

TUGBOAT

Volume 43, Number 1 / 2022

General Delivery	3	From the president / <i>Boris Veytsman</i>
	4	Editorial comments / <i>Barbara Beeton</i> Robin Fairbairns, 1947–2022; R.I.P. Brent Longborough; TUG 2022 — Online again; HTML versions of T _E X-related documents; Movable metal type; Calling all users of the UK T _E X FAQ
	6	Robin Fairbairns and UK TUG / <i>Jonathan Fine</i>
	7	The last decade at GUTenberg / <i>Jacques André, Patrick Bideault, Denis Bitouzé, Michel Bovani, Thierry Bouche, Maxime Chupin, Daniel Flipo, Yvon Henel</i>
L^AT_EX	10	Markdown 2.15.0: What's new? / <i>Vít Novotný, Dominik Reháč, Michal Hoftich, Tereza Vrabcová</i>
Tutorials	16	The DuckBoat — Beginners' Pond: CDs, but not Compact Disks / <i>Carla Maggi</i>
Graphics	23	Making open source textbooks, and diagrams with AldraTex / <i>Seth Bergmann</i>
Methods	28	Automatically removing widows and orphans with <code>lua-widow-control</code> / <i>Max Chernoff</i>
Software & Tools	40	l3build: The beginner's guide / <i>Joseph Wright</i>
	44	bib2gls: standalone entries and repeated lists (a little book of poisons) / <i>Nicola Talbot</i>
Macros	59	Transparent file I/O using the original T _E X program and the plain T _E X format / <i>Udo Wermuth</i>
Reviews	73	Book review: <i>L^AT_EX Beginner's Guide</i> , second edition, by Stefan Kottwitz / <i>Sarah Lang</i>
Hints & Tricks	75	The treasure chest / <i>Karl Berry</i>
	77	T _E X Live 2022 news / <i>Karl Berry</i>
Abstracts	78	<i>Die T_EXnische Komödie</i> : Contents of issues 4/2021–1/2022
	79	<i>Zpravodaj</i> : Contents of issue 2021/1–4
TUG Business	2	TUGboat editorial information
	2	TUG institutional members
	80	TUG Annual General Meeting Procedures
	81	TUG financial statements for 2021 / <i>Karl Berry</i>
Advertisements	82	T _E X consulting and production services
	83	T _E Xnology Inc.
News	84	Calendar

T_EX Users Group

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

Individual memberships

2022 dues for individual members are as follows:

- Trial rate for new members: \$30.
- Regular members: \$105.
- Special rate: \$75.

The special rate is available to students, seniors, and citizens of countries with modest economies, as detailed on our web site. Members may also choose to receive *TUGboat* and other benefits electronically, at a discount. All membership options are described at tug.org/join.

Membership in the T_EX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. Individual membership carries with it such rights and responsibilities as voting in TUG elections. All the details are on the TUG web site.

Journal subscriptions

TUGboat subscriptions (non-voting) are available to libraries and other organizations or individuals for whom memberships are either not appropriate or desired. Subscriptions are delivered on a calendar year basis. The subscription rate for 2022 is \$115.

Institutional memberships

Institutional membership is primarily a means of showing continuing interest in and support for T_EX 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: April 2022]

Printed in U.S.A.

Board of Directors

Donald Knuth, *Ur Wizard of T_EX-arcana*[†]

Boris Veytsman, *President**

Arthur Rosendahl*, *Vice President*

Karl Berry*, *Treasurer*

Klaus H \ddot{o} ppner*, *Secretary*

Barbara Beeton

Johannes Braams

Paulo Cereda

Kaja Christiansen

Ulrike Fischer

Jim Hefferon

Frank Mittelbach

Ross Moore

Norbert Preining

Raymond Goucher (1937–2019),

Founding Executive Director

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

**member of executive committee*

†honorary

See tug.org/board for a roster of all past and present board members, and other official positions.

Addresses

T_EX 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

T_EXnical support,
public mailing list:
support@tug.org

Contact the
Board of Directors:
board@tug.org

Copyright © 2022 T_EX Users Group.

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

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

Permission is also granted to make, copy and distribute translations of such editorial material into another language, except that the T_EX Users Group must approve translations of this permission notice itself. Lacking such approval, the original English permission notice must be included.

[The author] hopes that nobody will ever scrutinize any of his own writings as meticulously as he and others have examined the ALGOL Report.

Donald E. Knuth

“The Remaining Trouble Spots in ALGOL 60”

Communications of the ACM

(Volume 10, Number 10, October 1967)

TUGBOAT

COMMUNICATIONS OF THE T_EX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 43, NUMBER 1, 2022

PORTLAND, OREGON, U.S.A.

TUGboat editorial information

This regular issue (Vol. 43, No. 1) is the first issue of the 2022 volume year. The deadline for the second issue in Vol. 43 (the TUG'22 conference proceedings) is July 31, 2022, and for the third (regular) issue is October 15, 2022. Contributions are requested.

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

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

TUGboat editorial board

Barbara Beeton, *Editor-in-Chief*
Karl Berry, *Production Manager*
Robin Laakso, *Office Manager*
Boris Veytsman, *Associate Editor, Book Reviews*

TUGboat advertising

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

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, ams.org

Association for Computing
Machinery, *New York, New York,*
acm.org

Aware Software,
Newark, Delaware, awaresw.com

Center for Computing Sciences,
Bowie, Maryland

CSTUG, *Praha, Czech Republic,*
cstug.cz

CTAN, ctan.org

Duke University Press, *Durham,*
North Carolina, dukeupress.edu

Hindawi Foundation, *London, UK,*
hindawi.org

Institute for Defense Analyses,
Center for Communications
Research, *Princeton, New Jersey*

L3Harris, *Melbourne, Florida,*
l3harris.com

L^AT_EX Project, latex-project.org

MacT_EX, tug.org/mactex

Maluhy & Co., *São Paulo, Brazil,*
maluhy.com.br

Marquette University,
Milwaukee, Wisconsin,
marquette.edu

Masaryk University,
Faculty of Informatics,
Brno, Czech Republic, fi.muni.cz

Nagwa Limited, *Windsor, UK,*
nagwa.com

Overleaf, *London, UK,*
overleaf.com

StackExchange,
New York City, New York,
tex.stackexchange.com

T_EXFolio, *Trivandrum, India,*
texfolio.org

Université Laval, *Ste-Foy, Québec,*
Canada, bibl.ulaval.ca

University of Ontario, Institute
of Technology, *Oshawa, Ontario,*
Canada, ontariotechu.ca

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway, uio.no

V_TE_X UAB, *Vilnius, Lithuania,*
vtex.lt

Submitting items for publication

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

The *TUGboat* style files, for use with plain T_EX and L^AT_EX, are available from CTAN and the *TUGboat* web site, and are included in T_EX distributions. We also accept submissions using ConT_EXt. For deadlines, templates, author tips, and more, see tug.org/TUGboat.

