeset an approximation to the glyph: e.g., the "figure dash" is approximated by an en-dash.

With Unicode engines they either work (when the glyph is contained in the selected Unicode font) or they typeset nothing, producing a "Missing character" warning in the log file.

With all engines these characters can also now be accessed using the command names `\textnonbreakinghyphen`, `\textfiguredash` and `\texthorizontalbar`, respectively. *(github issue 404)*

### *Poor man's `\textasteriskcentered`*

The `\textasteriskcentered` symbol, used as part of the set of footnote symbols in LATEX, is assumed to be implemented by every font with the `TS1` encoding (when pdfTEX is used) or with the `TU` encoding for the Unicode engines. That assumption is unfortunately not correct for all fonts since, for example, the `stix2` fonts don't provide this glyph. A result is that one gets missing glyph messages when using `\thanks`, etc.

Therefore `\textasteriskcentered` now checks whether there is such a glyph and, if not, uses a normal "*", but slightly enlarged and lowered. This may not be perfect in all cases, but it is certainly better than no glyph showing up. *(github issue 502)*

### *The characters from textcomp are in the kernel*

A couple of releases back, the functionality of the `textcomp` package was integrated into the LATEX kernel. Thus it is no longer necessary to load this package in order to access glyphs such as `\textcopyright`, `\texteuro` or `\textyen`.

At this time the opportunity was also taken to bring some order to the chaos surrounding the question: "which glyphs from the `TS1` encoding are available in a given font?". This was done using an approach based on font families and collections, with the differing glyph coverage of the 'text symbols' being indicated by assigning to a font family or collection a "sub-encoding number" that indicates which glyphs from the `TS1` encoding are guaranteed to be available when using a font from that family or collection. This assignment ensures that LATEX always errs on the side of caution, possibly claiming that a glyph is not available even when it in fact is.

*A note on the history of "text symbols" and the `TS1` encoding:* The "text symbol encoding" (`TS1`) was originally designed at the Cork Conference as a companion to the `T1` encoding. In it various symbols that are not subject to hyphenation got assembled and the `textcomp` package was developed to make them accessible. Unfortunately the TEX community was a bit too enthusiastic and included several symbols only available in a few TEX fonts and some, such as the capital accents, not available at all but developed as part of the reference font implementation.

In hindsight that was a very bad idea because it meant that other existing fonts (at the time) and later new fonts that got developed were unable to provide the full set of glyphs that made up the `TS1` encoding. For existing free PostScript fonts people took the extra effort and produced virtual fonts that faked (some) of the missing glyphs. But this was and is a time-consuming effort so it was done for only a few basic fonts. But even then, only some fonts included all glyphs from `TS1` so the `textcomp` already back then contained a long list, dividing fonts into 5 categories according to which glyphs were implemented and which were missing.

When we recently integrated the functionality of the `textcomp` into the LATEX kernel many new free fonts had appeared and unfortunately the chaos around the question "which glyphs of the `TS1` encoding are implemented by which font" had increased with it. Not only did one find many new holes, it was next to impossible to order the set of fonts into a reasonable set of sub-encodings that are contained in each other in a single sequence.

In the end we decided on nine or ten sub-encodings with a reasonable number of fonts in each so that all fonts implemented all glyphs of the sub-encoding they got mapped to. Thus when typesetting with a font one could be sure that a command like `\textcopyleft` would either typeset the requested character (if the glyph was part of the sub-encoding the font belonged to) or it would raise an error, saying that the glyph is unavailable in that font. The mapping would ensure that LaTeX always errs on the side of caution, because it might claim a glyph is unavailable even though in fact it is.

For example, the old `pcr` (PostScript Courier) font (as well as most other older PS fonts) is mapped to sub-encoding 5 and therefore claims that `\textasciigrave` is unavailable even though in fact for Courier this is not true. If one uses such a font and this becomes an issue then there are a couple (suboptimal) possibilities. For one, one can alter the mapping of Courier and pretend that belongs to a fuller sub-encoding, e.g.

```
\DeclareEncodingSubset{TS1}{pcr}{2}
```

The downside is, that LaTeX then believes other glyphs that are in fact unavailable are also there, so that it is important to check that the final document doesn't have some missing glyphs.

An alternative is to pretend that `\textasciigrave` can always be taken from the `TS1` encoding (no questions asked):

```
\DeclareTextSymbolDefault{\textasciigrave}{TS1}
```

Again there is a danger that this is not true when it is used with a different font and would then generate a missing glyph.

Finally, and possibly the best solution, if not impossible for other reasons, is to simply use a different font, for example, to use the TeX Gyre Cursor font (a reimplementation of Courier with a much more complete glyph set).

## New or improved commands

### Adjusting `itemize` labels with `\labelitemfont`

The command `\labelitemfont` was introduced already with the LaTeX release 2020-02-02, but back then we forgot to describe it, so we do this now. Its purpose is to resolve some bad formatting issues with the `itemize` environment and also to make it easier to adjust the layout when necessary. What could happen in the past was that the `itemize` labels (e.g., the •) would sometimes react to surrounding font changes and could then suddenly change shape, for example to *•*.

This new command `\labelitemfont`, which defaults to `\normalfont`, can be used to provide additional control in the typesetting of each label. Thus by choosing different settings other effects can be achieved. Here are two examples:

```
\renewcommand\labelitemfont
   {\normalfont\fontfamily{lmss}\selectfont}
\renewcommand\labelitemfont
   {\rmfamily\normalshape}
```

The first definition will take the symbols from the font Latin Modern Sans, so that you get ▪, –, ∗ and · ; while the second variant freezes the font family and shape, but leaves the series as a variable quantity, so that an `itemize` in a bold context would show bolder symbols. Making `\labelitemfont` empty would give you back the buggy old behavior. *(github issue 497)*

### Producing several marks for one footnote

It is sometimes necessary to reference the same footnote several times: i.e., to produce several footnote marks using the same number or symbol. This is now easily possible by placing a `\label` within the referenced `\footnote` and referencing this label by using the new command `\footref`. This means that footnote marks can be generated to refer to arbitrary footnotes (including those in `minipage`s).

This `\footref` command has previously been available, but only when using certain classes or the footmisc package. *(github issue 482)*

### Allow `\nocite` in the preamble

A natural place for `\nocite{*}` would be the preamble of the document, but for historical reasons LaTeX issued an error message if it was placed there. This command is now allowed in the preamble. *(github issue 424)*

### Made `\\` generally robust

In 2018 most LaTeX user-level commands were made robust, including the `\\` command. However, `\\` gets redefined in various environments and not all these cases were caught: such as, in particular, its use as the row delimiter in `tabular` structures. This has been corrected so that `\\` should now be robust in all circumstances.

This change also fixed one anomaly present in the past: in a tabular preamble of the form

```
{l>{\raggedright}p{10cm}r}
```

a `\\` in the second column would have the definition used within `\raggedright` and so it would not indicate the (premature) end of the `tabular`. Thus, for example,

```
a & b1 \\ b2 & c \\
```

was interpreted as a single row of the `tabular` (as intended), whereas

```
a &    \\ b2 & c \\
```

resulted in two rows! This happened because the `\\` directly following the `&` got interpreted while it still had the "end the row" meaning and not yet the "start a new line within the second column" meaning.

With `\\` now being robust, the special scanning mode initiated by the `&` ends immediately when this command is seen: the second column is therefore then started, which results in the `\\` being interpreted as being within that column and hence as having its expected, within-column, meaning.

We have restored consistency here: now both of the above lines produce a single `tabular` row. As before, you can put `\raggedright\arraybackslash` in the `tabular`'s preamble for a column to ensure that `\\` is always interpreted as a tabular row separator when used in that column. And you can use `\tabularnewline` to explicitly ask for a new table row, even when `\\` has a different meaning within the current column. *(github issue 548)*

#### Allow extra space between name and address in letter class

The `\opening` command in the letter class expects the name and address to be separated by `\\`, but it didn't allow the use of an optional argument to add some extra space after the name. The code has now been slightly altered to allow this. *(github issue 427)*

#### Additions to `\tracingall`

In July 2020 David Jones suggested an extension to TeX engines, that added the possibility to set `\tracinglostchars=3` in order to generate an error message in case some character is missing from a font. In previous years, a warning about a missing character was silently printed to the `.log` file (if `\tracinglostchars` $> 0$) and to the terminal (if $> 1$). This extension was added for TeX Live and MiKTeX (except in Knuth's TeX, of course), so that with `\tracinglostchars` $> 2$ you now also get an error message for each missing glyph.

Later, in January 2021, Petr Olšák suggested yet another extension: a new primitive parameter `\tracingstacklevels` that, when both it and `\tracingmacros` are positive, will add to the tracing information for each macro a visual indication (using dots) of its nesting level in the macro expansion stack.

These changes have both now been added to LaTeX's debugging macros `\tracingall` and `\tracingnone`, so that these two new extensions are activated/deactivated as appropriate, so long as the TeX engine supports them. An example document demonstrating these parameters is in the linked GitHub issue. *(github issue 524)*

## Code improvements

#### Execute `\par` at the end of `\marginpar`

Previously, LaTeX ended a `\marginpar` without ever explicitly calling `\par`. This command is now explicitly added because it is essential to the correct working of the paragraph hooks.

Another case where this issue caused problems was the lineno package, where the last line was not numbered if the `\marginpar` ended without an explicit `\par`. *(github issue 489)*

#### Execute `\AtEndDocument` hook in vertical mode

Until now `\end{document}` executed the code from the `\AtEndDocument` hook as its first action. This meant that this hook was executed in horizontal mode if the user left no empty line after the last paragraph. As a result, one could get a spurious space added when, for example, that code contained a `\write` statement. This was fixed and now `\enddocument` first issues a `\par` to ensure that it always goes into vertical mode. *(github issue 385)*

#### Color groups made permanent

The use of color in certain LaTeX constructs, especially boxes, needs an extra layer of grouping to ensure that the color setting does not *escape* and continue outside the box when it shouldn't. To support this, the LaTeX kernel defines a number of commands, e.g., `\color@begingroup` to be used in such places.

Until now, these commands were initially set as no-ops and only the color packages redefined them to become real groups; this methodology complicates the coding as one has to account for a group being present or not (depending on what is loaded in the document). The kernel therefore now permanently adds these "color groups". *(github issue 488)*

#### Provide the raw option list to key/value option handlers

Before any further processing of the option list, the original (un-normalized, "raw" and unchanged) list of package or class options is now saved, as `\@raw@opt@...`; this list is not used by the standard option processing code but it is now available for use by extended class/package processing systems. Note that, for compatibility reasons, the standard option processing code has not been changed.

One aspect of this change does affect the standard processing: any tokens to the right of an `=` sign are removed from consideration when constructing the "unused option list". For example, in this release `clip=true` and `clip=false` both contribute `clip` to the list of options that have been used. *(github issue 85)*

#### New for latexrelease: `\NewModuleRelease`

To explain the need for this new feature, we shall consider the following example: in the 2020-10-01 release, LaTeX's new hook management system was added to the kernel (see [3]) and, as with all changes to the kernel, it was added to latexrelease; this made it possible to roll back to a date where this module didn't yet exist, or to roll forward from an older LaTeX release to get the hook management system (by loading the latexrelease package). However, this method of rolling back from a later release to the 2020-10-01 release didn't quite work because it would try to define all the commands from lthooks again; and this would of course result in the expected errors from commands defined with `\newcommand` or (as in lthooks) `\cs_new:Npn`.

To solve such issues, we now provide `\NewModuleRelease` so that completely new modules can be defined using the facilities of latexrelease in such a way that, when rolling back or forward, the system will know whether the code of the new module has to be read or completely ignored. More details on this can be found in the latexrelease documentation (get this with `texdoc latexrelease`). *(github issue 479)*

#### Small fix for rolling back prior to 2020-02-02

Whereas the latexrelease package can usually emulate an older LaTeX kernel without much problem, rolling back to before the 2020-02-02 release didn't work properly: this is because the management of the `\ExplSyntaxOn/Off` status for packages (after an expl3-based package is loaded) cannot be removed by the rollback without messing up the catcodes. This has been fixed so that rollback is now more careful not to leave `\ExplSyntaxOn` after a package ends. *(github issue 504)*

## Changes to packages in the graphics category

### Removed warning when loading graphics files

A previous release sometimes mistakenly caused a (false) warning message to appear when using a generic graphics rule to find and load a graphics file with an unknown extension. This warning would incorrectly say that the file was not found, whereas the file would in fact be correctly loaded. The warning now doesn't show up in that case. *(github issue 516)*

### Fixed loading of `gzipped` PostScript files

A previous release mistakenly changed the file searching mechanism so that compressed PostScript graphics files would raise an error when being loaded with `\includegraphics`. This has been fixed so that `gzip`ped graphics files now load correctly. *(github issue 519)*

## Changes to packages in the tools category

### layout: Added language options

This package now recognizes `japanese` and `romanian` as language options. *(github issues 353 and 529)*

### array and longtable: Make `\\` generally robust

The fix for this issue was also applied to these packages; see above. *(github issue 548)*

### longtable: General bug fix update

This is a minor update to the longtable package that fixes several reported bugs: notably the possibility of incorrect page breaks when floats appear on the page where a `longtable` starts. As this may affect page breaking in existing documents, a rollback to longtable 4.13 (`longtable-2020-01-07.sty`) is supported.

*(gnats issue tools/2914 3396 3512)*
*(github issue 133 137 183 464 561)*

### trace: Additions to `\traceon`

The `\tracingstacklevels` and `\tracinglostchars` extensions to `\tracingall` (see above) were also added to `\traceon` in the trace package, so its users can also benefit from these new debugging possibilities. *(github issue 524)*

### bm: Better support for commands with optional arguments

Some uses of optional arguments in `\bm` stopped being supported (in 2004) when `\kernel@ifnextchar` was used internally by the format instead of `\@ifnextchar`. This update handles both versions of this command and restores the original behavior.

In addition, package options for guiding the use of "poor man's bold" in fallback situations were added. *(github issue 554)*

## Changes to packages in the amsmath category

The fix for issue 548 was also applied in amsmath; see above. *(github issue 548)*

## References

[1] Frank Mittelbach and Chris Rowley: *LaTeX Tagged PDF—A blueprint for a large project.* `https://latex-project.org/publications/indexbyyear/2020/`

[2] *LaTeX documentation on the LaTeX Project Website.* `https://latex-project.org/help/documentation/`

[3] LaTeX Project Team: *LaTeX 2ε news 32.* `https://latex-project.org/news/latex2e-news/ltnews32.pdf`

# Markdown 2.10.0: LaTeX themes & snippets, two flavors of comments, and LuaMetaTeX

Vít Novotný

## Abstract

Celebrating its fifth birthday, the Markdown package has received five new features: user-defined LaTeX themes, LaTeX setup snippets, semantic HTML comments, lexical TeX comments, and support for the LuaMetaTeX engine. In this article, we introduce each of these features and show how they can be used in practice. We also discuss five ideas for the future of Markdown and show how you can help turn them into a reality.

## 1 LaTeX themes & snippets

The goal of the Markdown package is simply to bring fire to the users of TeX, so that they can playfully incinerate each and every element of their markdown documents. The Markdown package does not aim to provide comprehensive defaults that would satisfy every kind of a document. Instead, all attention is directed towards making it easy for TeX programmers to style markdown. Although this goal fulfills the Unix philosophy of *doing one thing well*, the Markdown package would be well-served by encouraging users to lecture it on ever-new *things* and release their lecture notes to the whole world.

It is a paradox that one of the greatest successes of the LaTeX project may be its initial lack of features. Unlike in the cathedral of ConTeXt, where packages are few and most development is centralized, an extraordinary bazaar of action, ferment, and innovation has sprung up in the wake of the LaTeX $2_\varepsilon$ kernel. To make it easier for Markdown users to share their lecture notes and combine them into thick tomes of tutelage, version 2.10.0 of the Markdown package has introduced *LaTeX themes & setup snippets.*

In Section 1.1, we introduce three example LaTeX themes that are included with the Markdown package. In Section 1.2, we show how users can create and use their own LaTeX themes. In Section 1.3, we introduce LaTeX setup snippets and show how users can create and use their own setup snippets.

### 1.1 Built-in themes

LaTeX themes are user-defined *building blocks* that

1. specify what markdown elements *do*,
2. can be shared and applied as a single unit, and
3. can be combined with one another to achieve high-level goals without low-level programming.

Since LaTeX is a Turing-complete language, what markdown elements *do* is not restricted to presentation, but includes computation and logic: We may typeset a table in various ways, but we may also store it as a matrix and compute its determinant, inverse, and eigenvalues, or we may apply it to an image as a linear transformation and display the result.

The Markdown package comes with three example LaTeX themes, listed by increasing complexity. These examples should help you develop an intuition for themes and what you can accomplish with them.

#### 1.1.1 The witiko/tilde theme

The witiko/tilde theme redefines the tilde (~), so that it produces a non-breaking space:

```
\documentclass{article}
\usepackage[theme=witiko/tilde]{markdown}
\begin{document}
\begin{markdown}
Bartel~Leendert van~der~Waerden
\end{markdown}
\end{document}
```

The above code will produce the text "Bartel·Leendert van·der·Waerden", where the middle dot (·) represents a non-breaking space.