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

Other TUG publications

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

If you have such items or know of any that you would like considered for publication, please contact the Publications Committee at tug-pub@tug.org.

From the president

Boris Veytsman

My previous two columns were written about licenses for free software. The topic is vast, and I would like to discuss another aspect of it today.

First, a little bit of history. Gutenberg’s printing press was a revolutionary invention, which radically decreased the cost of the written word. One of the first applications of the invention was the printing of indulgences. One of the first printed indulgences was presumably produced by Gutenberg himself (Figure 1). Indulgences were one of the reasons for Martin Luther’s revolt against the Catholic church. Luther himself understood the power of the printing press very well. He was able to publish his tracts in hundreds of thousands of copies. Luther hired the best printers, including the famous illustrator Lucas Cranach the Elder. However, Luther’s adversaries, both Catholics and competing Protestants, also used the press to spread their texts. Some historians say that the Reformation and Counter-Reformation would not be possible without Gutenberg’s invention.

This leads to an interesting question. Religious wars took millions of lives in the 16th and 17th centuries. Did the widespread use of incendiary pamphlets contribute to this carnage? And if yes, how much blame can we put on the printing press?

Similar things occurred many times. It is common knowledge that most human inventions were quickly used to kill human beings. The domestication of the horse gave rise to cavalry, the invention of the wheel led to military chariots, people learned to fly and started to bomb other people, and so forth. What I would like to stress here is that information technologies are not an exception to this rule. From writing to printing press to telegraph to radio to television to computers — every innovation was quickly used for killing people, or for military propaganda, which also led to killing people. Even our \TeX work may have military uses. I must confess that in my \TeX portfolio there are styles written and paid for by the US Army.

Thus those of us who work in information technologies can be reasonably sure that our work will someday be used to kill. A fresh example: after the vendors of commercial operating systems left Russia, it plans to switch its military to the “Russian OS”, reportedly being a clone of Linux. Can Free Software prevent this? A naïve approach would be to try to add the clause “Thou shalt not kill with this software” to the license. However, free software guidelines such as DFSG explicitly forbid discrimination with respect



Figure 1: Fragment of 31-line indulgence, from en.wikipedia.org/wiki/31-line_Indulgence

to the field of use. If you forbid the use of your software for nefarious aims, you automatically make your software non-free. Are these guidelines reasonable? I think yes. At the end of the day the rule of law is the moral authority. If somebody wants to use your software for amoral purposes, they can easily disregard this clause of your license. On the other hand, such clause may prevent other people from using your work for defense. There was no way to prevent the tables of logarithms, sines and cosines to be used in artillery: once a mathematical theorem is published, it can be used by anybody for any purpose.

I think I can agree with the position of Canonical, the maintainer of Ubuntu (ubuntu.com/blog/canonical-standing-with-ukraine). In response to the Russian invasion of Ukraine, Canonical terminated support, professional services and partnership with Russian users. However, it did not restrict the access for security patches for these users, noting that “free software platforms like Ubuntu, VPN technologies, and Tor, are important for those who seek news and dialogue outside state control”, and directing any subscription income to Ukrainian humanitarian causes.

There are, however, grounds for hope. Besides being used for war propaganda, the printing press started rapid progress of science and literature, that, after all, made us live longer, healthier, and maybe even happier lives. We also can hope that our work at the end of the day will make the world slightly better. We cannot prevent bad people from using our work, but we can hope our work is used by good people.

◇ Boris Veytsman
president (at) tug dot org

From the president

Editorial comments

Barbara Beeton

Robin Fairbairns, 1947–2022

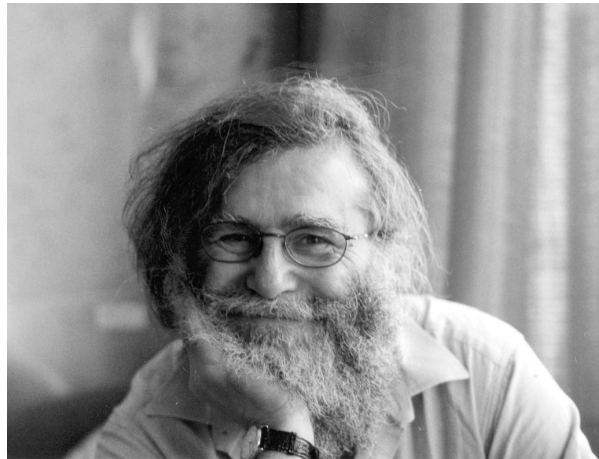
Robin left us on 25 February 2022, after a long period of ill health. An interview by Dave Walden in February 2005¹ tells how he came to the TeX world, even before he joined the Computer Laboratory of the University of Cambridge, bringing (L^A)TeX with him.

At Cambridge, Robin was in charge of many activities related to TeX. Early on, he was an avid TeX user, and actively answered questions on `texhax` and `comp.text.tex`. When the University of Aston determined that they were no longer able to host the UKTeX Archive, Robin rescued it and brought it to Cambridge, where it became one of the original nodes of CTAN. (In turn, this freed Sebastian Rahtz to put together the first versions of TeX Live.²) Robin is listed as (co-)author and/or maintainer of a rather large number of packages on CTAN,³ among them `capt-of`, `covington` (linguistic support), `endnotes`, `footmisc`, `moreverb`, `rotating`, and `setspace`, to mention some that I’ve seen still in use recently.

Always helpful, and a good explainer, Robin (with others) selected the 100 most interesting questions asked on various Internet fora and created the UK-TUG FAQ, which they published in *Baskerville* (*The Annals of the UK TeX Users’ Group*) in December 1994 (vol. 4, no. 6).⁴ The online version moved to Cambridge in 1999, joining CTAN, with Robin as editor. With the maintenance of CTAN centralized under DANTE, the Cambridge site became a mirror, and after Robin’s retirement at the end of December 2014 the FAQ found a new home.⁵ However, much of the content still owes its existence and clarity to Robin.

Robin attended the first meeting of the British user group, UK-TUG, and was persuaded to serve on the committee; he also served as chairman, and was the formal publisher of *Baskerville*. He was active in TUG as well, serving on the board in 1996–1997. He attended several TUG annual meetings, and was editor of the proceedings for two of them: 1995 (St. Petersburg, Florida) and 2000 (Wadham College, Oxford, UK). The *TUGboat* production team was formed during the editing of the 1995 proceedings, with Robin as a founding member; he

remained a member through 2019. Also a *TUGboat* author,⁶ Robin introduced the “New (L^A)TeX 2_ε TUGboat Macros”,⁷ a tutorial for the package he helped to develop, based on his experience with the 1995 proceedings.



Last, but far from least, Robin was a member of the L^ATeX3 Project. The memorial on their site includes this recognition:⁸

Robin was a tireless member of the community who helped many people for many years with his kind, friendly, and patient approach. He clearly looked to improve the TeX ecosystem and worked hard to leave it in a better place after his time was up. He certainly succeeded in that.

I couldn’t agree more, and will miss him greatly.

R.I.P. Brent Longborough

Brent Longborough, an active participant since its inception in TeX.Stackexchange and other (L^A)TeX-related pursuits, left us peacefully on 6 December 2021, surrounded by his family. Brent characterized himself as an “old-ish IT geezer, young at heart”.

Born in Exeter, in Devon, England, he moved with his family to a number of different UK towns and cities, prefiguring his later peripatetic life. He attended Oxford University, concentrating in chemistry, but decided early that this was not for him, and switched to IT, which became his permanent direction.

In 1980, he moved to Rio de Janeiro, working for a British airline. There he mastered the Portuguese language, met his wife, Celia Navarro, and acquired a family.

¹ tug.org/interviews/fairbairns.html
² tug.org/TUGboat/tb18-2/tb55tlguide2.pdf
³ ctan.org/author/fairbairns
⁴ mirrors.ctan.org/usergrps/uktug/baskervi/bask4_6.pdf
⁵ texfaq.org

⁶ tug.org/TUGboat/Contents/listauthor.html#Fairbairns,Robin
⁷ tug.org/TUGboat/tb17-3/tb52guid.pdf
⁸ latex-project.org/news/2022/03/16/robin-fairbairns/

For many years, Brent was employed by IBM, still in Brazil. After leaving IBM, he worked for a Turkish airline, moving with his family to Turkey, becoming conversant in Turkish as well. When he became involved with \TeX , he participated in the development of `arara`, including the translation of `arara` into Turkish.

Brent continued moving around the world, living at various times in Mexico, Colombia and India, returning to the UK after retirement, where he settled with his family in Wales. Languages were well represented in Brent’s skill set, and included French and Spanish (as well as innumerable programming languages), and finally, Welsh. After his return to the UK in 2002, Brent served (in 2013) as a member of the UK-TUG committee.

Brent’s contributions to CTAN were related to the use of `git`, and the package `tagging`, which supported the creation of a source document from which multiple distinct outputs could be generated.

A representative example of his contributions to Stackexchange discussions is his response to the question “Why is \LaTeX so complicated?”, which can be seen at <https://tex.stackexchange.com/a/222505>.

A memorial to Brent was held via Zoom on 23 January 2022, led by his son, Gus Navarro, and attended by many of his friends. Conducted alternately in Portuguese and English, his message honored his father’s care and attention through an eventful life. Remembrances by friends reinforced the memory of these qualities.

Thanks to Paulo Cereda for sharing notes from the Zoom memorial and providing other helpful information.

TUG 2022 — Online again

Owing to the ongoing uncertainty concerning the ability to travel and meet with no restrictions based on COVID-19 status, it has been decided to, once again, conduct the conference online. This will take place from 22–24 July 2022. For details so far visit the conference web page at tug.org/tug2022.

As always, participation is encouraged, both through submission of papers and volunteering to assist with the organization. Instructions for submitting a presentation are on the web page. If you wish to volunteer, send a message to tug2022@tug.org.

HTML versions of \TeX -related documents

Two significant developments have occurred to support (visual) access to \TeX -related material on the web: an HTML version of the PGF/TikZ manual, and HTML output from articles on the arXiv.

The manual was provided by Dominik Peters at <https://tikz.dev>, as announced in January on the [tex.stackexchange](https://tex.stackexchange.com) forum.⁹ It’s searchable, and sections are provided as links identified with ¶. (The graphic of the opening page in the posting is very attractive.)

The second development is the availability of math articles on the arXiv via arxiv.org as “responsive HTML5 web pages” (the “5” is an indication of the HTML version). Translation from the source is accomplished using \LaTeX ML.¹⁰ More details of the project can be found on the arXiv blog.¹¹ We would be pleased to learn your reactions to the output of this project.

The W3C MathML Working Group is working on the problem of audio and tactile accessibility for visually impaired readers. This is even more complicated than conversion to HTML for sighted readers. At an appropriate time, we hope to learn about the problems they’ve encountered and how those were overcome.

Movable metal type

Earlier this year, Boris Veytsman sent to me and several others interested in the printing arts a link to the scans of a beautiful copy of a Gutenberg bible, resident in the Beinecke Rare Book and Manuscript Library at Yale University.¹² Leafing through the pages, I observed that not only was some of the text printed in two colors (black with select passages in red), but ornate initials and delicate marginal decorations were also present. While admiring this handiwork, a followup message arrived from one of the other recipients, admiring the printing workmanship that had produced these marvelous decorations.

Stop! Given the mechanical requirements of printing with movable type, it wasn’t possible that the decorations could be produced on the press. They must be added by hand after the print run. Since I would have expected the individual making these comments to know the process, but clearly that was not the case, it seemed appropriate to present a brief introduction. Here I will be very brief, but with the intention of creating something closer to a “how to” manual at a later time.