#### 1.1.2 The witiko/dot theme

The witiko/dot theme renders fenced code blocks with the dot infostring using Graphviz tools:

~~~
\documentclass{article}
\usepackage[theme=witiko/dot]{markdown}
\begin{document}
\begin{markdown}
```dot A parse tree of "Let's eat grandma!"
digraph tree {
    graph [margin = 0]; node [shape = none]
    edge  [arrowhead = none]
    {rank=same; S}
    {rank=same; VP1[label = VP]}
    {rank=same; Let
                NP1[label = NP]
                VP2[label = VP]}
    {rank=same; us; eat; NP2[label = NP]}
    {rank=same; grandma}
    S -> VP1; VP1 -> Let; VP1 -> NP1
    VP1 -> VP2; NP1 -> us; VP2 -> eat
    VP2 -> NP2; NP2 -> grandma }
\end{markdown}
\end{document}
~~~

The above code will produce Figure 1. The size of the graphics as well as other attributes can be controlled with the \setkeys{Gin}{...} command of the graphicx package. The placement of the figure can be controlled by redefining the \fps@figure LaTeX command.

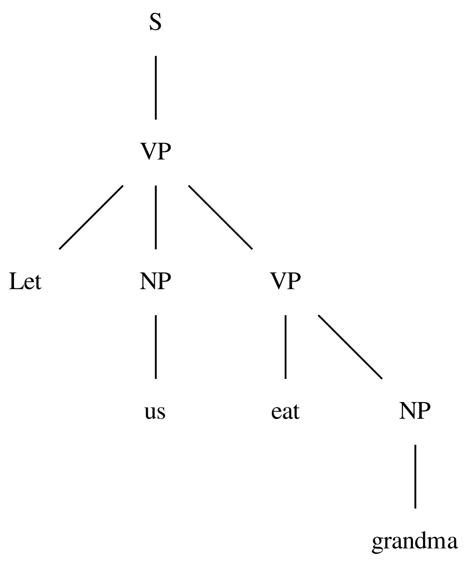
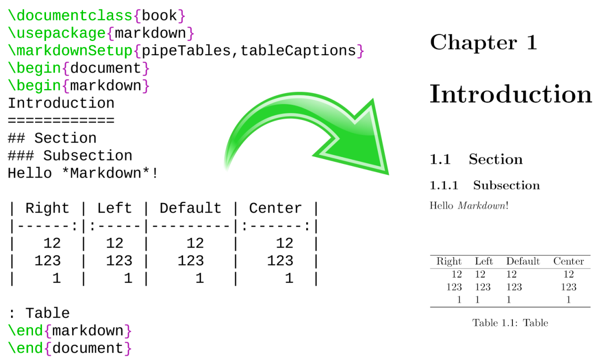


**Figure 1**: A parse tree of “Let’s eat grandma!”



```
\documentclass{book}
\usepackage{markdown}
\markdownSetup{pipeTables,tableCaptions}
\begin{document}
\begin{markdown}
Introduction
============
## Section
### Subsection
Hello *Markdown*!

| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |

: Table
\end{markdown}
\end{document}
```

**Chapter 1**

**Introduction**

**1.1 Section**

**1.1.1 Subsection**

Hello *Markdown*!

| Right | Left | Default | Center |
|---|---|---|---|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

Table 1.1: Table

**Figure 2**: The banner of the Markdown package

#### 1.1.3 The `witiko/graphicx/http` theme

The `witiko/graphicx/http` theme downloads online images using either GNU Wget or cURL, whichever is available on your system, and displays them:

```
\documentclass{article}
\usepackage{markdown}
\markdownSetup{texComments, contentBlocks,
               theme=witiko/graphicx/http}
\begin{document}
\begin{markdown}
https://github.com/witiko/markdown/raw%
                  /main/banner.png
      (The banner of the Markdown package)
\end{markdown}
\end{document}
```

The above code will produce Figure 2 (grayscaled for *TUGboat*). As before, the size and placement of the figure can be controlled using the `\setkeys{Gin}{...}` and `\fps@figure` LaTeX commands.

### 1.2 Creating your own theme

To create your own LaTeX theme, you should decide on a name in the form ⟨*theme author*⟩/⟨*target package*⟩/⟨*private naming*⟩, where ⟨*theme author*⟩ specifies the provenance of the theme, ⟨*target package*⟩ specifies a LaTeX or software package that the theme relates to, and ⟨*private naming*⟩ specifies additional slash-delimited name segments. The ⟨*target package*⟩ and ⟨*private naming*⟩ name segments are optional, but at least one of them must be present.

Let us suppose that Jane Doe wishes to create a simple theme for the Beamer LaTeX package. Beamer creates presentation slides and Jane’s theme will redefine markdown’s first- and second-level headings to typeset the titles and subtitles of presentation slides. Therefore, Jane has decided to name her theme `jdoe/beamer/headings`.

Next, Jane will *munge* the name of the theme by substituting slashes (`/`) with underscores (`_`), and she will attach the prefix `markdowntheme` and the suffix `.sty` to arrive at the following filename:

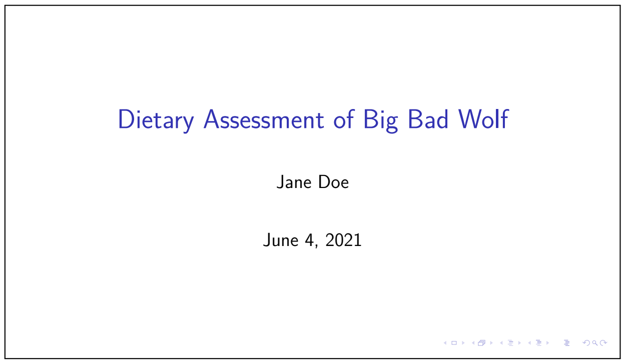
```
markdownthemejdoe_beamer_headings.sty
```

Jane will create a text file with the above filename and the following content:

```
\ProvidesPackage{markdownthemejdoe_beamer_%
                 headings}[2021/06/04]
\markdownSetup{
    rendererPrototypes = {
        headingOne = {\frametitle{#1}},
        headingTwo = {\framesubtitle{#1}}
    }
}
```

Finally, Jane will use her new theme in her presentation slides, together with the `witiko/dot` theme, which she uses to typeset dietary assessments:

```
\documentclass[
    aspectratio=169
]{beamer}
\usepackage[
    theme = witiko/dot,
    theme = jdoe/beamer/headings
]{markdown}
\setkeys{Gin}{
    width=\columnwidth,
    keepaspectratio
}
\title{Dietary Assessment of Big Bad Wolf}
\author{Jane Doe}
\date{June 4, 2021}
\begin{document}
\maketitle
\begin{frame}[fragile]
\begin{markdown}
```



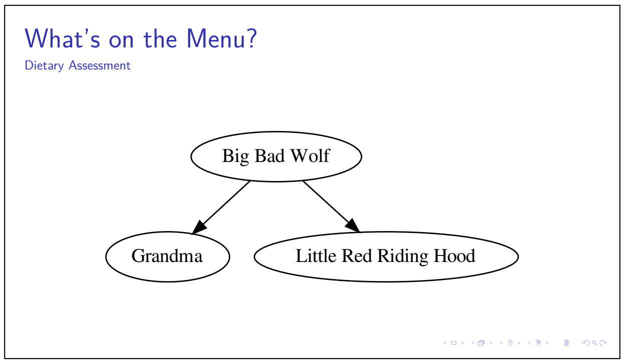


**Figure 3**: Presentation slides produced by Jane Doe using her `jdoe/beamer/headings` LaTeX theme

~~~
# What's on the Menu?
## Dietary Assessment
``` dot
digraph tree {
    Wolf -> Grandma
    Wolf -> Hood
    Wolf [label = "Big Bad Wolf"]
    Hood [label = "Little Red Riding Hood"]
}
\end{markdown}
\end{frame}
\end{document}
~~~

The above code will produce the two presentation slides shown in Figure 3. After adding a couple more features, Jane publishes her theme on CTAN, so that other authors can benefit from it.

### 1.3 Setup snippets

Let us suppose that Jane Doe has decided to create another theme named `jdoe/lists/roman`, which will make her ordered lists use Roman numerals:

```
\ProvidesPackage{markdownthemejdoe_lists_%
                 roman}[2021/06/04]
\markdownSetup{
    rendererPrototypes = {
        olItemWithNumber = {%
            \item[\romannumeral#1\relax.]%
        }
    }
}
```

Jane attempts to apply her theme in a local scope, displaying one of her ordered lists in Arabic numerals and another ordered list in Roman numerals:

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
\begin{markdown}
1. wahid
2. aithnayn
\end{markdown}    % This won't work!
\begin{markdown*}{theme=jdoe/lists/roman}
3. tres
4. quattuor
\end{markdown*}
\end{document}
```

However, the above code will fail and produce the following LaTeX error: "Can be used only in preamble". LaTeX themes are full-featured LaTeX packages, which make permanent changes to a document, can only be loaded in the preamble of a document, and thus can't be applied in a local scope.

LaTeX setup snippets make it possible to *separate the cause from the effect*: We will load a LaTeX theme once in the preamble, the theme will define setup snippets, and we can apply the snippets in local scopes. Jane will first shorten her theme to `jdoe/lists`, making the `roman` segment into a snippet:

```
\ProvidesPackage{markdownthemejdoe_lists}%
                [2021/06/04]
\markdownSetupSnippet{roman}{
    rendererPrototypes = {
        olItemWithNumber = {%
            \item[\romannumeral#1\relax.]%
        }
    }
}
```

Next, Jane will separate the loading of her `jdoe/lists` theme from using her `roman` setup snippet:

```
\documentclass{article}
\usepackage[theme=jdoe/lists]{markdown}
\begin{document}
\begin{markdown}
1. wahid
2. aithnayn
\end{markdown}
\begin{markdown*}{snippet=jdoe/lists/roman}
3. tres
4. quattuor
```

```
\end{markdown*}
\end{document}
```

The above code will produce the following list:

1. wahid
2. aithnayn
iii. tres
iv. quattuor

Notice how the setup snippet `roman` has been automatically *namespaced* to `jdoe/lists/roman`. This makes it less likely that different themes will define setup snippets with the same name. Snippets can also be defined outside of themes, in which case namespacing is not applied and `roman` stays `roman`.

## 2 Two flavors of comments

In TeX, comments fulfill several distinct roles:

1. We can use comments to prevent the processing of some parts of our code without deleting them:

   ```
   %\author{Authors anonymized for review}
   \author{John Doe \and Jane Roe}
   ```

2. We can use comments to write two parallel documents, a technique frequently used to produce documentation in literate programming [4]:

   ```
   % The \cs{foo} command prints ``bar'':
   %  \begin{macrocode}
   \def\foo{bar}
   %    \end{macrocode}
   ```

3. We can use comments to insert little side notes:

   ```
   % Aren't we missing a comma here?
   Let's eat grandma!
   ```

4. We can use comments to prevent TeX's input processor from inserting spaces or starting a new paragraph when word-wrapping newline characters are encountered:

   ```
   My parents have first met in Llanfairp%
   wllgwyngyllgogerychwyrndrobwllllantysi%
                                liogogogoch.
   ```

The language of markdown started out as a preprocessor for the HTML language. As a consequence, markdown does not provide its own syntax for comments and authors are expected to use HTML comments instead:

```
<!-- Aren't we missing a comma here? -->
Let's eat grandma!
```

Unlike TeX's comments, which consume the rest of a line (including the occasional grandma), HTML comments consume just a part of a line, which increases their expressiveness at the expense of verbosity:

```
Let's <!-- eat --> visit grandma!
```

However, the markdown language only allows HTML comments in text, not in the middle of other elements, such as hyperlinks. This makes HTML comments unsuitable for general word-wrapping (point 4):

```
<!-- This won't work! -->
[1]: http://a.very.long.url/that/should<!--
   -->/enjoy%20some%20serious#word-wrapping
```

Although HTML comments can be extracted from an HTML document to create a parallel document for literate programming (point 2), no such option exists in the TeX Markdown package. As a consequence, users of the Markdown package will find HTML comments useful but often lacking compared to TeX comments.

In Section 2.1, we first show how version 2.10.0 of the Markdown package improves the support for HTML comments. In Section 2.2, we introduce a new flavor of markdown comments, which can be combined with HTML comments to cover all use cases of TeX comments and more.

### 2.1 Semantic HTML comments

Since version 2.3.0, the Markdown package has recognized HTML elements, entities, processing instructions, and comments when the `html` option is enabled. HTML entities are resolved, and HTML elements, processing instructions, and comments are omitted from the output:

```
\documentclass{article}
\usepackage[html]{markdown}
\begin{document}
\begin{markdown}
<!-- Aren't we missing *a comma* here? -->
Let's eat <emph>grandma</emph>&excl;
\end{markdown}
\end{document}
```

The above code will produce the text "Let's eat grandma!"

HTML comments are *semantic* in the sense that they are not stripped away by an input processor, but recognized as an element of the markdown language. Since version 2.10.0, the Markdown package includes a renderer that makes the text of the comments actionable:

```
\documentclass{article}
\usepackage{marginnote}
\usepackage[html]{markdown}
\markdownSetup{
  renderers = {
    inlineHtmlComment = {\marginnote{#1}},
  },
}
```

```
\begin{document}
\begin{markdown}
<!-- Aren't we missing *a comma* here? -->
Let's eat grandma!
\end{markdown}
\end{document}
```

The above code will produce the text "Let's eat grandma!" with the comment displayed as a margin note as we see here. This makes HTML comments useful for inserting notes (point 3).

Aren't we missing *a comma* here?

Notice that the inline markdown markup for emphasis is recognized as well. This support for nested formatting makes HTML comments useful for writing parallel documents (point 2).

### 2.2 Lexical TeX comments

The Markdown package has always supported the `hybrid` option, which allows users to use LaTeX commands such as `\label` and `\ref` inside markdown:

```
\documentclass{article}
\usepackage[hybrid]{markdown}
\begin{document}
\begin{markdown}
I conclude in Section~\ref{sec:conclusion}.

Conclusion
==========
\label{sec:conclusion}
In this paper, we have discovered that most
grandmas would rather eat dinner with their
grandchildren than get eaten. Begone, wolf!
\end{markdown}
\end{document}
```

However, since the conversion of markdown to TeX does not preserve newlines, using TeX comments in the `hybrid` mode will lead to unexpected results. For example, typesetting the markdown document from point 4 on page 189 in the `hybrid` mode will produce the text "My parents have first met in Llanfairp".

Since version 2.6.0, the Markdown package has supported the `stripPercentSigns` option, which makes it possible to use TeX comments to produce documentation in literate programming (point 2). [5]

Since version 2.10.0, the Markdown package sets the category code of the percent sign to *other* when typesetting markdown documents, so that TeX comments can't produce malformed documents in the `hybrid` mode. Additionally, a *lexical* input processor that recognizes the regular language of TeX comments (for technical details, see Figure 4) has been added to the Markdown package and can be enabled with the `texComments` option. For example, typesetting the markdown document from point 4 on page 189 with the `texComments` option enabled will produce the expected text "My parents have first met in Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch."

TeX comments are *lexical* in the sense that they are unaware of markdown. Therefore, we can use them for general word-wrapping (point 4):