Movable type, from Gutenberg’s time (mid-15th century) through the middle of the 19th century, consisted of individually cast metal letters that were

⁹ tex.stackexchange.com/a/630330

¹⁰ math.nist.gov/~BMiller/LaTeXML

¹¹ blog.arxiv.org/2022/02/21/

arxiv-articles-as-responsive-web-pages/

¹² [orbis.library.yale.edu/vwebv/holdingsInfo?bibId=3391099, collections.library.yale.edu/catalog/2020598](https://orbis.library.yale.edu/vwebv/holdingsInfo?bibId=3391099&collections.library.yale.edu/catalog/2020598)

composed into text by hand. In order to keep them in their proper locations, it was necessary to place them on a flat solid surface and constrain them by locking them into a frame, or chase, so that they wouldn't fall apart into "pi", forcing the compositor to start all over, after picking up the little pieces and sorting them into a recognizable order.

Watch this brief video, showing how an 83-year-old Japanese printer carefully selects her type, arranges it, locks it up on the press, and prints a business card.¹³ Then contemplate how much more effort went into the printing of the Gutenberg bible.

Calling all users of the UK TeX FAQ

This is not new news, but may not be known generally, and it can come as a surprise when links pointing to the FAQ come up broken.

The UK TeX FAQ was created under the auspices of UK-TUG. Full publication first occurred in *Baskerville* 4:6, December 1994, as *The New TeX FAQ: Your 100 Questions Answered*.¹⁴ At that time, the home for the FAQ had moved from the online `comp.text.tex` newsgroup to Cambridge University, where it came under the tender care of Robin Fairbairns. When Robin retired, the FAQ was cut loose, but the established url maintained independently, and linked to the new location. This has become too expensive, and the academic ID has been terminated. This leaves a large number of links throughout the Internet without a recognizable target. An effort is being made to update links in online forums or on static pages where this is possible. Of course, it is impossible in such places as mail archives, but since the FAQ itself is still alive (if a bit inactive) anyone knowing of a reference that can be fixed is encouraged to fix it.

Here is the formula. The former syntax of the url was this:

```
http://www.tex.ac.uk/cgi-bin/
    texfaq2html?label-<topic>
```

This should now be

```
https://texfaq.org/FAQ-<topic>
```

Please use the new form when referring to the FAQ, and if you encounter the old url, please correct it if you can, or ask someone who can.

◇ Barbara Beeton
<https://tug.org/TUGboat>

Robin Fairbairns and UK TUG

Jonathan Fine

Dear Members

I'm saddened to hear that Robin Fairbairns has died. I knew that for some time he had health problems. His family, friends and colleagues have my condolences. I write with a mixture of sadness, regrets and gratitude.

About 30 years ago I learnt of his role in the TeX community, and was delighted that he and I lived close to each other in Cambridge. I learnt a lot from him, particularly working together in person on the TeX FAQ, which had recently become the responsibility of the UK TeX Users Group.

In 1994 Chris Rowley as chair of UK TUG reported [Baskerville 5.1, p35]

The notable innovation in 1994 has been the regular appearance of Baskerville. Under the editorship of Sebastian Rahtz, and with Robin Fairbairns and Jonathan Fine as publisher and distributor, issues of our newsletter now reach members approximately every two months.

It contains an interesting variety of TeX-related articles in addition to notices of meetings and subscription forms. In my opinion, Baskerville is the best in content and, by a wide margin, in timeliness of all the journals and newsletters produced by TeX user groups.

Thanks to Sebastian, Robin and Jonathan, and also to all those who have written items for Baskerville.

With the tragic early death of Sebastian in 2016, and Robin this year, of the four persons named in that news story only myself and Chris Rowley are still alive.

◇ Jonathan Fine
 Milton Keynes
 UK
 jfine2358 (at) gmail.com
<https://jfine2358.github.io/>

¹³ [youtube.com/watch?v=UqDU6U01328](https://www.youtube.com/watch?v=UqDU6U01328)

¹⁴ <http://uk.tug.org/wp-installed-content/uploads/2008/12/46.pdf>

The last decade at GUTenberg

Abstract

The French \TeX user group, GUTenberg, has elected a new board in November 2020. Since then, it is coming back to life and serves again the French-speaking community.

But in the meantime, a letter written by the former President of GUTenberg, J r my Just, still a member of the board, was published in *TUGboat* 42:1, pp. 12–13. Unfortunately, it was highly inaccurate and incomplete, and we are exercising our right to reply to it. (Indeed, we would say more about the character of that letter but it is best to avoid raising our voices on these pages.)

1 Local groups: inception and day-to-day work

A local users' group's day-to-day activity is made out of little things: replying to emails; updating a website; updating the members' register; preparing the next issues of the group's publications; caring for a server; doing some bookkeeping; preparing the next gatherings or training sessions; providing software to members; and so on.

These little things need to be sorted out, one by one, so that, one year after the other, the group goes on. It is a delicate mechanism that needs caring people to attend it.

Back in the days before the web, any local \TeX users' group, at its inception, was providing software (and information about it) to its members. The groups were lively: joining them was almost the only way to have the chance to use \TeX .

Then our international community of users made the software programs and their documentation available on the internet, so that anyone could have access to it, freely and instantly. This amazing achievement was due to the step-by-step process described above: some passionate users were dedicating some of their time to it.

2 \TeX 24/7 online availability and its effect on LUGs

When this goal was reached, when everything that was needed to use (\La) \TeX became available 24/7, local users' groups had to prove themselves to still be useful. Some members who had joined before in order to be provided with software and documentation were not to be seen any more.

A decreasing membership affected the fragile mechanism described above. It became even more fragile. And needed even more care than before. More dedication — at the very moment where its

workforce was decreasing. At the very moment where its usefulness was at stake.

It happened here and there. For example in France, at GUTenberg, a decade ago.

Luckily, since then, in most cases, the users' groups have proved useful: the dedication of their members, their good will, the love for \TeX that they spread around showed that the usefulness of a group goes far beyond providing software and documentation. When sharing them, we share much more than them: at the end, it's about looks and smiles, just as any human activity.

And that is the fun of it.

3 An agony

Unfortunately, a decade ago, things went on differently at GUTenberg. When more care was needed, care was diminishing. When lack of care was mentioned, dissent emerged — instead of better care.

It is a classic story, made out of classic issues that could have been fixed by people binding other people together. Minor issues that could have been fixed by patience, care and good will.

But surprisingly, in times when communication can be easy, misunderstanding prevailed. Passionate people left, tired of seeing their ideas not being replied to in the way that they would have expected before.