```
[1]: http://a.very.long.url/that/should/%
     enjoy\%20some\%20serious#word-wrapping
```

In conclusion, both the semantic HTML comments and the lexical TeX comments are well-suited for preventing the processing of some parts of our documents (point 1) and for writing parallel documents (point 2). Whereas HTML comments are better-suited for writing and optionally typesetting little side notes (point 3), TeX comments can be used for word-wrapping anywhere in the text (point 4).

## 3 LuaMetaTeX

The LuaMetaTeX engine is a minimalist development version of LuaTeX, which removes many features of LuaTeX and alters some interfaces. LuaMetaTeX is used in the ConTeXt LMTX format [1], which may soon make its way into TeX Live and other TeX distributions. Since the Markdown package supports ConTeXt, the time is ripe to make Markdown play nice with LuaMetaTeX as well.

LuaTeX contains the Selene Unicode library, which the Markdown package uses to transform Unicode text. LuaMetaTeX removes Selene Unicode, but uses Lua 5.4, which contains a built-in `utf8` library. The `utf8` library has a similar interface and functionality to Selene Unicode. Since version 2.10.0, the Markdown package will use either Selene Unicode or `utf8`, whichever is available in Lua.

LuaTeX contains the `kpathsea` library, which is responsible for finding files in the TeX directory structure. The Markdown package uses `kpathsea` to find JSON files that map filename extensions to names of programming languages for the `contentBlocks` syntax extension. LuaMetaTeX removes the `kpathsea` library, but provides an optional library interface, which allows the use of an external `kpathsea` library when it is available. Since version 2.10.0, the Markdown package will search for files only in the current working directory when `kpathsea` is unavailable. This should only affect ConTeXt Standalone: in full TeX distributions, where an external `kpathsea` library is available, there should be no change of behavior.

## 4 What's next and how do I contribute?

There are many intriguing ideas for the future of Markdown. Some of these ideas are already under

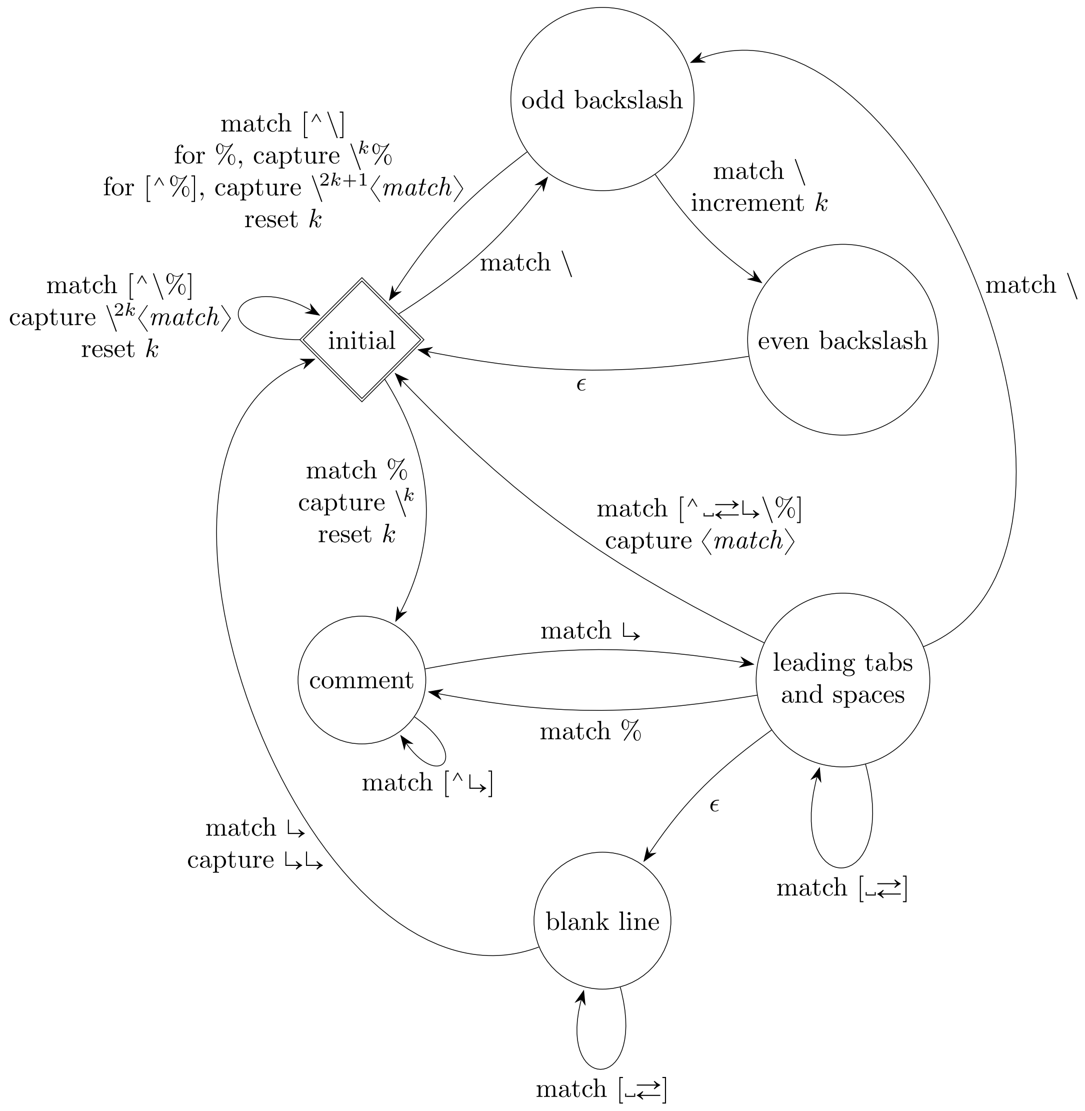


**Figure 4**: An automaton that strips TEX comments from markdown input with the `texComments` option.

The automaton contains a counter $k$, which is initially zero. The automaton reads and *matches* characters from the input to transition between *states*. During a state transition, the automaton may *capture* character strings by writing them to the output, *increment* counter $k$ by one, or *reset* counter $k$ back to zero.

Until the automaton encounters a backslash (\) or a percent sign (%), it stays in the **initial** state, which is similar to the state $M$ of TEX's input processor [2, Chapter 8], capturing every character that it matches.

When the automaton encounters a sequence of backslashes, it counts the pairs of backslashes in counter $k$. If a percent sign follows an **odd backslash**, the percent sign has been *escaped* and the automaton captures $k$ backslashes and the percent sign: This makes capture a surjective function, allowing us to write `\%` as `\\\%`. If a character other than a percent sign or a backslash follows an **odd** or **even backslash**, the automaton captures $2k + 1$ or $2k$ backslashes, respectively, and the matched character: This makes capture an identity function for inputs that do not contain a percent sign.

When the automaton encounters a percent sign that has not been escaped, the automaton captures $k$ backslashes and reads the rest of the line as a **comment** without capturing any characters. After reading the newline character (↳), the automaton enters a state similar to the state $N$ of TEX's input processor, reading any **leading tabs and spaces** (⇄ and ␣) without capturing any characters. If the automaton encounters additional newline characters, there has been a **blank line** in the input and the automaton captures two newline characters (↳↳) before it transitions back to the **initial** state.

development by contributors and soon to become the present reality, whereas some other ideas are only now beginning to be discussed,[1] and others yet are waiting to be discovered by you.

For your inspiration, I list some existing ideas for improving Markdown by increasing complexity and suggest how you can contribute:

### 4.1 Actionable HTML attributes

In my previous article [5, Section 2.4], I introduced HTML attributes as a way of typesetting only small parts of markdown documents. However, the HTML attributes are currently not *actionable*, which means that users can't react to them from TEX.

If the HTML element identifiers were actionable, we could rewrite the code from Section 2.2 without the `hybrid` mode and all its security problems. Additionally, if HTML class names were actionable, we could apply setup snippets without switching between markdown and LATEX:

```
I conclude in Section <#sec:conclusion>.

Conclusion {#sec:conclusion .some-snippet}
==========
In this paper, we have discovered that most
grandmas would rather eat dinner with their
grandchildren than get eaten. Begone, wolf!
```

Future development should add syntax extensions such as Pandoc's `fenced_divs`, `bracketed_spans`, and `inline_code_attributes` for specifying HTML attributes on elements other than headings.

If you would like to contribute, you should have a look at issue 91[2] and the Contributing section of the `README.md` document.[3] The introductory article by Henri Menke [3] about writing parsing expression grammars (PEG) in the Lua LPeg library is recommended reading.

### 4.2 Jekyll front matter

Jekyll is a static site generator that takes Markdown documents and converts them to a website. In Jekyll, each Markdown document can start with *front matter*: a block in your amazing markup language (YAML) that can specify various metadata:

```
---
title: Of *Wolves* and _Grandmas_
author:
- name: Little Red Riding Hood
- name: Big Bad Wolf
---
```

If Jekyll's front matter were supported in Markdown, we could set up all metadata of a document from Markdown without ever switching to TEX.

If you would like to contribute, you should have a look at issue 22[4] and the implementation drafted by Marei Peischl in pull request 77.[5] The article by Henri Menke [3] about writing PEG in the Lua LPeg library is again recommended reading.

### 4.3 The `witiko/graphicx/http` theme in Lua

The `witiko/graphicx/http` LATEX theme from Section 1.1.3 requires either GNU Wget or cURL to download online images. We could remove both prerequisites by using the `socket.http` Lua library.

In issue 82,[6] I drafted an implementation and listed several issues that prevent its use:

1. The `http.request` method mishandles redirects.
2. LuaMetaTEX lacks the `socket.http` library.
3. The `\directlua` command needs to be replaced with a shell escape for non-Lua TEX engines.

Lua programmers familiar with the Luasocket library are encouraged to help tackle points 1 and 2.

### 4.4 Integration with Pandoc

Pandoc is a Haskell library for converting between dozens of document formats. Since it would be difficult to write conversion functions for every pair of formats, Pandoc uses an intermediate abstract syntax tree (AST), so that every document format only needs a conversion function from the document format to the AST and back. If the Markdown package understood the AST, we could typeset any of the document formats of Pandoc while maintaining full control over the formatting:

```
\documentclass{article}
\usepackage[theme=jdoe/lists]{markdown}
\begin{document}
\pandocInput[snippet=jdoe/lists/roman]
            {of-wolves-and-grandmas.docx}
\end{document}
```

If you would like to contribute, you should have a look at the corresponding issues[7] and the GitHub repository of Dominik Rehák,[8] who is extending the Lunamark Lua parser with an AST reader. Ideas on how to best integrate the AST reader into the interface of Markdown will be appreciated.

[1] github.com/witiko/markdown/issues & /discussions
[2] github.com/witiko/markdown/issues/91
[3] github.com/witiko/markdown#contributing
[4] github.com/witiko/markdown/issues/22
[5] github.com/witiko/markdown/pull/77
[6] github.com/witiko/markdown/issues/82
[7] github.com/witiko/markdown/issues/25 & /62
[8] github.com/drehak/lunamark

### 4.5 Direct mapping of elements

In Section 1.2, Jane Doe has created a `jdoe/beamer/headings` LaTeX theme for producing presentation slides. However, we still needed to use the LaTeX `frame` environment for each presentation slide. Could we produce presentation slides without switching between markdown and LaTeX?

The Markdown package relies on TeX's *expansion processor*: For example, the Lua parser converts the markdown text "`# What's on the Menu?`" into the TeX code

```
\markdownRendererHeadingOne{What's on the Menu?}
```

which TeX then *expands* to

```
\frametitle{What's on the Menu?}
```

and typesets. However, we can't always rely on TeX's expansion processor. For example, the Beamer command `\begin{frame}` will read input until it has found a matching `\end{frame}` command. If `\end{frame}` is hidden behind expansion, it will never be found.

One solution would be to make the conversion of "`# What's on the Menu?`" configurable, so that instead of producing `\markdownRendererHeadingOne`, it can *map directly* to "`\begin{frame}{What's on the Menu?}`". If you would like to contribute, you should have a look at issue 92[9] and the Contributing section of the `README.md` document.[10]

On a more decentralized level: play with the Markdown package, cherish it, use it in your writing, and find ways to abuse its syntax in unexpected and unsettling ways. LaTeX themes make it easier than ever to share your discoveries and compose them into a beautiful cacophony of mayhem.

[9] `github.com/witiko/markdown/issues/92`
[10] `github.com/witiko/markdown#contributing`

**Figure 5**: Grandma Jane and the Big Bad Wolf celebrate the fifth birthday of the Markdown package. Illustration by `fiverr.com/quickcartoon`.

### Book announcement

Lloyd R. Prentice and I are writing a book: *Publish Beautiful Books with Markdown: Fast Track to LaTeX* (`publishbeautifulbooks.com`) is about book design, typography, and technology that allows you to go from well-chosen words to a beautiful book with just a few keystrokes.

Lloyd is a novelist and indie book publisher. His novels include *Freein' Pancho* and *The Gospel of Ashes*. He wrote and produced the three-year running web manga *Aya Takeo* with illustrator Sonia Leong; print version published in three volumes by UK independent publisher and comic collaborative Sweatdrop Studios. Lloyd also co-wrote and published the technical programming book *Build It with Nitrogen: The Fast-Off-the-Block Erlang Web Framework*. In an earlier life, Lloyd designed and developed nearly 100 consumer and educational software products for major US publishers. Web experience includes design and development of a soup-to-nuts application to support marketing and management of world-class technical conferences.

I am a computer scientist, university teacher, and digital typography enthusiast. Before I developed the Markdown package, I had prepared a dozen TeX document templates for universities, scientific journals, and small businesses. I am the technical editor of the Czechoslovak TeX users group ($\mathcal{CS}$TUG) Bulletin, which publishes research works on electronic document preparation and digital typography.

### References

[1] H. Hagen. ConTeXt LMTX. *TUGboat* 40(1):34–37, 2019. `https://tug.org/TUGboat/tb40-1/tb124hagen-lmtx.pdf`

[2] D.E. Knuth. *The TeXbook*. Addison-Wesley, 1986.

[3] H. Menke. Parsing complex data formats in LuaTeX with LPEG. *TUGboat* 40(2):129–135, 2019. `https://tug.org/TUGboat/tb40-2/tb125menke-lpeg.pdf`

[4] F. Mittelbach. Format LaTeX documentation, 2021. `https://ctan.org/pkg/doc`

[5] V. Novotný. Markdown 2.7.0: Towards lightweight markup in TeX. *TUGboat* 40(1):25–27, 2019. `https://tug.org/TUGboat/tb40-1/tb124novotny-markdown.pdf`

◇ Vít Novotný
Studená 453/15
Brno, 638 00
Czech Republic
`witiko (at) mail dot muni dot cz`
`github.com/witiko`

# `bib2gls`: sorting

Nicola L. C. Talbot

**Abstract**

When using `makeindex` and `xindy`, it's advisable to provide a sort value when the actual value contains commands or other awkward content that can confuse sorting. With `bib2gls`, which was written specifically for the glossaries-extra package, the advice is the reverse. In general you shouldn't explicitly supply the sort value but instead make use of `bib2gls`'s system of fallbacks. This provides a more flexible approach that makes it easier to share `bib` files across multiple documents that may require different ordering.

## 1 Sorting with `\makeglossaries`

Symbols can be quite difficult to order. Consider the following document that just uses the base glossaries package [9]:

```
\documentclass{article}
\usepackage[style=treegroup]{glossaries}
\makeglossaries
\loadglsentries{constants}
\begin{document}
\gls{pi}, \gls{e}, \gls{gelfondcons} and
\gls{root2}.
\printglossary[nonumberlist]
\end{document}
```

The file `constants.tex` contains the following:

```
\newglossaryentry{pi}{name={\ensuremath{\pi}},
description={ratio of circumference of a circle
 to its diameter},symbol={3.14159}}

\newglossaryentry{e}{name={\ensuremath{e}},
description={Euler's number},symbol={2.71828}}

\newglossaryentry{root2}{
name={\ensuremath{\surd 2}},symbol={1.41421},
description={Pythagoras' constant}}

\newglossaryentry{gelfondcons}{
name={\ensuremath{e\sp\pi}},symbol={23.1406926},
description={Gelfond's constant}}
```

I've used the `symbol` key to store the approximate value, which means that it can be shown in parentheses in the glossary with the `treegroup` style.

The `sort` key hasn't been explicitly set, so its value is obtained from the `name`; but further, since `makeindex` doesn't recognise (LaTeX)TeX commands, it treats `\ensuremath` as a literal backslash followed by 10 letters. The initial backslash results in the entry being placed in the 'symbols' group. All four sort values start with `\ensuremath{` so the relative ordering of those terms is based on the 13th character

**Glossary**

**Symbols**

$\pi$ (3.14159) ratio of circumference of a circle to its diameter.
$\surd 2$ (1.41421) Pythagoras' constant.
$e^\pi$ (23.1406926) Gelfond's constant.
$e$ (2.71828) Euler's number.

**Figure 1**: Default ordering with `makeindex`

**Glossary**

**Numbers**

$\surd 2$ (1.41421) Pythagoras' constant.

**E**

$e$ (2.71828) Euler's number.

**P**

$\pi$ (3.14159) ratio of circumference of a circle to its diameter.

**Figure 2**: Ordering with `xindy` ($\pi$ has sort=pi and Gelfond's constant uses `\sp`)

onwards (backslash comes before 'e' but after '}'): `\p`, `\s`, `e}` and `e\` (Figure 1).

If I switch to `xindy` (which requires adding the xindy package option) then the document build will fail: `xindy` discards commands and characters such as `{ }` and `$`, which means that the sort value for the `pi` entry ends up as an empty string, which `xindy` doesn't allow. A sort value that's acceptable to `xindy` must be provided. For example:

```
\newglossaryentry{pi}{name={\ensuremath{\pi}},
sort={pi},
description={ratio of circumference of a circle
 to its diameter},symbol={3.14159}}
```

This will place the $\pi$ entry in the 'P' letter group (Figure 2).

The $\surd 2$ entry ends up in the 'numbers' group because once the commands and braces have been stripped only '2' is left. Similarly, 'e' is all that remains for both Euler's number and Gelfond's constant. Since `xindy` merges items with duplicate sort values this means that the locations from Gelfond's constant ends up merged into Euler's number location list. In this simple example, the locations are all page 1 and they've been suppressed with the nonumberlist option so it appears as though Gelfond's constant has been ignored.

If I use `^` instead of `\sp` then the sort value for Gelfond's constant becomes `e^` instead of just `e`, which now means all sort values are unique (Figure 3).

## Glossary

**Numbers**

$\surd 2$ (1.41421) Pythagoras' constant.

**E**

$e$ (2.71828) Euler's number.
$e^{\pi}$ (23.1406926) Gelfond's constant.

**P**

$\pi$ (3.14159) ratio of circumference of a circle to its diameter.

**Figure 3**: Ordering with `xindy` ($\pi$ has sort=pi and Gelfond's constant uses `^`)

Let's suppose now that there's a chance that I might have an editor who insists on using an upright font for constants. Providing some commands will make it easier to switch:

```
\newcommand{\constante}{\mathrm{e}}
\newcommand{\constantpi}{\uppi}
```

(`\uppi` requires upgreek [3].) Now `e` and `\pi` need to be replaced with `\constante` and `\constantpi` in the `name` values. This means that with `xindy` the `e` entry will end up with an empty sort value. This will also happen to Gelfond's constant if `\sp` is used. If `^` is used instead then this will end up as the only character in the sort value.

This means that, particularly with `xindy`, entries that are symbols will typically need to have the `sort` key set as they are likely to degrade into empty strings or non-unique values. In the case of these constants, their approximate numeric values may be a more appropriate sort value. A helper command can make it easier to assign. For example:

```
\newcommand{\newconstant}[5][]{%
 \newglossaryentry{#2}{name={#3},symbol={#4},
description={#5},sort={#4},#1}}
\newconstant{pi}{\ensuremath{\constantpi}}
 {3.14159}{ratio of circumference of a
  circle to its diameter}
\newconstant{e}{\ensuremath{\constante}}
 {2.71828}{Euler's number}
\newconstant{root2}{\ensuremath{\surd 2}}
 {1.41421}{Pythagoras' constant}
\newconstant{gelfondcons}
 {\ensuremath{\constante^\constantpi}}
 {23.1406926}{Gelfond's constant}
```

Although `makeindex` can numerically order integers, it doesn't recognise decimals, so all the entries end up in the 'symbols' group ordered according to a string comparison, where '23' comes between '2.' and '3.' (Figure 4).

## Glossary

**Symbols**

$\surd 2$ (1.41421) Pythagoras' constant.
$\mathrm{e}$ (2.71828) Euler's number.
$\mathrm{e}^{\pi}$ (23.1406926) Gelfond's constant.
$\pi$ (3.14159) ratio of circumference of a circle to its diameter.

**Figure 4**: Ordering with `makeindex` by approximate value

## Glossary

**Numbers**

$\surd 2$ (1.41421) Pythagoras' constant.
$\mathrm{e}^{\pi}$ (23.1406926) Gelfond's constant.
$\mathrm{e}$ (2.71828) Euler's number.
$\pi$ (3.14159) ratio of circumference of a circle to its diameter.

**Figure 5**: Ordering with `xindy` by approximate value

With `xindy`, by default the entries end up in the 'numbers' group (since the sort values all start with a digit) and digits come before punctuation so '23' is placed between '1.' and '2.' (Figure 5). In order to sort numerically with `xindy`, it's necessary to use the `numeric-sort` module. You can either call `xindy` directly with `-M numeric-sort` or add the following to the document preamble:

```
\GlsAddXdyStyle{numeric-sort}
```

This now produces the desired result (Figure 6).

If I change my mind and decide to order by the description, I can simply change the definition of `\newconstant`. Other possibilities are to order by definition or by first use in the document (which require the sort=def or sort=use package options). These options work by assigning a numerical (integer) value to the `sort` key that corresponds to the desired order. The value is zero-padded in the event that `xindy` is called without the `numeric-sort` module.

Suppose I now want to switch to using `bib2gls` [7] with glossaries-extra [8]. Ordering by definition or use can now be indicated with the resource options (not package options) sort=unsrt or sort=use.

## Glossary

**Numbers**

$\surd 2$ (1.41421) Pythagoras' constant.
$\mathrm{e}$ (2.71828) Euler's number.
$\pi$ (3.14159) ratio of circumference of a circle to its diameter.
$\mathrm{e}^{\pi}$ (23.1406926) Gelfond's constant.

**Figure 6**: Ordering with `xindy -M numeric-sort` by approximate value

No comparisons are required in these cases, as it's simply a matter of iterating over the list of entries obtained from parsing the bib file or the list of records obtained from parsing the aux file.

Since all entries now have to be defined in the bib file, it's not possible to define a command like \newconstant, but bib2gls provides a flexible way of determining what the sort value should be.

## 2 bib2gls fallbacks

bib2gls has a set of fallbacks that are used if it needs to access a field which hasn't been set. The different entry types have different fallbacks. For example, when sorting entries the default behaviour is to obtain the sort value from the sort field. If this field *has not been set* then the value is obtained from the sort field's fallback. In the case of @entry, the fallback is the value of the name field. In the case of @symbol and @number, the fallback is the entry label (as given in the bib file).

If the fallback is also missing, then the fallback's fallback is used (if one is available) and so on. For example, consider the entry defined as:

```
@index{duck}
```

This only has a label (duck) and no fields. So when bib2gls tries to access the sort field and finds that it hasn't been set, it then tries the fallback for the sort field, which is the value of the name field for this entry type. The name field also hasn't been set, so the fallback for that field is required, which is the entry label. Therefore the sort value ends up as 'duck'.

Now consider

```
@indexplural{duck}
```

Again the fallback for the missing sort field is the value of the name field, which is also missing, but now the fallback for the name field is the value of the plural field, which is also missing. The fallback for plural is the value of the text field with the letter 's' appended. The fallback value for the text field is the entry label. Therefore the sort value ends up as 'ducks'.

Now consider

```
@index{glossary,plural={glossaries}}
@entry{gloscol,
  parent={glossary},
  description={collection of glosses}
}
@entry{gloslist,
  parent={glossary},
  description={list of technical words}
}
```

The sort value for the gloscol entry is obtained as follows:

1. Look up the value of the sort field. This isn't set, so use the fallback value, which is the value of the name field.
2. The name field isn't set, so use the fallback value for that, which is the parent entry's name.
3. The parent field provides the parent's *label* (glossary), so look up the value of the name field for the parent entry.
4. The name field isn't set for the glossary entry, so use the fallback value for that, which is the entry's label.

Therefore the sort value ends up as 'glossary'. The same process for gloslist leads to the same sort value.

It's possible to change the default fallbacks, but some fields, such as description, don't have a fallback, so that will terminate a fallback trail.

If the sort field is explicitly set, then *the fallback is not required* so in that situation changing the system of fallbacks has no effect. The recommendation is that you don't explicitly set the sort field but instead use the fallback system to choose the most appropriate field according to the entry type.

If you select a different field for the sort value, then that field's fallback (if provided) will be used instead; e.g., with the option sort-field=description then any entries which don't have the description field set will have an empty sort value (since there's no fallback for this field).

## 3 Examples

The examples below all use the same set of bib files but use different settings to adjust the order. For brevity, all entries are selected with no locations. The condensed style is designed to show the ordering in as compact a form as possible, for illustrative purposes only. The document fonts are set with:

```
\usepackage[light,condensed,math]{iwona}
\usepackage[T1]{fontenc}
```

The name is formatted in a bold font, but this will not be visible for mathematical content or pictographs. If the symbol field is set, it's shown in parentheses before the description. (Bold parenthetical content is part of the entry's name.) The upgreek and marvosym [2] packages are required for some of the symbols.

The entry definition set up in the preamble for each example is:

```
\setabbreviationstyle{long-short-sm-desc}
\setabbreviationstyle[acronym]{nolong-short-em}

\GlsXtrLoadResources[
 selection=all,save-locations=false,
 ⟨options⟩]
```

This uses the 'sm' abbreviation style for entries defined using `@abbreviation`, which requires the relsize package [1]. The entries defined using `@acronym` will use the 'em' abbreviation style which formats the short form using `\emph`.

Additional `\GlsXtrLoadResources` commands may be present for some examples. The main body of the document just contains `\printunsrtglossary`. Sample entries from each `bib` file are shown below. The complete `bib` files can be downloaded [4].

`abbreviations.bib` contains entries such as:

```
@abbreviation{xml,
  short={XML},
  long={extensible markup language},
  description={a markup language that defines
        a set of rules for encoding documents}
}
@acronym{nasa,
  short={NASA},
  long = {National Aeronautics and Space
          Administration}
}
```

`constants.bib` contains entries such as:

```
@number{pi,
  description={pi},
  name={\ensuremath{\constantpi}},
  symbol={3.14159}
}
@number{root2,
  description={Pythagoras' constant},
  name={\ensuremath{\surd2}},
  symbol={1.41421}
}
@number{zero,
  description={zero},
  name={\ensuremath{0}}
}
```

As with the earlier `makeindex` and `xindy` examples, the `symbol` field has been used to store the approximate values so that they can easily been seen in the glossary.

The custom commands such as `\constantpi` are also provided:

```
@preamble{"
\providecommand{\constanti}{\mathrm{i}}
\providecommand{\constante}{\mathrm{e}}
\providecommand{\constantpi}{\uppi}
\providecommand{\constantgamma}{\upgamma}
\providecommand{\constantphi}{\upphi}
\providecommand{\constantlambda}{\uplambda}"}
```

These definitions can be detected by `bib2gls` and will be used if they are encountered within any sort values.

`entries.bib` contains entries such as:

```
@entry{mineral,
  name = {mineral},
  description = {solid, inorganic,
                naturally-occurring substance}
}
@entry{quartz,
  parent = {mineral},
  name = {quartz},
  description = {hard mineral consisting
                of silica}
}
```

`pictographs.bib` contains entries such as:

```
@symbol{heartsuit,
 name={\ensuremath{\heartsuit}},
 description={heart}
}
@symbol{phone,
  name={\Mobilefone},
  description={mobile phone}
}
```

`terms.bib` contains a mixture of `@index`, `@indexplural` and `@entry`, such as:

```
@index{sample}
@indexplural{homograph}
@entry{diamondjubilee,
  name={diamond jubilee},
  description={sixtieth anniversary}
}
```

It also uses an unknown entry type, for example:

```
@homograph{mineral.diamond,
  name={diamond},
  description={metastable allotrope of carbon}
}
@homograph{shape.diamond,
  name={diamond},
  description={four-sided shape with
              equal sides}
}
```

These entries will be ignored unless they are aliased to an entry type that `bib2gls` recognises.

## 3.1 Default sorting

The first example document uses the default sort settings, but it needs to alias the custom `@homograph` entries to make them behave as though they'd been defined with `@entry` instead:

```
src={terms,pictographs,abbreviations,constants},
entry-type-aliases={homograph=entry}
```

Since no sort method has been specified and there's no document language, the sort method will be alphabetical according to my locale (en-GB). The `--group` switch is used when invoking `bib2gls`. The result is shown in Figure 7.

Note that the entries defined with `@symbol` and `@number` have been ordered according to their label. (For example, `root2` and `phone`.) The homographs (such as 'diamond') trigger a warning from `bib2gls`:

```
Identical sort values for 'shape.diamond' and
'mineral.diamond'
Falling back on ID
```

This means that the *relative* ordering of the homographs is based on their labels (using a simple character code comparison) so the mineral diamond is placed before the shape diamond. Both will still have 'diamond' as the sort value when compared with other entries.

The action to perform in the event of duplicate sort values can be changed by setting a value for identical-sort-action. For example, you can order them according to first use in the document (which doesn't make sense for this example) or according to which entry was defined first in the `bib` file. You can also choose another field to determine the final relative ordering, but only a simple character code comparison is used (not a locale-sensitive alphabetical comparison).

If you have a set of homographs and you want to use a field containing natural language to determine their relative order then you may prefer to use the sort-suffix field instead, which can append the contents of another field to the sort value. This suffix will apply to all sort values, not just the homographs.

The abbreviations have been ordered according to the short form. This means that both XHTML and XML are placed in the 'X' letter group, even though the abbreviation style chosen in the document shows the long form first. This ordering is, however, appropriate for the acronyms such as 'Ofcom' and 'Ofsted'.

## 3.2 Sort suffix and fallbacks

The next example makes some adjustments to the resource options:

```
src={terms,pictographs,abbreviations,constants},
entry-type-aliases={homograph=entry}
sort-suffix=description,
symbol-sort-fallback=name,
abbreviation-sort-fallback=long
```

This will result in an error from inputenc arising from the upright Greek letter `\uppi`. The sort fallback for the symbol entries has been switched to the `name` field. Although the letter group *label* is numeric, `bib2gls` attempts to assign an appropriate title, which it obtains from the sort value of the first entry to be assigned to that group (in this case $\pi$).

Since the sort method is using my en-GB locale, the upright Greek letters will all be placed in their own group at the end because there's no rule for them in the English comparator being used. Since this final group is one that you will typically need to change, `bib2gls` provides a command to make it easy to do this:

```
\newcommand{\bibglssetlastgrouptitle}[2]{%
  \glsxtrsetgrouptitle{#1#2}{Greek}}
```

It's necessary to define this command *before* calling `\GlsXtrLoadResources`, or it will have no effect.

The result is shown in Figure 8. This has still produced some oddities.

As already mentioned, the Greek letters are all at the end of the glossary. The en-GB comparator recognises them as letters (rather than punctuation or other symbols) but they don't form part of the en-GB alphabet so they are all lumped together in a single group.

Some of the symbols have been placed in the 'symbols' group (0, 1, $\sqrt{2}$). This is because `bib2gls` recognised the commands in the `name` field (which is now being used as the sort fallback for symbols) for those entries and was able to convert them into Unicode. The comparator being used (en-GB) recognised those Unicode characters as symbols.

A search of `bib2gls`'s transcript file shows that `bib2gls` was also able to interpret the card suit commands but not the marvosym commands. For example, consider the `phone` entry:

1. The `sort` field hasn't been set, so use the fallback for `@symbol` (which is now the `name` field): `\Mobilefone`. Since `bib2gls` doesn't recognise this command the sort value is empty.
2. The sort-suffix=description setting then appends the contents of the `description` field, so the sort value is now `mobile phone`.

This means that the `phone` entry ends up in the 'M' letter group. Now consider the `heartsuit` entry:

1. The `sort` field hasn't been set so use the fallback for `@symbol` (which is now the `name` field): `\ensuremath{\heartsuit}`. These commands are recognised by `bib2gls` and are converted into the Unicode character U+2661. So the sort value consists of the single character ♡.
2. The sort-suffix=description setting then appends the contents of the `description` field, so the sort value is now '♡heart' but the en-GB comparator considers the heart character as ignorable punctuation and so the sort value becomes 'heart'.

This means that the `heart` entry ends up in the 'H' letter group.

The sort fallback value for abbreviations is given by the setting of abbreviation-sort-fallback. However, this is used by both `@abbreviation` and `@acronym`. In this case, it's necessary to differentiate between `@abbreviation` (which needs to be sorted according to the long form) and `@acronym` (which needs to be sorted according to the short form). This requires using custom-sort-fallbacks instead, which can also be used to differentiate between `@number` and `@symbol`.

The sort-suffix option has also caused 'diamond jubilee' to be placed before the shape 'diamond'. This is because the description is now included in the sort value but by default no separator is inserted before the suffix. So the `shape.diamond` entry starts by fetching the sort fallback value from the `name` field ('diamond') and then appends the description so the sort value becomes 'diamondfour-sided shape with equal sides'. The default break-at=word setting marks the word boundaries so the final sort value is:

```
diamondfour-sided|shape|with|equal|sides|
```

The `diamondjubilee` entry starts by fetching the sort fallback value from the `name` field 'diamond jubilee' and then appends the description so the sort value becomes 'diamond jubileesixtieth anniversary'. Again the default break-at setting marks the word boundaries so the final sort value is:

```
diamond|jubileesixtieth|anniversary
```

The pipe character comes before the letter 'f' so 'diamond jubilee' ends up before 'diamond'.

This can be fixed either by switching to using identical-sort-action or by inserting a marker before the suffix. This marker would need to be a punctuation character or symbol that the comparator recognises as coming before letters. It also needs to come before the word boundary marker and should not be a character that's discarded by the collator.

### 3.3 Sort suffix marker and custom fallbacks

The next example needs to remove the definition of `\bibglssetlastgrouptitle` since the modified settings will now place the Greek characters in the 'symbols' group.

This example still uses sort-suffix to append the description to the sort value. I've chosen digits for the markers to ensure that they're not discarded by the alphabetical collator and word separator used by the sort method.

The resource options are now:

```
src={terms,pictographs,abbreviations,constants},
entry-type-aliases={homograph=entry}
sort-suffix=description,sort-suffix-marker=0,
break-marker=1,symbol-sort-fallback=name,
abbreviation-sort-fallback=long,
custom-sort-fallbacks={acronym=short,
                       number=symbol}
```

The result is shown in Figure 9. The entries defined with `@number` now use the `symbol` field to obtain the missing sort value. Unlike `makeindex` and `xindy`, the alphabetical sort methods don't create a 'numbers' group for items with numeric sort values but instead use the 'symbols' group for any entries that don't belong to a letter group.

The numbers aren't in numeric order (23.140692 is between 2.71828 and 3.14159). In fact, since the sort-suffix option has been set, the sort values for the constants aren't simple numbers but are the number followed by the description. The digit markers (0 and 1) have also caused the pictographs to appear between Euler's constant (0.57721) and Apéry's constant (1.2020569).

There are two entries defined with `@number` that don't have the symbol field set (one and zero). However, the sort-suffix setting appends the description so they end up ordered according to their description. (If you try this example and find them in the 'N' letter group, you need to update your version of `bib2gls`.)

### 3.4 Aliasing and concatenation

The sort-suffix option is turning out to be quite problematic for this example set of entries. The reason for using it was to ensure that the homographs were sorted by name and then description. The option identical-sort-action=description could be used as an alternative, but it won't allow for a locale-sensitive word sort of the description.

Each item in the custom-sort-fallbacks list has the general format:

⟨*original entry type*⟩=⟨*field1*⟩+⟨*field2*⟩...+⟨*fieldN*⟩

where ⟨*original entry type*⟩ is the entry type *as specified in the* `bib` *file*. The homographs were defined in the `bib` file using a custom entry type `@homograph`, which is then aliased to `@entry`. If I use `entry` within custom-sort-fallbacks it will only apply to entries that were defined within the `bib` file with `@entry` (not the entries that were aliased to `@entry`).

If I want to specifically change the sort fallback for entries defined with my custom `@homograph` without altering the fallback for the other entries then I need to use `homograph` for ⟨*original entry type*⟩, and I can use the concatenation operator (+) to create a fallback value that's formed by combining multiple fields. The default separator is a space but may be changed with field-concat-sep.

So this example dispenses with sort-suffix and relies instead on aliasing and the custom sort fallback. The resource options are now:

```
src={terms,pictographs,abbreviations,constants},
entry-type-aliases={homograph=entry},
symbol-sort-fallback=description,
field-concat-sep={.},
custom-sort-fallbacks={abbreviation=long+short,
number=symbol+name,homograph=name+description}
```

The result is shown in Figure 10. Note that I've had to change the default field concatenation separator; otherwise, the default space would result in 'diamond jubilee' (which now doesn't include the description in the sort value) being placed between the two 'diamond' entries (which do have their descriptions in the sort value).

The sort-symbol-fallback setting ensures that the pictographs are sorted according to their descriptions but this setting is overridden by custom-sort-fallbacks for the `@number` entries. Concatenating the `symbol` and `name` fields means that the 'zero' and 'one' entries (which don't have the `symbol` field set) are sorted according to their `name` fields and so are now in the 'symbols' group. However, the alphabetical word ordering means that they're not ordered numerically.

## 3.5 Sub-blocks

As discussed in the previous *TUGboat* article [5], `\printunsrtglossary` simply iterates over all defined entries for the given glossary. When used with `bib2gls` the entry definitions are in the `.glstex` file that's loaded by `\GlsXtrLoadResources`. If there's more than one instance of this command, each `.glstex` file is input sequentially and the entry labels are added to the internal list associated with the corresponding glossary.

As a result, a single `\GlsXtrLoadResources` command doesn't have to correspond to a single glossary. It may be used to process multiple glossaries at the same time (if there's some way of assigning the glossary type) or it may be used to process a sub-block of a single glossary. Each sub-block may be sorted according to a different method. The ordering of the sub-blocks corresponds to the order of `\GlsXtrLoadResources` commands.

When dividing the glossary into sub-blocks, it's possible for letter groups to become fragmented. For example, if my first sub-block contains ant, bee and zebra and the second sub-block contains aardvark, duck and wombat, there will be two 'A' letter groups. This is a contrived example, as it would result in a glossary with the rather odd order: ant, bee, zebra, aardvark, duck, wombat. It's more usual to use different sort methods for each sub-block, which may form different groups. Alternatively, you can override the sort method's group formation and force all entries in a sub-block to belong to a single group.

This next example will have multiple sub-blocks. The first block will be for the mathematical constants, ordered by the numerical value. The approximate value can be obtained from the `symbol` but, as noted above, this doesn't deal with 0 and 1. There are two ways of approaching this:

- the fallback can be made from a combination of the symbol and name (as in the previous example) and then strip any non-numeric content;
- if the `symbol` hasn't been set then copy the name into it.

The first method can be achieved with the resource options:

```
symbol-sort-fallback=symbol+name,
sort-replace={{[^0-9\string\.\string\-]+}{}}
```

The second method can be achieved with the resource options:

```
symbol-sort-fallback=symbol,
replicate-fields={name=symbol}
```

The only problematic entry is $\sqrt{-1}$, which is a complex number and therefore doesn't have a defined order within a set of real numbers. Whilst the TeX parser library used by `bib2gls` recognises `\surd` it doesn't recognise `\sqrt`. Unrecognised commands are ignored, so the sort value ends up as `-1`.

Here, I use the first method to avoid altering the `symbol` field. The resource command is:

```
\GlsXtrLoadResources[
 selection=all,save-locations=false,
 src={constants},sort=double,
 symbol-sort-fallback=symbol+name,
 sort-replace={{[^0-9\string\.\string\-]}{}}
]
```

The sort=double setting uses a double-precision floating point comparator.

The second sub-block contains all of the pictographs. I've now decided to order them according to the character code of the closest matching Unicode symbol. This isn't a problem for the card suits as the TeX parser library recognises the commands, but it doesn't (currently) have support for the marvosym commands. To handle them, I provide suitable definitions within `@preamble`. Using `\providecommand` will ensure these definitions don't override the marvosym definitions within the document. (Alternatively, write-preamble=false can be used to prevent `bib2gls` from writing the contents of `@preamble` to the `.glstex` file.)

For example, I can add the following to the `pictographs.bib` file:

```
@preamble{"
\providecommand{\Email}{\symbol{"1F584}}
\providecommand{\Letter}{\symbol{"1F582}}
\providecommand{\Mobilefone}{\symbol{"1F581}}
\providecommand{\Telefon}{\symbol{"1F57F}}"}
```

Alternatively, this could be added to another file called, say, marvosym.bib which can be loaded at the same time as pictographs.bib:

```
\GlsXtrLoadResources[
 selection=all,save-locations=false,
 src={marvosym,pictographs},
 symbol-sort-fallback=name,sort=letter-case
]
```

The sort=letter-case setting uses a case-sensitive character code comparison. Unlike the locale-sensitive sort methods, there's no attempt to detect word breaks and no characters, such as punctuation, are ignored.

This just leaves the terms and abbreviations:

```
\GlsXtrLoadResources[
 selection=all,save-locations=false,
 src={terms,abbreviations},
 entry-type-aliases={homograph=entry},
 symbol-sort-fallback=description,
 field-concat-sep={.},
 custom-sort-fallbacks={abbreviation=long+short,
  homograph=name+description}
]
```

This is basically the same as the previous example except that the constants.bib and pictographs.bib files are omitted.

The result is shown in Figure 11. If grouping is enabled (that is, if bib2gls is invoked with --group or -g) then all the numerical sort methods (such as sort=double) set the group label to glsnumbers which has the title given by the language-sensitive \glsnumbersgroupname. The character code sort methods (such as sort=letter-case) will assign the group based on the first character of the sort value. For the case-sensitive comparator, this can result in both lower and uppercase letter groups. Any character that isn't a letter (according to the Unicode specifications) is assigned to the group glssymbols, which has the title given by the language-sensitive \glssymbolsgroupname.

Alternatively you can force all entries in a given sub-block into a specific group with the group setting. For example:

```
\glsxtrsetgrouptitle{mcons}
  {Mathematical Constants}
\GlsXtrLoadResources[group={mcons},
 src={constants},⟨other settings⟩
]
\glsxtrsetgrouptitle{icons}{Pictographs}
\GlsXtrLoadResources[group={icons},
 src={pictographs},⟨other settings⟩
]
```

The sub-blocks can be reordered by simply rearranging the \GlsXtrLoadResources commands. (If you find yourself wanting to automatically order by sub-block title then you should actually be using a hierarchical glossary instead [6].)

The ability to provide bib2gls with commands in the @preamble makes it easier to adjust the sort value. For example, by providing the closest matching Unicode value for symbols (as above) or by providing a command that allows you to omit or reorder parts of the sort value. For example:

```
@preamble{"\providecommand{\sortart}[2]{#2}"}
@homograph{bravocry,name={bravo},
  description={\sortart{a}{cry of approval}}
}
```

This command would also need to be defined in the document:

```
\newcommand{\sortart}[2]{#1 #2}
```

So although it's not possible to programmatically define entries (like the earlier \newconstant command) the use of fallbacks, aliases and commands provided in the @preamble allows a flexible approach that can be customised on a per-document basis.

## References

[1] D. Arseneau, M. Swift. The relsize package, 2013. ctan.org/pkg/relsize.

[2] M. Miklavec, T. Henlich, M. Vogel. The marvosym package, 2012. ctan.org/pkg/marvosym.

[3] W. Schmidt. The upgreek package, 2003. ctan.org/pkg/upgreek.

[4] N. Talbot. Sample bib files. dickimaw-books.com/latex/tugboat-bib2gls.

[5] N. Talbot. bib2gls: selection, cross-references and locations. *TUGboat* 41(3), 2020. tug.org/TUGboat/tb41-3/tb129talbot-bib2gls-more.pdf.

[6] N. Talbot. Logical glossary divisions (type vs group vs parent), 2020. dickimaw-books.com/gallery/?label=logicaldivisions.

[7] N. Talbot. bib2gls: Command line application to convert .bib files to glossaries-extra.sty resource files, 2020. ctan.org/pkg/bib2gls.

[8] N. Talbot. The glossaries-extra package, 2020. ctan.org/pkg/glossaries-extra.

[9] N. Talbot. The glossaries package, 2020. ctan.org/pkg/glossaries.

◇ Nicola L. C. Talbot
School of Computing Sciences
University of East Anglia
Norwich Research Park
Norwich NR4 7TJ
United Kingdom
https://www.dickimaw-books.com

*A*

$\zeta(3)$ (1.2020569) Apéry's constant

*B*

**bravo** cry of approval
**bravo** a hired ruffian or killer

*C*

♣ club
$\lambda$ (1.30357) Conway's constant

*D*

**diamond** metastable allotrope of carbon
**diamond** four-sided shape with equal sides
**diamond jubilee** sixtieth anniversary
◇ diamond

*E*

e (2.71828) Euler's number
📧 email
🖂 letter
$\gamma$ (0.57721) Euler's constant

*G*

$e^{\pi}$ (23.140692) Gelfond's constant
$\phi$ (1.61803) golden ratio

*H*

♡ heart
**homographs**
**hypertext markup language (HTML)** the standard markup language for creating web pages

*I*

i ($\sqrt{-1}$) imaginary unit

*L*

☎ telephone

*M*

**mathematical markup language (MathML)** markup language for describing mathematical notation

*N*

***NASA*** National Aeronautics and Space Administration

*O*

***Ofcom*** Office of Communications
***Ofsted*** Office for Standards in Education
1 one

*P*

📱 mobile phone
$\pi$ (3.14159) ratio of circumference of a circle to its diameter

*R*

$\sqrt{2}$ (1.41421) Pythagoras' constant

*S*

**sample**
♠ spade
**scalable vector graphics (SVG)** XML-based vector image format

*X*

**extensible hypertext language (XHTML)** XML version of HTML
**extensible markup language (XML)** a markup language that defines a set of rules for encoding documents

*Z*

0 zero

**Figure 7**: Default sorting (see p. 197)

*Symbols*

0 zero
1 one
$\sqrt{2}$ (1.41421) Pythagoras' constant

*B*

**bravo** a hired ruffian or killer
**bravo** cry of approval

*C*

♣ club

*D*

◇ diamond
**diamond jubilee** sixtieth anniversary
**diamond** four-sided shape with equal sides
**diamond** metastable allotrope of carbon

*E*

e (2.71828) Euler's number
📧 email
**extensible hypertext language (XHTML)** XML version of HTML
**extensible markup language (XML)** a markup language that defines a set of rules for encoding documents
$e^{\pi}$ (23.140692) Gelfond's constant

*H*

♡ heart
**homographs**
**hypertext markup language (HTML)** the standard markup language for creating web pages

*I*

i ($\sqrt{-1}$) imaginary unit

*L*

🖂 letter

*M*

**mathematical markup language (MathML)** markup language for describing mathematical notation
📱 mobile phone

*N*

***NASA*** National Aeronautics and Space Administration

*O*

***Ofsted*** Office for Standards in Education
***Ofcom*** Office of Communications

*S*

**sample**
**scalable vector graphics (SVG)** XML-based vector image format
♠ spade

*T*

☎ telephone

*Greek*

$\gamma$ (0.57721) Euler's constant
$\lambda$ (1.30357) Conway's constant
$\pi$ (3.14159) ratio of circumference of a circle to its diameter
$\phi$ (1.61803) golden ratio
$\zeta(3)$ (1.2020569) Apéry's constant

**Figure 8**: Sort suffix and fallbacks (see p. 198)

*Symbols*

$\gamma$ (0.57721) Euler's constant
♣ club
◇ diamond
📧 email
♡ heart
🖂 letter
📱 mobile phone
♠ spade
☎ telephone
$\zeta(3)$ (1.2020569) Apéry's constant
$\lambda$ (1.30357) Conway's constant
$\sqrt{2}$ (1.41421) Pythagoras' constant
$\phi$ (1.61803) golden ratio
i ($\sqrt{-1}$) imaginary unit
e (2.71828) Euler's number
$e^{\pi}$ (23.140692) Gelfond's constant
$\pi$ (3.14159) ratio of circumference of a circle to its diameter

*B*

**bravo** a hired ruffian or killer
**bravo** cry of approval

*D*

**diamond** four-sided shape with equal sides
**diamond** metastable allotrope of carbon
**diamond jubilee** sixtieth anniversary

*E*

**extensible hypertext language (XHTML)** XML version of HTML
**extensible markup language (XML)** a markup language that defines a set of rules for encoding documents

*H*

**homographs**
**hypertext markup language (HTML)** the standard markup language for creating web pages

*M*

**mathematical markup language (MathML)** markup language for describing mathematical notation

*N*

***NASA*** National Aeronautics and Space Administration

*O*

***Ofcom*** Office of Communications
***Ofsted*** Office for Standards in Education
1 one

*S*

**sample**
**scalable vector graphics (SVG)** XML-based vector image format

*Z*

0 zero

**Figure 9**: Sort suffix marker and custom fallbacks (see p. 199)

*Symbols*

$\gamma$ (0.57721) Euler's constant
0 zero
$\zeta(3)$ (1.2020569) Apéry's constant
$\lambda$ (1.30357) Conway's constant
$\sqrt{2}$ (1.41421) Pythagoras' constant
$\phi$ (1.61803) golden ratio
1 one
$i$ ($\sqrt{-1}$) imaginary unit
$e$ (2.71828) Euler's number
$e^{\pi}$ (23.140692) Gelfond's constant
$\pi$ (3.14159) ratio of circumference of a circle to its diameter

*B*

**bravo** a hired ruffian or killer
**bravo** cry of approval

*C*

♣ club

*D*

**diamond** four-sided shape with equal sides
**diamond** metastable allotrope of carbon
♢ diamond
**diamond jubilee** sixtieth anniversary

*E*

📧 email
**extensible hypertext language (XHTML)** XML version of HTML
**extensible markup language (XML)** a markup language that defines a set of rules for encoding documents

*H*

♡ heart
**homographs**
**hypertext markup language (HTML)** the standard markup language for creating web pages

*L*

🖂 letter

*M*

**mathematical markup language (MathML)** markup language for describing mathematical notation
✆ mobile phone

*N*

***NASA*** National Aeronautics and Space Administration

*O*

***Ofcom*** Office of Communications
***Ofsted*** Office for Standards in Education

*S*

**sample**
**scalable vector graphics (SVG)** XML-based vector image format
♠ spade

*T*

☎ telephone

**Figure 10**: Aliasing and concatenation (see p. 200)

*Numbers*

$i$ ($\sqrt{-1}$) imaginary unit
0 zero
$\gamma$ (0.57721) Euler's constant
1 one
$\zeta(3)$ (1.2020569) Apéry's constant
$\lambda$ (1.30357) Conway's constant
$\sqrt{2}$ (1.41421) Pythagoras' constant
$\phi$ (1.61803) golden ratio
$e$ (2.71828) Euler's number
$\pi$ (3.14159) ratio of circumference of a circle to its diameter
$e^{\pi}$ (23.140692) Gelfond's constant

*Symbols*

♠ spade
♡ heart
♢ diamond
♣ club
☎ telephone
✆ mobile phone
🖂 letter
📧 email

*B*

**bravo** a hired ruffian or killer
**bravo** cry of approval

*D*

**diamond** four-sided shape with equal sides
**diamond** metastable allotrope of carbon
**diamond jubilee** sixtieth anniversary

*E*

**extensible hypertext language (XHTML)** XML version of HTML
**extensible markup language (XML)** a markup language that defines a set of rules for encoding documents

*H*

**homographs**
**hypertext markup language (HTML)** the standard markup language for creating web pages

*M*

**mathematical markup language (MathML)** markup language for describing mathematical notation

*N*

***NASA*** National Aeronautics and Space Administration

*O*

***Ofcom*** Office of Communications
***Ofsted*** Office for Standards in Education

*S*

**sample**
**scalable vector graphics (SVG)** XML-based vector image format

**Figure 11**: Sub-blocks (see p. 201)

# Programming bibliographies

Jean-Michel Hufflen

**Abstract**

We are interested in situations such that using the full expressive power of a programming language is needed when 'References' sections are generated for a source text suitable for LaTeX. The data model used by BibTeX is inadequate from this point of view; the biblatex package is based on a more efficient data model, but workarounds may be needed in some circumstances.

## 1 Introduction

Early computer programs were very different from those running nowadays. Sometimes they were written using programming languages with syntactical features now viewed as strange; they were running on computers whose performance was not comparable with today's. Besides, these early programs, in the 1950s, were only handled by people specialised in computer science: on the one hand, some rigid syntax was required for inputs, on the other hand, outputs were provided using raw forms, and graphical interfaces were nonexistent.

Things have evolved rapidly for years now, and nowadays many people are able to use programs and operating systems even if they do not have any knowledge about programming. A good example is given by interactive word processors (WYSIWYG[1]), such as Microsoft Word. Secretaries are able to use them, as end users simply typing input texts. Any end user of Word is able to customise it, mainly by means of graphical menus. So nowadays a rich collection of programs — including Word — can be used as any object of everyday life, like a washing machine or a dishwasher.

What is the point of non-interactive typesetting systems (WYSIWYM[2]), such as LaTeX, according to this point of view? In fact, if an end user only deals with default constructs or predefined document classes, LaTeX can be viewed as a kind of *black box* simply accepting input source texts and producing output texts. Some customisation and extension can be reached by means of *packages*, introduced by LaTeX 2$\varepsilon$ [10]. Stronger customisation is allowed by means of *commands*, written using TeX's programming language. Theoretically, this language has the same expressive power as a Turing machine's language, so any function can be programmed using it.[3]

In practice, TeX's language — which is based on *macros* — has been mainly designed to handle *text fragments*. As a kind of counter-example, even though we can implement a sort procedure using this language [9], that is a worthwhile exercise for its own sake, rather than an actual advantage for using (LA)TeX. In fact, some 'pre-computations' — before typesetting a fragment — such as capitalising some words, or putting them using lower or uppercase characters, can be easily expressed using TeX's language, but more advanced features related to programming features are difficult to handle.

This point is one of the reasons why LuaTeX has been developed [2], allowing tasks more related to programming to be delegated to a more modern programming language. Since the initial versions of LuaTeX, (LA)TeX is fully able to typeset the result of a computation performed by a program, for example, sorting an array before displaying its successive elements. Typesetting the results built by a spreadsheet program is another example.

In this article, we propose to apply such a view to *bibliography processors*, that is, generating 'References' sections for LaTeX documents. In other words, many end users should be able to use such a program without any knowledge about programming, but some specific applications may need such knowledge — provided that the format used by this processor is open — and it is important for such a bibliography processor to allow the direct programming of some specific functions.

In Section 2, we recall the tasks a bibliography processor should perform and we make precise our terminology. Section 3 briefly reports the main bibliography processors' current state. Then Section 4 goes thoroughly into some specific points. After some discussion, Section 5 concludes about our approach.

## 2 Tasks of a bibliography processor

Let us consider a source text (`.tex` file) including some bibliographical citations by means of LaTeX's `\cite` command, whose argument is a so-called *citation key.*

(i) A bibliography processor can use citation keys to *search* bibliography databases (`.bib` files) for corresponding *entries.*

[1] **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et.

[2] **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **M**ean.

[3] There are many online documents about this subject, more or less easy to read. A didactic written document is [6], showing how to implement a kind of $\lambda$-calculus in TeX.

doi.org/10.47397/tb/42-2/tb131hufflen-bibprog

(ii) These entries should be *sorted*, unless the document's bibliography is *unsorted*, that is, the order of the bibliography's items is the order of first citations throughout the document's body.

(iii) Finally, each bibliographical entry should be arranged into a bibliographical *reference* that future readers can consult, most often at the document's end, in which case they are stored in a `.bbl` file. Citation keys used throughout the document are replaced during this step: a reference may be identified by a number in *plain* bibliography styles, by an alphanumeric label in *alpha* styles. More advanced bibliography styles, such as *author-date* or *short-title*, have been successfully implemented within LaTeX, as reported in [11, Ch. 12].

## 3 Bibliography processors

For a long time, BibTeX [13] was the only bibliography processor usually associated with LaTeX. It uses information stored in auxiliary (`.aux`) files and is able to perform the steps (i), (ii) and (iii). BibTeX is still used, at least by some conference submission tools, although it is an old program, which does not address modern requirements such as multilingual encodings. BibTeX's bibliography styles have been implemented using the `.bst` language [12], sometimes complemented by a LaTeX 2$\varepsilon$ package within some advanced styles such as `natbib` or `jurabib` [11, Ch. 12]. This language for bibliography styles — based on handling a stack — is old-fashioned, more suitable for small changes than programming a new bibliography style. However, this language expresses general *algorithms*. That is, an end user developing a new style using this language has available the possible sequence of operations a computer can perform. Nevertheless, the same observation can be made as for using TeX's language: this `bst` language is difficult to handle practically.

During the last decade, a new *modus operandi* has been put into action by the biblatex package [8]: formatting bibliographical references — that is, most of step (iii) — is wholly controlled by TeX macros and deferred to LaTeX's next pass. Steps (i) and (ii) can be delegated to BibTeX, but biber [7] is preferred, because more features are available. The biber program does not use `.aux` files, but *control files* (`.bcf` files), built when the biblatex package is loaded with the option `backend=biber`. In addition, this biblatex/biber duo has introduced many new fields, many new bibliography types and styles; it seems widespread in the humanities.

The `bib` module of ConTeXt — another format built on top of TeX [1] — works analogously, in the sense that many tasks are deferred to ConTeXt's next pass; modern versions of this module are mostly implemented using the Lua programming language.

As another possible bibliography processor, we have personally put into action the implementation of a successor to BibTeX: MlBibTeX [3], written in the Scheme programming language. We think that a functional programming language is very suitable for this kind of task, because such languages allow the use of *functions as parameters*, e.g. in:

```
(lambda (f2) (f2 1 2))
```

To add (resp. multiply) `1` and `2`, just apply this expression to `+` (resp. `*`). A bibliography style can be compared to this expression, because the *resources* are known — they are the successive bibliographical entries — whereas the way to arrange these data together is to be determined w.r.t. the bibliography style chosen. MlBibTeX has been developed according to such an approach. It has been used rather occasionally, but as far as we know, using it has always proven successful. In particular, it has been used to populate the French official site for open archives from a collection of `.bib` files, as reported in [4]. A new version has been announced in [5]; its new features include a better interface with Scheme definitions. This point seems to us to be important and we are going to explain why.

## 4 Some operations

### 4.1 Sorting

Let us consider again the tasks enumerated in §2. Step (ii) — sorting the extracted bibliographical entries — would be very difficult to program in TeX's macro language, as mentioned in the introduction. Although the biblatex package tends to use LaTeX to perform as many operations as possible, this sort step is still performed by the bibliography processor associated, BibTeX or biber. The former provides a 'basic' lexicographical sort procedure, in practice only suitable for the English language (without manual adjustments). The biblatex package allows some customisation of the sort procedure performed by biber, by means of the `sorting` option and *mnemonics* denoting sort keys — e.g., `nyt` is a sorting scheme for name, year, and title [8, §3.1.2.1].

*Neither* of these two processors can perform a numeric sort:[4] with these two bibliography processors, the year information is correctly processed because years are supposed to use the same number of digits. Let us go thoroughly into chronological

[4] ... although biblatex's documentation is very precise about the *types* used within fields.

order: BibTEX does not sort w.r.t. the month information; biber could do that, since the month information handled by biblatex consists of numbers,[5] but in practice no predefined sorting scheme does so.

The biblatex/biber duo does offer more ways to sort bibliographical entries in comparison with 'old' BibTEX. Moreover, its sorting schemes allow people unfamiliar with computer science to specify precise sort keys, but:

- a *descending* sort can be applied only to years within predefined schemes;[6]
- the number of *person names* — for authors or editors — considered for sorting — given by the `maxnames` constant [8, §3.1.2.1] — is limited;[7] the sort procedure does not deal with *unbounded* number of person names, even if the maximum number considered can be changed by end users;
- some bibliographical fields mainly used as sort keys — e.g., `AUTHOR`, `TITLE`, `YEAR` — may be substituted by sort-suitable fields during this procedure — `SORTNAME`, `SORTTITLE`, `SORTYEAR`: excess markup for corresponding information is avoided, but there is some risk of *information redundancy*;
- some exotic sorts can be handled by redefining the beginning and end of the sort procedure, by means of the bibliographic fields `PRESORT` and `SORTKEY` [8, §3.6]; more difficult cases can be processed by defining new sorting schemes by means of the `\DeclareSortingTemplate` command [8, §4.5.6] — to be put in a source `.tex` file — but this last command defines intermediate sort keys by means of additional fields rather than new sort procedures.

In our personal opinion, the biblatex package's conventions allow people unfamiliar with programming to deal with a rich collection of possible sorts, but if you need to add a new sort procedure, you do not have the full expressive power of a programming language.[8] As an ambitious example, we personally were in charge of the publication list of our laboratory some years ago: we had to sort this list first by research teams, second by *categories*,[9] third by decreasing years, fourth by authors' names (increasing), fifth by decreasing months. That would have been possible with biblatex/biber but quite difficult and/or requiring some information redundancy.

### 4.2 Labelling references

The automatic generation of non-ambiguous bibliographical keys for *references* — the keys that will appear within the resulting document — is now satisfactory. Let us remark that if end users would like to use their own keys, they should use a *short-title* system.[10] If we consider *alpha* styles, we would like to point out that such a style does not belong to the author-date system. BibTEX and biblatex add a letter if several references share the same author and year, that is:

[Rob 1964a] Kenneth Robeson. *The Man of Bronze.* No. 1 in *Doc Savage Series.* Bantam Books, October 1964.

[Rob 1964b] Kenneth Robeson. *The Thousand-Headed Man.* No. 2 in *Doc Savage Series.* Bantam Books, October 1964.

According to another approach, we can label a first or unique reference with a name's abbreviation and a year. If need be, a possible second reference with the same information is given an additional letter, that is:

[Rob 1965] Kenneth Robeson. *The Polar Treasure.* No. 4 in *Doc Savage Series.* Bantam Books, April 1965.

[Rob 1965a] Kenneth Robeson. *Brand of the Werewolf.* No. 5 in *Doc Savage Series.* Bantam Books, April 1965.

The first way is often used in modern documents, but the second can be observed. Our opinion is that this problem can be solved with access to the function generating successive labels within the bibliography processor.

## 5 Conclusion

We think that MlBibTEX's new version will be finished at this year's end. We expect to provide an open system, comparable with the Emacs[11] editor. That is, usable as it is, but also *customisable* and *extensible.* As mentioned in the introduction, LaTEX

[5] Two-digit numbers, of course.

[6] To do this for new sorting schemes, use the `\sort` construct with the option `direction=descending` inside a `\DefineSortingTemplate` command [8, §4.5.6].

[7] That is the same within 'old' BibTEX, because a unique string is built as a sort key for each entry. So the number of co-authors or co-editors used during BibTEX's sort procedure is limited by this string's length.

[8] . . . even if you can program using Perl, the language used for biber's implementation.

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

[10] With 'old' BibTEX, end users can use the predefined `KEY` field with a suitable bibliography style. But it seems that such a *modus operandi* has very rarely been put into action in practice.

[11] **E**diting **MAC**ros.

has these qualities, and the success of LuaTEX shows that the addition of a modern programming language's power has opened new windows.

Even if not all of a bibliography processor's customers are computer scientists, tasks performed by such a tool belong to programming. When the UNIX operating system emerged, the idea was that each person using it would be close to someone who precisely knew how UNIX worked. That is, a kind of *synergy*. We think that a comparable synergy should be reached for an open bibliography processor, which implies that the structures it uses are open. This was already the case for MlBIBTEX's last version. Likewise, most of the new fields and types introduced by biblatex will be recognised and handled by MlBIBTEX's last version, so we can claim that we are not in opposition to biblatex. We aim to bring another view of what a bibliography processor should be.

**Acknowledgements**

It is a great pleasure to thank the first version's proofreaders, Barbara Beeton and Karl Berry.

**References**

[1] CONTEXTGARDEN: *Bibliographies in MkIV*. July 2012. `https://wiki.contextgarden.net/Bibliography_mkiv`.

[2] Hans HAGEN: "LuaTEX: Howling to the Moon". *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, vol. 23, pp. 63–68. April 2006.

[3] Jean-Michel HUFFLEN: "MlBIBTEX's Version 1.3". *TUGboat*, vol. 24, no. 2, pp. 249–262. July 2003. `https://tug.org/TUGboat/tb24-2/tb77hufflen.pdf`

[4] Jean-Michel HUFFLEN: "Using MlBIBTEX to Populate Open Archives". In: Tomasz PRZECHLEWSKI, Karl BERRY, Gaby GIC-GRUSZA, Ewa KOLSAR and Jerzy B. LUDWICHOWSKI, eds., *Typographers and Programmers: Mutual Inspirations. Proc. BachoTEX 2010 Conference*, pp. 45–48. April 2010.

[5] Jean-Michel HUFFLEN: "From MlBIBTEX 1.3 to 1.4". In: Tomasz PRZECHLEWSKI, Karl BERRY, Bogusław JACKOWSKI and Jerzy B. LUDWICHOWSKI, eds., *Various Faces of Typography. Proc. BachoTEX 2015 conference*, pp. 13–17. Bachotek, Poland. April 2015.

[6] Alan JEFFREY: "Lists in TEX's Mouth". *TUGboat*, vol. 11, no. 2, pp. 237–245. June 1990. `https://tug.org/TUGboat/tb11-2/tb28jeffrey.pdf`

[7] Philip KIME and François CHARETTE: *biber. A Backend Bibliography Processor for biblatex. Version biber 2.16 (biblatex 3.16)*. 19 December 2020. `https://ctan.org/pkg/biber`.

[8] Philip KIME, Moritz WEMHEUER and Philipp LEHMAN: *The biblatex Package. Programmable Bibliographies and Citations*. Version 3.16. 31 December 2020. `https://ctan.org/pkg/biblatex`.

[9] Kees VAN DER LAAN: "Sorting within TEX". *TUGboat*, vol. 14, no. 3, pp. 319–328. October 1993. `https:///TUGboat/tb14-3/tb40laan-sort.pdf`

[10] Leslie LAMPORT: *LATEX: A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[11] Frank MITTELBACH and Michel GOOSSENS, with Johannes BRAAMS, David CARLISLE, Chris A. ROWLEY, Christine DETIG and Joachim SCHROD: *The LATEX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[12] Oren PATASHNIK: *Designing BIBTEX Styles*. February 1988. Part of the BIBTEX distribution. `https://ctan.org/pkg/bibtex`

[13] Oren PATASHNIK: *BIBTEXing*. February 1988. Part of the BIBTEX distribution. `https://ctan.org/pkg/bibtex`

◇ 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`
`members.femto-st.fr/Hufflen-Jean-Michel/en`

## TUG 2021 abstracts

TUG 2021 abstracts

Editor's note: Links to videos and other information posted at: `tug.org/tug2021`.

— — ∗ — —

### XLingPaper's use of TEX technologies

*Andy Black, Hugh Paterson*

XLingPaper is a plugin to XMLMind, an XML editor designed for publishers. We describe XLingPaper's development history and its dependencies on TEX packages for PDF creation.

XLingPaper does three things. 1) It controls the user interface of a powerful tool, only allowing valid document sections to be inserted into a document, thus reducing user friction in the document production process. 2) It provides a constrained number of document sections which are relevant in the production of linguistically-oriented publications, e.g., grammars, dissertations, theses, journal articles, edited volumes, etc. 3) It exports documents to a variety of formats, e.g., PDFs, ePUB, OpenOffice Writer, HTML.

### Introduction to LATEX — in Spanish

*Alexánder Borbón Alpízar*

This presentation, in Spanish, provided a basic introduction to LATEX, mentioning its principal distributions and the most-commonly used editors. In addition, the basic structure of a simple document was shown, along with how to write mathematical equations, tables and figures.

To review these topics it is suggested to check the author's book *Edición de textos científicos con LATEX. Composición, gráficos, diseño editorial y presentaciones beamer*, available at:
`tecdigital.tec.ac.cr/revistamatematica/Libros/LaTeX/MoraW_BorbonA_LibroLaTeX.pdf`

### 2020: A year in review, living on an island

*Paulo Cereda*

In this talk, Paulo recalls 2020 on the Island of TEX: an eventful year with a new backend for the online TEX and LATEX documentation lookup system, the release of a tool for finding fonts that contain a given Unicode glyph, a major update for `arara` and other actions and initiatives as a means to enrich the TEX ecosystem. Yet, a new adventure is about to unfold, for the Island has bold and exciting plans for the future.

### Variable fonts

*David Crossland*

Making the web more beautiful, fast, and open through great typography.

### R and LATEX: Typesetting graphs in a reproducible way

*Vic van Dijk*

Knitr ties LATEX and R together in a very powerful combination. Ti*k*Z typesets visually appealing graphs from R code. Data is processed upon typesetting a report. All calculations can be made available to the reader as R code. This simplifies reproducible research. R offers a whole ecosystem of statistic procedures, graph packages, and even connections to other systems such as Python and MATLAB.

In this talk I will show the applications of R and LATEX that I came across. My aim is to typeset beautiful graphs in a widely accessible manner. One example is available at:
`setyourtext.com/en/blog/graphing-nitrogen-dioxide-air-concentration-data`

### Towards 21st century digital typography

*Jonathan Fine*

This abstract is a short essay giving the framework for my talk. I take a long view. In my talk I'll provide some details and examples. My talk is about digital typography in 2050 and 2070, and the conditions for its emergence that are already present.

A few billion years ago life in the oceans began oxygenating the atmosphere. By 350 million years ago life on land was creating what we now call fossil fuel (coal, crude oil and natural gas). A few million years ago the genus homo (man) emerged.

Birds have song and dance. The tool-making Neanderthals (250,000 to 40,000 years ago) probably had language. Human art and music arose at least 40,000 years ago. Around 14,000 years ago agriculture and settlement started to replace nomadic hunt and gather. Writing (on tablets) followed about 5,000 years ago.

Ancient history (3000BC to AD500) includes about 80 civilizations worldwide with written records. This is a very rich period which still influences contemporary thought in art, religion, society, culture and politics.

Along with the rise of the European Renaissance in the 1400s, printing with moveable type emerged, to replace hand copying of books. This is typography, born out of calligraphy (writing with pen or brush).

By the 20th century there were massive printing presses, producing a million copies or more of each issue of a newspaper, which were then distributed on a national basis. (In 1950 the *News of the World* sold over 8 million copies each week.)

Also in the 20th century there was electrification, wireless stations and receivers, and studios. This distributed spoken voice news, and music, to millions.

Cinema and then television provided moving images to accompany the sound.

By 2020 vast torrents of information were being created and transmitted using computers and networks (mobile phones, wi-fi and 4G). Computers are everywhere, even in electric light bulbs. The present context is very different from the 1970s, when Don Knuth started his foundational work on digital typography, and the creation of TEX and Metafont.

Gutenberg and others replaced hand copying of books by the printing press. Knuth and others replaced mechanical typography by computer (or digital) typography. Both produce only static visual images.

If humanity avoids destroying its culture and civilization, then the digital typography of 2050 will be different again. It is already emerging. One major component is the (world wide) web and its servers and browsers. This was pioneered by Tim Berners-Lee. Another is the smart mobile phone (now dominated by Apple and Android). A third is the large high-resolution flat screen television. A fourth is the ubiquity of computers.

I am now in my late 60s. I hope to be alive to see the digital typography of 2050, and if so I expect some surprises. Maxwell's unification of electricity and magnetism (1865) lives on as the theoretical basis for electrification, wireless and much more. I hope the work of Knuth and others in digital typography can similarly be transmitted as useful living tools and skills to those who follow us.

I do not expect to be alive in 2070, yet alone the 100th birthday of TEX (2078 to 2082). I hope my contribution adds to the cause for celebration.

## Code and math in the dark

*Jonathan Fine*

It's said: Easy reading is hard writing. Certainly both reader and writer need to make extra effort, when the reader is visually impaired, and the material is technical. This talk is about improving the accessibility of TEX and its outputs. This is not a typesetting talk. It is a user experience and social interaction talk.

For sighted readers the printed page assists short-term memory, as does typography. They reduce the cognitive load. The eye can pick up subtle hints. Clarity of organisation and writing will reduce the cognitive load for both visually impaired and sighted readers, provided they have sufficient verbal skills.

This year I've had regular online discussions about accessibility with blind and visually impaired persons, and listened in on their forum conversations with each other. I've learnt a lot from this.

The introduction of computers and networks has been, with some exceptions, an enabling technology for the visually impaired. A screen reader allows the user to hear what is written, without needing a sighted assistant. And video calls by mobile phone means that the sighted assistant need not be physically present.