The fragile mechanism described above wasn't running smoothly any more. The group began to lack people to care for its day-to-day activities. At GUTenberg's board, more and more people were missing, some of them tired from seeing their projects being rebuffed by a president who was lacking time.

GUTenberg's activity was harshly decreasing. Its journal was seldom published — before ceasing publication. Some of its subscribers complained by email; some emails were not even replied to. During those hard times, calls for help were heard; some were granted hearing. Some not. GUTenberg's mandatory annual gatherings were not organized any more. And it is all about looks and smiles, just as any human activity: when the activity itself is missing, the good will fades out quickly. The president was by then alone. Because of his own inability to work with others. It is a pity to read his words about "the atmosphere among the board is far from the friendly one that we had when the board was working smoothly": there was no work and no atmosphere at all. This is how a group dies: when only one person remains, you can't call it a group any more.

No publications. No gatherings. No financial reports sent to the members. No banking activity *at all* in 2019. A few subscriptions remained, but the

joint GUTenberg+TUG subscriptions weren't even reported to TUG. No emails on the board's mailing list from August to December 2018. These are facts. Not "personal point of views", such as the ones in the article that we are replying to.

4 Memory matters

The group was dead. But its memory remained. Former members were still around. Still using T_EX. Lonely, but remembering. And some started to talk together. To talk about the group that they were missing: GUTenberg. Its name was mentioned again. And again. Nothing blazing. But embers. And some breath blown on it. By some people still caring about a group that used to be. Each of them concerned about GUTenberg's unacceptable demise.

We were not many. But we were ... a group.

5 A call for change

On January 17, 2020, an email was sent to the association's only member still in charge:

Subject: AG GUTenberg

Date: Fri, 17 Jan 2020 09:44:53 +0100

Hello Jérémy,

We send you this email as a collective; you will find our names hereunder.

We first thank you for updating GUTenberg's website so that online subscriptions are now available for 2020.

We are well aware that you have spent and still spend a lot of energy for the association and that you are overwhelmed by your professional activity.

Our bylaws provide for an annual general assembly with moral and financial reports. Nevertheless, no general assembly was held since 2013, no report was published since then. In addition to that, the board members are only elected for a four-years term, i.e. since 2017, no one has a valid term any more.

It is therefore urgent to organize a general assembly to decide if we dissolve our association or if we restart GUTenberg on a sound basis.

To this end, it is necessary to have an accurate financial report: a clear view on our assets with a history of our bank transactions, but also a list of our debts to our members who subscribed to services that were not provided, such as our Journal, the *TUGboat* issues etc.

The best would be for you to call this general meeting, in Paris, before the end of

March, where you would present the current state of the association.

In the absence of a positive response from you before the end of January, we would have to call this general meeting ourselves.

Looking forward to reading from you, sincerely,

This message was signed by 8 names, now depicted as a "small group of people". Such a depiction is a common way to make people suspicious. Well, let's introduce those people. By alphabetical order:

Jacques André founder, board member 1988–2007; typography historian, Jacques has contributed often to the association's publications; his books have been reviewed in *TUGboat*, where he has also published articles.

Patrick Bideault co-opted board member, deputy assistant secretary in charge of the memberships paid by PayPal, now president and maintainer of the `coffeestains` package, `texnique.fr` moderator.

Denis Bitouzé board member 2010–2014, maintainer of several packages, co-creator and moderator of `texnique.fr`, organizer of the annual (and famous) Dunkirk L^AT_EX training, co-author of books about L^AT_EX, author of an open access L^AT_EX course and tireless L^AT_EX flag-bearer, now secretary.

Thierry Bouche board member 1997–2017, former editor of *Cahiers GUTenberg*.

Michel Bovani board member 1999–2003, maintainer of the `fourier` package.

Maxime Chupin board member 2009–2017, maintainer of the `bclogo`, `luamesh` and `matapli` packages, now deputy secretary.

Daniel Flipo board member 1993–2005, maintainer of many packages including `babel-french`.

Yvon Henel board member 2009–2017, maintainer of many packages, now deputy treasurer.

We sincerely hope that any suspicion has now vanished. We were only a group of honest people, worried about an association that we had in common, to which we were deeply attached.¹ Please notice that we never *asked* to dissolve GUTenberg. We only *mentioned* that it could be possible, just like Jérémy Just himself did before in several messages; this mention was due to the association's by-then current state.

Then, the horrendous article that we are replying to mentions a shadow cabinet. Yes, we have used

¹ Four among us have applied to the election and are now in charge at GUTenberg.

this expression once. As a joke. And we deeply regret it: to be understood, a joke needs views to be shared by the addressee. But anyone can understand that we have imagined ourselves in charge of GUTenberg. This is why we have published our election platform, which is not even mentioned in Mr Just's text: he prefers mentioning imaginary projects blocked on purpose and the abovementioned nonexistent friendly atmosphere among the by-then nonexistent board.

We recognize that J r my Just has spent time for the association. However, this time spent has slowly but surely led to the demise of the association. He has gradually disgusted many of us from working with him over the past few years.

As a matter of fact, J r my Just does not have the same conception of the association's work as we do: for him, it is mainly the sum of individual activities (his moral assessment of the General Assembly of 2020 was only an assessment of his T Xian activities), whereas for us, it is a collective project, with collective choices, and collective stimulation giving body to the association and making it much more than the sum of the activities of its members.

6 GUTenberg's revival

To relaunch the association, for the General Assembly that we forced to be held, some of us put together a team, wrote a project and presented ourselves for the election of the board.² Their purpose was to rebuild, as a team, and we were glad to be joined by others at the new board, after the election. Nowadays the association serves its community again. But, to our great disappointment, one board member spits in the soup, as we say in French. It is hard for us to understand the acrimonious article we are replying to: why isn't its author happy to have new colleagues in the board? Why isn't he happy to see the association's revival? It's one thing to be overwhelmed by a job that you have applied for without being suited for it (and no one blames anyone for this, as it is very hard to know in advance what it means to serve an association). It's a another thing to blame others when they dedicate themselves to clean up the mess and rebuild the house. This is why we have exercised our right of reply.

We apologize to the readers that are not concerned by the matter. But fake news has to be fought. Always.