Louis Braille, who became blind as a young child, developed the tactile code for reading and writing that we now know simply as Braille. Screen readers allow the visually impaired to write computer software. The major screen readers are JAWS, Orca and NVDA. It should be no surprise that their leading developers, Glen Gordon, Mark Mulcahy, Michael Curran and James Teh are all blind.

To summarize, my talk will share what I've learned from my interactions with blind and visually impaired users, and how it relates to the accessibility of TEX and its outputs.

## LATEX to HTML conversion with TEX4ht

*Michal Hoftich*

TEX4ht is a converter from LATEX to HTML and several other output formats. Recent work focuses on keeping current with package updates, and supporting new packages. In this talk, I will discuss its current status and recent development. I will show how to change the look of the generated document, how to select the right way to produce math (including MathJax and MathML), and how to fix some common issues caused by clashes with unsupported packages or commands.

## Cary Graphic Arts Collection Pressroom Tour

*Amelia Hugill-Fontanel*

A visit and work discussion with a letterpress printer. For much more, see `cary.rit.edu`.

## Graphics with PGFPlots — in Portuguese

*Emílio Kavamura*

The workshop, in Portuguese, intends to briefly present the features of PGFPlots for LATEX users. The topics covered start from the environment description, present the types of graphics and their components, and possible customizations. The presentation ends with the use of the `animate` package to provide animations from a set of inline text graphics. The code for the examples presented is available for you to try and to evaluate the capabilities of the graphic environment in LATEX.

## Bidirectional typesetting in TeX: Past, present, and future

*Vafa Khalighi*

This talk is based primarily on my last 15 years of TeX development in the area of bidirectional typesetting. I will look at the current state of bidirectional typesetting in TeX, discuss the issues I have faced, the current challenges, and what needs to be done.

I will also discuss how the `bidi` package is used for typesetting bidirectional documents and show a few sample documents (books, theses, and other types of documents) produced by the `bidi` package. Some capabilities of the `bidi` package will be demonstrated live.

## Persian typesetting in TeX: Past, present, and future

*Vafa Khalighi*

This talk is based primarily on my last 15 years of TeX development in the area of Persian typesetting. I will look at the current state of Persian typesetting in TeX, discuss the issues I have faced, the current challenges, and what needs to be done.

I will also discuss how the `xepersian` package is used for typesetting mainly Persian documents and show a few sample documents (books, theses, and other types of documents) produced by the `xepersian` package. Some capabilities of the `xepersian` package will be demonstrated live.

## Tactile mathematics: Enabling sighted and blind people to share mathematical experience

*Alexei Kolesnikov, Al Maneki, Michael Cantino, Rob Beezer, Volker Sorge*

High quality automated transcription of mathematical texts, including graphics, into tactile form is an open problem. In this talk, we describe the reasons for producing tactile forms of mathematical texts. We will describe common challenges involved in transcription, and progress made to date. We make the case that semantically rich source files are needed to produce adequate tactile and audio-tactile forms of scientific materials.

## Reviving Type 3 fonts for modern LuaLaTeX documents

*Marcel Krüger*

For a long time, Type 3 fonts in LaTeX-generated PDF files were known for (undesirable) bitmap fonts, but that's only a small aspect of what this font format can do. With OpenType color fonts, the idea behind Type 3 fonts has seen a revival, and LuaTeX recently added support for adding such fonts for non-bitmap use cases too.

In this talk I want to look at how this format can be used to create smaller and simpler PDF files involving color fonts and user-generated glyphs and consider advantages and disadvantages in contrast to traditional alternatives like virtual fonts or macro-based solutions.

## Taming the beast — Advances in paragraph tagging with pdfTeX and XeTeX

*Frank Mittelbach*

In this talk I demonstrate and describe our solution for automatically tagging paragraphs when using engines such as pdfTeX or XeTeX. The situation with LuaTeX is different, and simpler, and therefore not the subject of this talk. I briefly touch on the problems one encounters and explain the approaches we used to overcome them. This will be done with a number of demonstrations intermixed with theoretical explanations.

This work is part of our multi-year journey to gradually modernize LaTeX so that it can automatically produce high-quality tagged and "accessible" PDF without the need to post-process the result of the LaTeX run.

## Accessible research reports: Case study, including acronyms and glossaries

*Ross Moore, Tom Price*

US government agencies have a need for properly accessible PDFs. The practice of 'remediation' (adjusting and augmenting the PDF after the typesetting phase) is both expensive and produces generally poor results.

In this talk we show how a much better product can be created directly using LaTeX, adapted for constructing documents that fully conform to PDF/UA-1 and PDF/A-3a. LaTeX sources are handled at three levels: (i) initial data capture by research scientists, (ii) heavy editorial work to enable accessibility aspects, (iii) production-level processing to produce feature-rich tagging and full accessibility. The two speakers will discuss different aspects of these three levels, according to their own involvement in this generalised workflow.

Of particular interest is the use of acronyms and glossaries to enrich the PDF with features that associate technical terms and abbreviations with a fully expanded description of the meanings of those terms, accessible both visually and to assistive technology for non-visual readers.

Here are some links to websites referred to early in the talk.

- NOAA Statement on Accessibility: `noaa.gov/accessibility`

- PDF 508 Accessibility Checklist: `ssa.gov/accessibility/checklists/pdf/pdfchecklist.html`
- Ugly Truth: PDF Accessibility Services Don't Meet 508 Document Compliance Requirements: `krisrivenburgh.medium.com/pdf-accessibility-services-dont-meet-508-document-compliance-requirements-e659dccbdb3b`
- Next-generation PDF: `ngpdf.com`, preferred for validation (using veraPDF) of Tagged PDF documents, and conversion into HTML.

Here's a link to the lecture slides, and to where the main example PDF may appear, once published by NOAA, and thereby released into the public domain.

- Acronyms for NOAA Center Reference Documents (CRDs): `tug.org/tug2021/assets/pdf/Thomas-E-Price-Jr-slides.pdf`
- Examples of Tagged PDF documents built using LaTeX: `web.science.mq.edu.au/~ross/TaggedPDF/`

### Publishing for all: Using LaTeX to help improve the accessibility of an open-access journal

*Michael Nolan, Todd Pagano, Suhas Chikkanaravangala Vijayakumar, Rahul Jaiswal*

A screen reader is a vital tool that helps individuals who are blind or low-vision read digital text. Unfortunately, not all file formats receive the same level of support from screen readers. For example, while PDF files have accessibility features that can be used, they are often not the preferred file format for screen reader users. Between line breaks, multiple columns, symbols, and images, screen readers often struggle with academic journal articles in certain file formats.

We will discuss the collaboration of the Open at RIT project with an open access journal and their combined goal of improving accessibility and readership for all. We will explore the difficulties that journals face on their journey towards accessibility, why this journey is worth making, and show how using LaTeX to publish both to our traditional PDF format as well as a more accessible HTML format allowed us to make a big leap towards becoming a more accessible journal.

### Data-driven documents using Jupyter Notebooks and Overleaf

*Simon Porter*

We will show how Digital Science combines Jupyter Notebooks and Overleaf projects for automated creation of professional-looking documents, and team collaboration. An example: `gigantum.com/sjcporter/gigaleaf-example`

### How a LaTeX-based company lived through a modern day pandemic

*Aravind Rajendran, Rishi T, Apu V, Rahul Krishnan S*

Now we all know what a pandemic is and how dreadful it is for all mankind. Most generations who are alive today would have not known one, never seen one, never felt one. Surviving these trying times has made us all adapt to change and we are no stranger. As a typesetter for the leading scientific, technical and medical publishers around the world, we have helped typeset thousands of pages of research articles on COVID-19. Somewhere through our business of 'typesetting' — directly or indirectly we feel we have helped. Our main objective during this challenging period, was to keep the business rolling safely, making sure our employees and customers were not let down. Through this journey we have helped our staff work safely ensuring they had no job loss. And for the scientific community we have worked tirelessly, helping publishers continue publishing research articles quickly without losing a single committed due date. This is how we assured business continuity and certainty for our employees, customers and business.

Looking back, with gratitude we can now confidently shout out, yes, we have achieved what we had set out for — making LaTeX work for a business, ensuring no job losses, standardising our workflows, making data-driven analytical decisions within LaTeX workflows and to sum it all, keep delivering aesthetically pleasing documents to our customers and delighting them always. It was indeed not a cake walk, times were superbly challenging and we have persevered. Finding the Goldilocks between aesthetics and efficiency has always been a challenge for large production houses. But in this pandemic time we have achieved just that, the right balance, "Our Goldilocks" for LaTeX-based typesetting. This is our journey, this is our story; the story still continues...

### How to make a logo/symbol for a font

*Matheus Rocha*

Learn how to create a new symbol and make an OpenType font for your logo to be used in TeX and elsewhere.

### Producing a book for Amazon KDP

*Paulo Ney de Souza*

Details of producing a book in LaTeX for Amazon Kindle Direct Publishing.

**Any colo(u)r you like**
*Joseph Wright*

TeX itself has no built-in support for colour, which is therefore handled by specials or engine-specific extensions. For LaTeX $2_\varepsilon$, the different interfaces are abstracted out by the color package. However, there is a lot that the color package does not do; for example, handling colour model interconversion, mixing colours or device-specific colour spaces. Packages such as `xcolor` and `colorspace` fill that gap, whilst the `luacolor` package addresses a separate issue: avoiding the need to use whatsits for colour at all.

As part of wider efforts to enhance the LaTeX kernel via `expl3` additions, recent work on the `l3color` package has brought many of these concepts into a single set of interfaces. That means not only copying existing ideas but also ensuring maximal functionality. In my talk, I will explore the work on `l3color`, highlighting where it can go beyond the predecessor packages in ease of use and functionality.

◇ TUG 2021 abstracts
`https://tug.org/tug2021`

## *MAPS* 51 (2021)

*MAPS* is the publication of NTG, the Dutch language TeX user group (`https://www.ntg.nl`).

MAPS REDACTIE, Redactioneel [Editorial]; p. 1

HANS HAGEN, Waarom is een NTG-lidmaatschap belangrijk? [Why is an NTG membership important?]; p. 2

Discusses the importance of NTG membership.

HANS HAGEN, Proud or ashamed; pp. 3–6

A critique of an official Dutch COVID-related document produced with TeX.

HANS HAGEN, Lost in fonts; pp. 7–8

This article explains how to deal with old TTF or OTF fonts that do not have any OpenType features in the new releases of ConTeXt that use LuaTeX or LuaMetaTeX with native font loading.

HANS HAGEN, UTF-8 in MetaPost; pp. 9–12

A new small extension to the MetaPost library used in LuaMetaTeX, enabling UTF-8 input natively.

HANS HAGEN, Extensions related to programming macros; pp. 13–24

This paper presents a number of fundamental extensions in LuaMetaTeX to make programming macros in it more comfortable. [Another version was published in *TUGboat* 42:1.]

J.A.M. VERMASEREN, Playing with Axodraw; pp. 25–39

This paper shows some of the features of Axodraw. It emphasizes applications, not only in the field of physics, but also in completely unrelated fields such as mathematical tiling constructions, fashion patterns and the design of sudokus.

TOMÁŠ SZANISZLO, Two Questions and Answers Sessions by Donald Knuth at FI MU; pp. 40–64

In October 2019, the Faculty of Informatics, Masaryk University, hosted Donald Knuth as a guest who led two question and answer sessions at this occasion, dedicated to the themes of Computer Science and art. Besides some background on these lectures, you can also find their transcripts in this article. [A shorter version was published in *TUGboat* 41:2.]

HANS VAN DER MEER, Translations from a vocabulary; pp. 65–68

Formerly part of the module `hvdm-xml` but now split off into an independent module with its own description. Used for making other modules language sensitive. The module is especially tailored for XML use.

HANS VAN DER MEER, Macros and Lua snippets; pp. 69–76

Described is a module containing a number of helper macros, many of them programmed in Lua.

JERZY LUDWICHOWSKI, GUST e-foundry font projects, closing report 2019–2020; pp. 77–78

[Published in *TUGboat* 42:1.]

PIETER VAN OOSTRUM, Fancyhdr Ontwikkeling [Fancyhdr developments]; pp. 79–96

This article provides an overview of the development of the LaTeX package `fancyhdr`, and the tools used in it. It also presents an overview of the testing method.

SIEP KROONENBERG, Ontwikkelingen in TeX Live [Developments in TeX Live]; pp. 97–98

This article highlights changes in TeX Live in recent years.

[Received from Wybo Dekker.]

## *Die TEXnische Komödie* 2–3/2021

*Die TEXnische Komödie* is the journal of DANTE e.V., the German-language TEX user group (dante.de). Non-technical items are omitted.

### *Die TEXnische Komödie* 2/2021

VOLKER RW SCHAA, Protokoll der 62. Mitgliederversammlung von DANTE e.V. am 11. März 2021 in Magdeburg (remote) [Protocol of the 62. General Meeting of DANTE e.V. on March 11 2021 in Magdeburg (remote)]; pp. 6–14

Minutes of the General Meeting.

DORIS BEHRENDT, Bericht der Schatzmeisterin für das Jahr 2020 [Report of the Treasurer for 2020]; pp. 14–27

Report of the Treasurer for 2020.

JERZY LUDWICHOWSKI, Finaler Report 2019–2020 für die GUST e-foundry Font Projekte [GUST e-foundry font projects, final report 2019–2020]; pp. 28–31

[Published in *TUGboat* 42:1.]

STEPHAN LUKASCZYK, Tagungsbericht Frühjahrstagung 2021 [Report of the General Meeting Spring 2021]; pp. 31–38

This report summarizes the course of the spring meeting 2021 of DANTE e.V. Due to the COVID-19 pandemic the meeting was completely virtual, not at the Otto-von-Guericke University in Magdeburg.

THOMAS HILARIUS MEYER, MARTIN SIEVERS, Einladung zur Herbsttagung 2021 und 63. Mitgliederversammlung von DANTE e.V. [Invitation to the Autumn Conference 2021 and 63. General Meeting in Saarbrücken]; pp. 38–39

The Autumn Conference 2021 and 63. General Meeting will take place in Saarbrücken, if it is not held virtually.

UWE ZIEGENHAGEN, Von Trello nach LATEX [From Trello to LATEX]; pp. 41–47

Trello is an internet service based on the Kanban task management method used by millions of users worldwide. In this post I would like to show you how to use the Python API to create individual Read out Trello boards and export them to LATEX.

HARALD LICHTENSTEIN, Schleifen in Ti*k*Z [Looping in Ti*k*Z]; pp. 48–52

You can save a lot of typing effort when drawing with Ti*k*Z by using loops. Loop counters can be integers as well as letters. This short article shows how it works.

UWE ZIEGENHAGEN, Ergänzung zum `pgfornament` Artikel aus *Die TEXnische Komödie* 1/2021 [Remarks on the `pgfornament` article from *Die TEXnische Komödie* 1/2021]; pp. 53–55

Based on my article on `pgfornament` in *Die TEXnische Komödie* 1/2021 Heiko Oberdiek approached me with helpful comments on the TEX registers that I used in the article. Since his hints are certainly interesting for other TEX enthusiasts as well, I would like to share them here and thank Heiko for the feedback.

### Die TEXnische Komödie 3/2021

ADELHEID BONNETSMÜLLER, Having Fun with LATEX: Jetzt werden Boxen bunt [Having fun with LATEX: Colored boxes with `bclogo`]; pp. 11–18

Description of the `bclogo` package by Patrick Fradin and Maxime Chupin, part of Adelheid's "Having fun With LATEX" series.

RAINER-MARIA FRITSCH, Mit arara externe LATEX-Dokumente erzeugen und einfügen [Creating and inserting external LATEX documents with `arara`]; pp. 19–29

With `arara`, the production of LATEX documents can be automated in a plethora of ways. In this article we show how a main document and its standalone child documents can be created using `arara` commands in the preamble of the main document.

FRANK MITTELBACH AND CHRIS ROWLEY, LATEX Tagged PDF – Ein Entwurf für ein großes Projekt [LATEX Tagged PDF — A proposal for a big project]; pp. 30–44

In this article we outline the proposal for a project to improve LATEX's handling of Tagged PDF, a necessity for standards like PDF/UA.

LATEX PROJECT TEAM, LATEX News, Issue 32, Oktober 2021 [LATEX News, Issue 32, October 2021]; pp. 44–58

An overview of the changes in the LATEX release 2021-10-01.

JÜRGEN FENN, Neue Pakete auf CTAN [New packages on CTAN]; pp. 59–62

An overview of new packages on CTAN.

[Received from Uwe Ziegenhagen.]

# The Treasure Chest

These are the new packages posted to CTAN (`ctan.org`) from April–September 2021. 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*.

We hope this column helps people access the vast amount of material available through CTAN and the distributions. See also `ctan.org/topic`. Comments are welcome, as always.

◇ Karl Berry
`https://tug.org/TUGboat/Chest`

## biblio

`apalike-ejor` in `biblio/bibtex/contrib`
BibTeX style for *European Journal of Operational Research*.

`newcastle-bst` in `biblio/bibtex/contrib`
BibTeX support for Harvard at Newcastle style.

## fonts

`aboensis` in `fonts`
Late medieval OpenType cursive font.

`musixtex-fonts` in `fonts`
Fonts used by MusixTeX.

## graphics

`byo-twemojis` in `graphics/pgf/contrib`
Build your own Twemojis by stacking Ti*k*Z paths.

`cartonaugh` in `graphics/pgf/contrib`
Draw Karnaugh maps with up to six variables.

`coffeestains` in `graphics/pgf/contrib`
Save time by printing coffee stains directly on the page instead of adding them manually.

`easing` in `graphics/pgf/contrib`
A collection of easing functions for PGF.

`minim-hatching` in `graphics`
Creating tiling patterns with `minim-mp`.

`nndraw` in `graphics/pgf/contrib`
Draw neural networks.

`pfdicons` in `graphics/pgf/contrib`
Draw process flow diagrams in chemical engineering.

`strands` in `graphics/pgf/contrib`
Draw objects from strands, such as permutations.

`tikz-bricks` in `graphics/pgf/contrib`
Drawing customizable bricks with Ti*k*Z.

`tikz-swigs` in `graphics/pgf/contrib`
Split elliptical nodes horizontally or vertically.

`toneval` in `graphics/pgf/contrib`
Visualize tone value patterns.

`worldflags` in `graphics/pgf/contrib`
The national flags of the world, in Ti*k*Z.

`xistercian` in `graphics/pgf/contrib`
Cistercian numerals in LaTeX, using Ti*k*Z.

## info

`basiclatex-ru` in `info/russian`
An introduction to LaTeX, in Russian.

`tlmgr-intro-zh-cn` in `info`
Chinese translation of the `tlmgr-basics` document.

`visualfaq-fr` in `info`
FAQ LaTeX visuelle francophone.

## language

`bangla` in `language/bengali`
Comprehensive LaTeX package for Bangla.

## macros/generic

`texdimens` in `macros/generic`
Conversion of TeX dimensions to decimals.

`wichura-table` in `macros/generic`
TaBlE macros for (La)TeX, now under the LPPL.

## macros/latex/contrib

`acrotex-js` in `macros/latex/contrib`
Common JavaScript files used by `acrotex` et al.

`bjfuthesis` in `macros/latex/contrib`
Thesis class for Beijing Forestry University.

`bilingualpages` in `macros/latex/contrib`
Typeset two columns in parallel.

`bmstu-iu8` in `macros/latex/contrib`
Class for IU8 reports at Bauman Moscow State Technical University.

`codehigh` in `macros/latex/contrib`
Highlight code using `l3regex` or LPEG.

`crumbs` in `macros/latex/contrib`
Add navigation path to the page header.

`docassembly` in `macros/latex/contrib`
JavaScript for Acrobat Pro with any workflow.

`etl` in `macros/latex/contrib`
Expandable token list operations.

`ffcode` in `macros/latex/contrib`
Fixed-font code blocks using `minted` and `tcolorbox`.

`gamebooklib` in `macros/latex/contrib`
Set numbered entries in shuffled order.

`geradwp` in `macros/latex/contrib`
Document class for the *Cahiers du GERAD* series.

`graphicscache` in `macros/latex/contrib`
Cache `\includegraphics` calls.

`href-ul` in `macros/latex/contrib`
Underline hyperlinks as on the web.

`huawei` in `macros/latex/contrib`
Template for Huawei company documents.

`hvlogos` in `macros/latex/contrib`
TeX-related names as logos, extending `hologo`.

`graphics/pgf/contrib/toneval`

`iexec` in `macros/latex/contrib`
Input the output from shell commands.

`ifallfalse` in `macros/latex/contrib`
Compare a string against a set of other strings.

`kdpcover` in `macros/latex/contrib`
Covers for books published with Kindle Direct.

`keyparse` in `macros/latex/contrib`
Define and evaluate key-based replacement rules.

`lambdax` in `macros/latex/contrib`
Use lambda expressions within LaTeX.

`macrolist` in `macros/latex/contrib`
List operations for LaTeX 2$\varepsilon$.

`mecaso` in `macros/latex/contrib`
Formulas frequently used in rigid body mechanics.

`mlc` in `macros/latex/contrib`
Add `\makelabels` feature to KOMA-Script.

`nchairx` in `macros/latex/contrib`
Mathematical physics macros from chair X of Würzberg University.

`pgfmath-xfp` in `macros/latex/contrib`
Define `pgfmath` functions using `xfp`.

`palette` in `macros/latex/contrib`
Swappable palettes for colors and symbols.

`projlib` in `macros/latex/contrib`
Tools to simplify writing LaTeX documents.

`scrambledenvs` in `macros/latex/contrib`
Scrambled environments, e.g., randomized hints.

`smart-eqn` in `macros/latex/contrib`
Automatic math symbol styling.

`spbmark` in `macros/latex/contrib`
Customize superscript and subscript positioning, both text and math.

`styledcmd` in `macros/latex/contrib`
Handling multiple versions of user-defined macros.

`tiscreen` in `macros/latex/contrib`
Mimic screen of older Texas Instruments calculators.

`tabularray` in `macros/latex/contrib`
Typeset tabulars and arrays with LaTeX3.

`texsurgery` in `macros/latex/contrib`
LaTeX companion to Python project `texsurgery`.

`to-be-determined` in `macros/latex/contrib`
Highlight text passages that need further work.

`uni-titlepage` in `macros/latex/contrib`
Universal titlepages, configurable and predefined.

`zref-check` in `macros/latex/contrib`
Flexible cross-references with contextual checks based on `zref`.

## macros/latex/contrib/beamer-contrib/themes

`thubeamer` in `m/l/c/b-c/themes`
`beamer` theme for Tsinghua University.

`beamertheme-simpledarkblue` in `m/l/c/b-c/themes`
Template for a simple presentation.

## macros/latex/contrib/biblatex-contrib

`biblatex-cv` in `m/l/c/biblatex-contrib`
Create a CV from BibTeX files.

`biblatex-lncs` in `m/l/c/biblatex-contrib`
BibLaTeX style for the *Springer Lecture Notes in Computer Science* series.

`biblatex-spbasic` in `m/l/c/biblatex-contrib`
BibLaTeX style for Springer's journals.

## macros/luatex/generic

`minim` in `macros/luatex/generic`
Modern plain format for the LuaTeX engine.

`minim-math` in `macros/luatex/generic`
OpenType math integration into plain LuaTeX.

`minim-mp` in `macros/luatex/generic`
`mplib` (MetaPost) integration into plain LuaTeX.

`minim-pdf` in `macros/luatex/generic`
Low-level PDF/A integration into plain LuaTeX.

`minim-xmp` in `macros/luatex/generic`
Embed XMP metadata in PDF with plain LuaTeX.

## macros/luatex/latex

`pyluatex` in `macros/luatex/latex`
Execute Python code on the fly in LuaLaTeX.

## macros/luatex/optex

`pdfextra` in `macros/luatex/optex`
Extra PDF features for OpTeX.

## macros/unicodetex/latex

`fontsetup-nonfree` in `macros/unicodetex/latex`
`fontsetup` support for nonfree fonts; available from `https://contrib.texlive.info`.

`inputnormalization` in `macros/unicodetex/latex`
Wrapper for XeTeX/LuaTeX input normalization

`nwafuthesis` in `macros/unicodetex/latex`
Northwest A&F University (China) theses.

`tipauni` in `macros/unicodetex/latex`
Unicode characters from TIPA commands.

## macros/xetex/latex

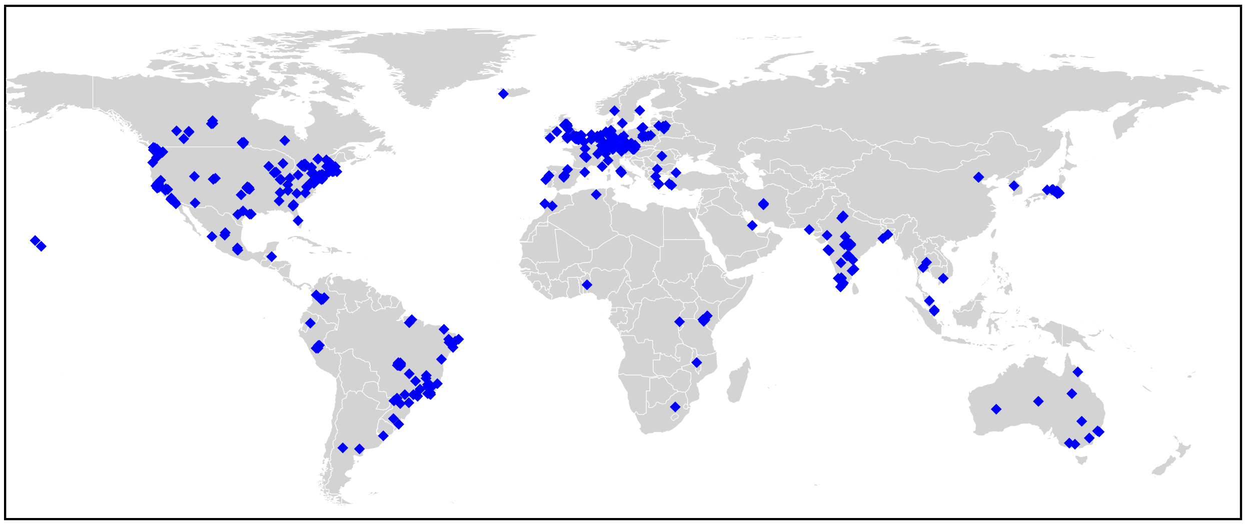
`arabic-book` in `macros/xetex/latex`
An Arabic book class for XeLaTeX.

`book-of-common-prayer` in `macros/xetex/latex`
Typeset liturgical documents in the style of the 1979 *Book of Common Prayer*.

## support

`optexcount` in `support`
Python script to count words in OpTeX documents.

Map of approximate locations of TEX Users Group 2021attendees; created by Boris Veytsman.



## TUG 2021 conference sponsors

TEX USERS GROUP

dante e.V.

THE UNIVERSITY of ADELAIDE

Google

Hello from TEXnology!

***Macro writing, Design, Data Visualization, E-Publishing, Innovations, and more***

We've been writing macro files and teaching LATEX for more than 30 years.

**Now offering LATEX classes via** zoom
See www.texnology.com/train.htm for class dates, description of course and sample of hyperlinked course notes.

See you on-line!

– Amy Hendrickson
amyh@texnology.com

And, come visit our site for many and diverse examples of our LATEX projects, with perhaps some ideas for projects you'd like to develop:

WWW.TEXNOLOGY.COM

*Science is what we understand well enough to explain to a computer. Art is everything else we do.*

*— Donald E. Knuth*

stmDOCS

*the confluence of art and science of text processing in the cloud!*

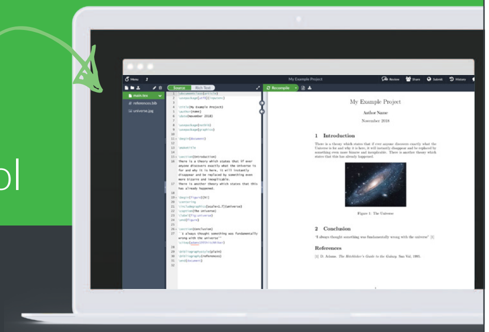
- empowering authors to self-publish
- assisted authoring
- TeXFolio — the complete journal production in the cloud
- NEPTUNE — proofing framework for TeX authors

STM DOCUMENT ENGINEERING PVT LTD

Trivandrum • India 695571 • `www.stmdocs.in` • `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**

# TeX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at `tug.org/consultants.html`. If you'd like to be listed, please see there.

**Dangerous Curve**
Email: `typesetting (at) 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 of us co-authored, designed, and illustrated a TeX book (*TeX for the Impatient*).

We can: convert your documents to LaTeX from just about anything, type your documents from 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 TeX-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. A member of TUG, we also have supported the GNU Project for decades (and even have worked 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`
Web: `http://www.texnology.com`
Full time LaTeX consultant for more than 30 years — Our macro packages are used by thousands of authors. See our site for many samples: `texnology.com`.

Macro packages for books, journals, slides, posters, e-publishing and more.

Design as well as LaTeX implementation for e-publishing or print books and journals, or specialized projects.

Data visualization, database publishing.

LaTeX training via Zoom: Many years experience in on-site training, now offering scheduled Zoom classes! See `www.texnology.com/train.htm`.

I'll be glad to hear from you!

**Dominici, Massimiliano**
Email: `info (at) typotexnica.it`
Web: `http://www.typotexnica.it`
Our skills: layout of books, journals, articles; creation of LaTeX classes and packages; graphic design; conversion between different formats of documents.

We offer our services (related to publishing in Mathematics, Physics and Humanities) for documents in Italian, English, or French. Let us know the work plan and details; we will find a customized solution. Please check our website and/or send us email for further details.

**Latchman, David**
2005 Eye St. Suite #6
Bakersfield, CA 93301
+1 518-951-8786
Email: `david.latchman (at) texnical-designs.com`
Web: `http://www.texnical-designs.com`
LaTeX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized LaTeX packages and classes to meet your needs. Contact us to discuss your project or visit the website for further details.

**Monsurate, Rajiv**
Web: `https://www.rajivmonsurate.com`
`https://latexwithstyle.com`
I offer: design of books and journals for print and online layouts with LaTeX and CSS; production of books and journals for any layout with publish-ready PDF, HTML and XML from LaTeX (bypassing any publishers' processes); custom development of LaTeX packages with

documentation; copyediting and proofreading for English; training in LaTeX for authors, publishers and typesetters.

I have over two decades of experience in academic publishing, helping authors, publishers and typesetters use LaTeX. I've built typesetting and conversion systems with LaTeX and provided TeX support for a major publisher.

**Sofka, Michael**
8 Providence St.
Albany, NY 12203
+1 518-331-3457
Email: `michael.sofka (at) gmail.com`

Personalized, professional TeX and LaTeX consulting and programming services.

I offer 30 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in TeX and LaTeX: Automated document conversion; Programming in Perl, Python, C, R and other languages; Writing and customizing macro packages in TeX or LaTeX, `knitr`.

If you have a specialized TeX or LaTeX need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

**Veytsman, Boris**
132 Warbler Ln.
Brisbane, CA 94005
+1 703-915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`

TeX and LaTeX consulting, training, typesetting and seminars. Integration with databases, automated document preparation, custom LaTeX packages, conversions (Word, OpenOffice etc.) and much more.

I have about two decades of experience in TeX and three decades of experience in teaching & training. I have authored more than forty packages on CTAN as well as Perl packages on CPAN and R packages on CRAN, published papers in TeX-related journals, and conducted several workshops on TeX and related subjects. Among my customers have been Google, US Treasury, FAO UN, Israel Journal of Mathematics, Annals of Mathematics, Res Philosophica, Philosophers' Imprint, No Starch Press, US Army Corps of Engineers, ACM, and many others.

We recently expanded our staff and operations to provide copy-editing, cleaning and troubleshooting of TeX manuscripts as well as typesetting of books, papers & journals, including multilingual copy with non-Latin scripts, and more.

**Warde, Jake**
90 Resaca Ave.
Box 452
Forest Knolls, CA 94933
+1 650-468-1393
Email: `jwarde (at) wardepub.com`
Web: `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 TeX from editors at Addison-Wesley who were using it to publish beautifully set math and computer science books.

Long story short, I started using LaTeX 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 TeX I can help. And if I cannot help I'll let you know right away.

## TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:
`tug.org/instmem.html`
Thanks to all for their support!

American Mathematical Society, *Providence, Rhode Island*

Association for Computing Machinery, *New York, New York*

Aware Software, *Newark, Delaware*

Center for Computing Sciences, *Bowie, Maryland*

CSTUG, *Praha, Czech Republic*

Duke University Press, *Durham, North Carolina*

Hindawi Foundation, *London, UK*

Institute for Defense Analyses, Center for Communications Research, *Princeton, New Jersey*

L3Harris, *Melbourne, Florida*

Maluhy & Co., *São Paulo, Brazil*

Marquette University, *Milwaukee, Wisconsin*

Masaryk University, Faculty of Informatics, *Brno, Czech Republic*

Nagwa Limited, *Windsor, UK*

Overleaf, *London, UK*

StackExchange, *New York City, New York*

Stockholm University, Department of Mathematics, *Stockholm, Sweden*

TeXFolio, *Trivandrum, India*

Université Laval, *Ste-Foy, Québec, Canada*

University of Ontario, Institute of Technology, *Oshawa, Ontario, Canada*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

VTeX UAB, *Vilnius, Lithuania*

## Calendar

**2021**

| | |
|---|---|
| Sep 18 | DANTE 2021 Herbsttagung (online), Saarbrücken, Germany. `www.dante.de/veranstaltungen/herbst2021` |
| Sep 20 – 25 | 15th International ConTeXt Meeting, “Expanding orbits”, Bassenge, Belgium. `meeting.contextgarden.net/2021` |
| Oct 1 – 3 | Oak Knoll Fest XXI (online), “Women in the Book Arts”, New Castle, Delaware. `www.oakknoll.com/fest` |
| Oct | Association Typographique Internationale (ATypI) annual conference (online), Paris, France. `www.atypi.org` |
| Oct 21 – 24 | TypeCon 2021, “Together”, A Virtual Conference. `typecon.com` |
| Nov 13 | TeXConf 2021 (Japan; online) `texconf2021.tumblr.com` |

**2022**

| | |
|---|---|
| Apr 10 – 13 | CODEX VIII, “Words on the Edge”, Richmond, California. `www.codexfoundation.org` |
| Jun 8 – 10 | Grapholinguistics in the 21st century — From graphemes to knowledge, Paliseau, France. `grafematik2022.sciencesconf.org` |
| Summer | Digital Humanities 2022, Alliance of Digital Humanities Organizations, Tokyo, Japan. `adho.org/conference` |
| Jul 11 – 15 | SHARP 2022, “Power of the written word”. Society for the History of Authorship, Reading & Publishing. University of Amsterdam, The Netherlands `www.sharpweb.org/main/conferences` |
| Aug 7 – 11 | SIGGRAPH 2022, Vancouver, Canada. `s2022.siggraph.org` |

**Owing to the COVID-19 pandemic, schedules may change. Check the websites for details.**

*Status as of 15 September 2021*

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568, email: `office@tug.org`). For events sponsored by other organizations, please use the contact address provided.

User group meeting announcements are posted at `tug.org/meetings.html`. Interested users can subscribe and/or post to the related mailing list, and are encouraged to do so.

Other calendars of typographic interest are linked from `tug.org/calendar.html`.