GUTenberg is back to life: 4 new issues of its bulletin have been published (see *TUGboat* 42:3, p.313–314), its journal is relaunched (see *TUGboat* 42:3, p.315), it is rewriting its bylaws, has a new server and a new url, `gutenberg-asso.fr`, selected by a poll among its members. Of course, it could do more and better, but at least, it works, and it works collectively.

The French-speaking community is lively, and its T X user group is happy to be serving it again. Long live GUTenberg! Long live TUG and LUGs all over the world! Support your community: join your local TUG!

- ◇ Jacques Andr 
- Founder, GUTenberg board member
1988–2007
- ◇ Patrick Bideault
- President 2020–2022 at GUTenberg
`pb-latex (at) gmx dot fr`
- ◇ Denis Bitouz 
- Secretary 2020–2022 at GUTenberg
- ◇ Michel Bovani
- GUTenberg board member 1999–2003
- ◇ Thierry Bouche
- Former editor-in-chief, *Cahiers GUTenberg*,
board member 1997–2017
- ◇ Maxime Chupin
- Deputy secretary 2020–2022 at GUTenberg
- ◇ Daniel Flipo
- GUTenberg board member 1993–2005
- ◇ Yvon Henel
- Deputy treasurer 2020–2022 at GUTenberg

² See our election platform (in French) at:
`gutenberg-asso.fr/IMG/pdf/liste-gut-renouveau--
profession-de-foi-projets-equipe.pdf`

Markdown 2.15.0: What's new?

Vít Novotný, Dominik Reháč, Michal Hoftich,
Tereza Vrabcová

Abstract

At TUG 2021, we celebrated the fifth birthday of Markdown in \TeX . In this article, we introduce new features developed in the months since, and ideas for the future development of the Markdown package.

The article is divided into three sections. In the first two sections, we introduce the new features of Markdown to the two main audiences of Markdown:

1. the writers, who type content in Markdown, and
2. the coders, who prepare templates and solutions.

In Section 3, we discuss ideas for the future to the third audience of Markdown: the developers, who alter and further improve the Markdown package.

1 Writer's newsletter

Michael Thompson from the `pandoc-discuss` mailing list characterized Markdown as a perfectly minimalist markup language that only faces the writer with one question: what the next sentence should be. [4] However, for some types of documents, the few structural elements of Markdown can be too few. The writers may enable the `hybrid` option and combine \TeX and Markdown markup, but this tends to reduce clarity, stability, and ease of reuse. To reduce the need for hybrid markup, we introduce new syntax extensions for Markdown in sections 1.1–1.4.

Since version 2.10.0 of the Markdown package, writers have been able to redesign their Markdown documents without programming using \LaTeX themes [3]. However, few \LaTeX themes have been publicly available until recently. In Section 1.5, we introduce \LaTeX themes, which self-publishers can use for typesetting books and publishing collaterals.

1.1 Task lists

To track progress on your goals, it can be useful to add checkboxes to list items. Since version 2.11.0, Markdown has supported the `taskLists` option:¹

```
\documentclass{article}
\usepackage[taskLists]{markdown}
\begin{document}
\begin{markdown}
- Tasks:
  1. [x] Draft title.
  2. [.] Draft outline.
  3. [ ] Copy edit.
\end{markdown}
\end{document}
```

Output:

- Tasks:
 - Draft title.
 - Draft outline.
 - Copy edit.

¹ github.com/witiko/markdown/issues/95

1.2 Emphatic line breaks

In poems and plays, line breaks carry a meaning and must be preserved. In Markdown, you can write a line break by ending a line with two or more spaces:

```
Memory and desire, stirring
Dull roots with spring rain.
```

However, this can be tedious for longer texts. Furthermore, the Markdown package only supports line breaks in the `\markdownInput` command, because \TeX strips trailing newlines from the input:²

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
\begin{markdown}
Memory and desire, stirring
Dull roots with spring rain.
\end{markdown}
\end{document}
```

Output:

Memory and
desire, stir-
ring Dull
roots with
spring rain.

Since version 2.12.0, the Markdown package supports the `hardLineBreaks` option,³ which makes every line break emphatic:

```
\documentclass{article}
\usepackage[hardLineBreaks]{markdown}
\begin{document}
\begin{markdown}
Memory and desire, stirring
Dull roots with spring rain.
\end{markdown}
\end{document}
```

Output:

Memory and
desire, stir-
ring
Dull roots
with spring
rain.

This makes it easier to typeset long poems and plays.

1.3 Cross-references

In technical and academic writing, cross-references between sections are common. Previously, writers would need to combine \TeX and Markdown markup:

```
\begin{markdown}
\documentclass{article}
\usepackage[hybrid]{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}
```

² This limitation of \TeX does not apply to Con \TeX t MkIV; see also github.com/witiko/markdown/issues/101.

³ github.com/witiko/markdown/issues/98

Since version 2.14.0, Markdown has supported attributes on section headings and the `relativeLinks` option,⁴ which enables cross-references in Markdown:

```
\begin{markdown}
\documentclass{article}
\usepackage[headerAttributes, relativeLinks]
  {markdown}
I conclude in Section <#sec:conclusion>.
```

```
Conclusion {#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}
```

1.4 Document metadata

Even though writers can prepare their documents in Markdown, they previously needed to specify metadata for their documents (such as the title or the author's name) in \TeX :

```
\begin{markdown}
\documentclass{article}
\usepackage{markdown}
\title{On \emph{Wolves} & \emph{Grandmas}}
\author{Little Red Cap}
\begin{document}
\maketitle
\begin{markdown}
When Little Red Cap entered the woods
a wolf came up to her.
\end{markdown}
\end{document}
```

Since version 2.11.0, the Markdown package has supported the `jekyllData` option,⁵ which allows us to write metadata in Markdown:

```
\begin{markdown}
\documentclass{article}
\usepackage[jekyllData]{markdown}
\begin{document}
\begin{markdown}
---
title: Of *Wolves* & _Grandmas_
author: Little Red Cap
---
When Little Red Cap entered the woods
a wolf came up to her.
\end{markdown}
\end{document}
```

⁴ github.com/witiko/markdown/issues/91

⁵ github.com/witiko/markdown/issues/22

1.5 \LaTeX themes for self-publishers

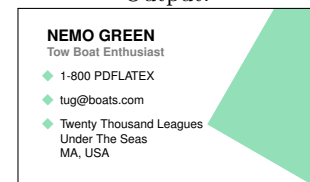
Writers who are unaccustomed to \TeX may find their precious time slipping away, spent scouring online forums looking for a fix for that *one thing* that is messing up the whole layout. In cooperation with the Writersglen publishing house, we have created a set of \LaTeX themes for typesetting books and publishing collaterals in Markdown.⁶

Let's show the ease of use of these templates with an example. Using the business card template, we might end up with a content file looking like this:

```
# Nemo Green
## Tow Boat Enthusiast

- 1-800 PDFLATEX
- tug@boats.com
- Twenty Thousand
  LeaguesLU
  Under The SeasLU
  MA, USA
```

Output:



As you can see, easy as pie! So why not give it a try? (The nice nemo green color is grayscale for the printed *TUGboat*.)

2 Coder's newsletter

In *Digital Typography* [2], Knuth stresses the importance of stability in \TeX and METAFONT, which ensures identical output across time and across different computer systems. Over the last forty years, this stability has allowed an ecosystem of third-party software to grow around \TeX . To make it easier to develop complex software solutions, we show how coders can integrate Markdown with third-party software in sections 2.1–2.3.

In Section 1, we showed new syntax extensions for Markdown. However, syntax extensions are costly to implement, undermine the minimalism of Markdown, and can never account for all components and concepts a writer may need. Therefore in Section 2.4, we present the concepts of *HTML attributes* and *attribute contexts*, which can be used to define domain-specific dialects of Markdown in \TeX without the need for new syntax extensions.

2.1 Building better APIs with YAML

In Section 1.4, we showed how authors can include metadata in their Markdown documents using the YAML language. To react to the metadata, we can use a high-level key–value interface in the `expl3` programming language:⁷

⁶ github.com/xvrabcov/md-templates

⁷ github.com/witiko/markdown/issues/22

```

\documentclass{article}
\usepackage[jekyllData]{markdown}
\ExplSyntaxOn
\tl_new:N \g_abstract_tl
\seq_new:N \g_authors_seq
\keys_define:nn { markdown/jekyllData } {
  abstract .tl_gset:N = \g_abstract_tl,
  /authors/* .code:n = {
    \seq_put_right:Nn
      \g_authors_seq
      { #1 }
  }, title .code:n = {
    \global \title { #1 }
  }, year .code:n = {
    \global \date {
      One~year~after~
      \int_eval:n { #1 - 1 }
    }
  },
}
\markdownSetup {
  renderers = {
    jekyllDataEnd = {
      \exp_args:NNx
        \global \author {
          \seq_use:Nn
            \g_authors_seq
            { \and }
        }
      \maketitle
      \section*{Abstract}
      \g_abstract_tl
    },
  },
}
\ExplSyntaxOff
\begin{document}
\begin{markdown*}{expectJekyllData}
title: 'This is a title: with a colon'
authors: [Jane Doe, John Doe]
year: 2022
abstract: |
  This is the
  abstract

  It contains
  two paragraphs.
\end{markdown*}
\end{document}

```

Output:

This is a title: with a colon	
Jane Doe	John Doe
One year after 2021	
Abstract	
This is the abstract It contains two paragraphs.	

2.2 Passing HTML through to T_EX4ht

Using the T_EX4ht system, we can convert T_EX documents to HTML for publishing on the web. Since

T_EX4ht uses L^AT_EX for the conversion, it supports the Markdown package out-of-the-box. However, it is still necessary to use correct command-line options depending on which T_EX engine we use. To use LuaT_EX, we can use the `--lua` option:

```
$ make4ht --lua document.tex
```

With other T_EX engines, we must use the `--shell-escape` option, which enables shell access:

```
$ make4ht --shell-escape document.tex
```

Since version 2.3.0, Markdown has supported the `html` option, which allows us to use HTML tags in Markdown documents. Since version 2.14.0, Markdown has also supported renderers for HTML tags.⁸ Unless redefined by the user, these renderers will pass any HTML elements through to the output of T_EX4ht, whereas they will be ignored in PDF output:

<code>\usepackage[html]{markdown}</code>	
<code>\begin{document}</code>	
<code>\begin{markdown}</code>	PDF output:
Hello world!	<div style="border: 1px solid black; padding: 2px;">Hello world!</div>
<code>\end{markdown}</code>	T _E X4ht output:
<code>\end{document}</code>	<div style="border: 1px solid black; padding: 2px;">Hello world!</div>

2.3 Integration with Pandoc

Pandoc is a tool for converting between dozens of document formats. In our proof of concept,⁹ we integrate Pandoc with the Markdown package so that we can typeset and style any document format understood by Pandoc directly from T_EX.

To give an example, we have prepared a manual page `wolf.1` in the roff language:

```

.SH NAME
wolf \- tool for befriending grandmas
.SH SYNOPSIS
.B wolf
[\fB-b\fR|\fB--befriend\fR]
[\fB-s\fR|\fB--scare\fR]
<\fIgrandma\fR>

```

Here is how we would typeset our manual page:

```

\documentclass{article}
\usepackage{pandoc-to-markdown, emoji}
\markdownSetup{renderers = {
  headingOne = {\section*{\emoji{wolf}#1}}}
\begin{document}
\pandocInput[format=man]{wolf.1}
\end{document}


```


⁸ github.com/witiko/markdown/issues/90

⁹ github.com/drehak/pandoc-to-markdown

Output:

```

 NAME
wolf - tool for befriending grandmas

 SYNOPSIS
wolf [-b|--befriend] [-s|--scare] <grandma>

```

Our proof of concept consists of a Lua writer that produces \TeX commands corresponding to Pandoc’s abstract syntax tree and a \TeX package that maps these commands to the renderers of the Markdown package. A rewrite of our Lua writer in Haskell will be offered as a basis of the upcoming plain \TeX writer for Pandoc.¹⁰

2.4 Actionable attributes and contexts

In Section 1.3, we showed how authors can add HTML attributes to section headings. We can react to the attributes by redefining attribute renderers. Furthermore, the HTML attributes of a Markdown element are surrounded by attribute contexts, which we can use to limit the effects of an attribute:¹¹

```

\documentclass{article}
\usepackage[headerAttributes]{markdown}
\markdownSetup{
  renderers = {
    headerAttributeContextBegin =
      \begingroup,
    headerAttributeContextEnd =
      \endgroup,
    attributeClassName = {%
      \markdownIfSnippetExists{#1}{%
        \markdownSetup{snippet=#1}%
      }{%
    },
  },
}
\markdownSetupSnippet{sans-serif}{
  code = {%
    \def\familydefault{\sfdefault}%
    \fontfamily{\familydefault}%
    \selectfont
  },
}
\begin{document}
\begin{markdown}
# A section
This section is typeset in a serif typeface.

# Another section {.sans-serif}
This section is typeset in sans-serif ...

```

¹⁰ github.com/jgm/pandoc/issues/1541

¹¹ github.com/witiko/markdown/issues/91

```

## A subsection
... and so is
this subsection.

```

```

# Another section
This section is,
again, typeset
in serif.
\end{markdown}
\end{document}

```

Output:

```

1 A section
This section is typeset in a serif typeface.

2 Another section
This section is typeset in sans-serif ...

2.1 A subsection
... and so is this subsection.

3 Another section
This section is, again, typeset in serif.

```

In Section 3.2, we discuss our plans for other elements of Markdown that may be able to receive HTML attributes in the future.

3 Developer’s newsletter

In the following sections, we describe ideas for improving the Lua parser (3.1 and 3.2), \LaTeX interface (3.3 and 3.4), ConTeXt interface (3.5), and Docker images (3.6) of Markdown.

3.1 Smart backslashes and math support

Since Markdown does not detect math at parse time, it can be difficult to write math:

```

\documentclass{article}
\usepackage{mathtools}
\usepackage[hybrid]{markdown}
\begin{document}
\begin{markdown}
$$$ x_i + y_j =
  \begin{dcases}
    a & b \\
    c & d
  \end{dcases} $$$
\end{markdown}
\end{document}

```

Output:

```

 $x_i + y_j =$ 

$$\begin{cases} a & b \\ c & d \end{cases}$$


```

Specifically, it is necessary to escape underscores and backslashes, and to be careful with indentation:

```

\documentclass{article}
\usepackage{mathtools}
\usepackage[hybrid]{markdown}
\begin{document}
\begin{markdown}
$$$ x\_i + y\_j =
\\begin{dcases}
  a & b \\
  c & d
\\end{dcases} $$$
\end{markdown}
\end{document}

```

Output:

```


$$x_i + y_j = \begin{cases} a & b \\ c & d \end{cases}$$


```

Also in our previous article [3, Figure 4], we showed how we can construct a smart lexical preprocessor that only requires the escaping of backslashes

when they precede another escapable character. Furthermore, we can use well-defined heuristics such as dollar signs to detect math at parse time and disable underscores, code listings, and other elements in it:¹²

```
\documentclass{article}
\usepackage{mathtools}
\usepackage[smartBackslashes, mathDollars]
    {markdown}
\begin{document}
\begin{markdown}
$$ x_i + y_j =
    \begin{dcases}
        a & b \\
        c & d
    \end{dcases}
$$
\end{markdown}
\end{document}
```

Desired output:

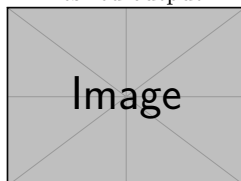
$$x_i + y_j = \begin{cases} a & b \\ c & d \end{cases}$$

3.2 Attributes on links and images

In Section 1.3, we showed how authors can add HTML attributes to headings in Markdown. In order to define domain-specific dialects of Markdown in \TeX , it may be useful to support HTML attributes on various other elements of Markdown, perhaps most importantly on links and images:¹³

```
\documentclass{article}
\usepackage[linkAttributes]{markdown}
\usepackage{graphicx}
\markdownSetup{
  renderers = {
    linkAttributeContextEnd =
      \endgroup,
    linkAttributeContextBegin = {%
      \begingroup
      \markdownSetup{
        renderers = {
          attributeKeyValue = {%
            \setkeys{Gin}{
              {#1} = {#2},
            }%
          },
        },
      }%
    },
  },
}
\begin{document}
\begin{markdown}
! [image] (example-image){width=5cm}
\end{markdown}
\end{document}
```

Desired output:



¹² github.com/witiko/markdown/issues/61

¹³ github.com/witiko/markdown/issues/123

3.3 Importing \LaTeX setup snippets

In our previous article [3, Section 1], we have introduced \LaTeX themes and snippets, which can be used to build powerful abstractions in Markdown. Suppose the `jdoo/longpackagename/lists` \LaTeX theme defines the `arabic`, `roman`, and `alpha` setup snippets. If we want to access these snippets by their short names, we must first load the theme and then assign names to the snippets:

```
\markdownSetup{
  theme=jdoo/longpackagename/lists}
\markdownSetupSnippet{arabic}{
  snippet=jdoo/longpackagename/lists/arabic}
\markdownSetupSnippet{roman}{
  snippet=jdoo/longpackagename/lists/roman}
\markdownSetupSnippet{alphabetic}{
  snippet=jdoo/longpackagename/lists/alpha}
```

In order to make the code easier to read and the intent clearer, it may be useful to have a dedicated syntax for importing setup snippets:¹⁴

```
\markdownSetup{
  importSnippets = {
    jdoo/longpackagename/lists = {
      arabic,
      roman,
      alpha as alphabetic,
    },
  },
}
```

3.4 Advanced renderer definitions in \LaTeX

At the moment, the `\markdownSetup` \LaTeX command only allows the redefinition of one renderer or renderer prototype at a time, which makes it difficult to redefine several renderers or renderer prototypes at once:

```
\markdownSetup{
  rendererPrototypes = {
    headingOne = {\chapter{🐱 #1}},
    headingTwo = {\section{🐱 #1}},
    headingThree = {\subsection{🐱 #1}},
    headingFour = {\subsubsection{🐱 #1}},
    headingFive = {\paragraph{🐱 #1}},
    headingSix = {\subparagraph{🐱 #1}},
  },
}
```

Furthermore, it is difficult to keep some parts of previous definitions without using low-level code:

```
\usepackage{etoolbox}
\patchcmd
  \markdownRendererHeadingOnePrototype
  {#1}{🐱 #1}{}{}
```

¹⁴ github.com/witiko/markdown/issues/107

In order to make it easier to redefine renderers and renderer prototypes partially and in bulk, it may be useful to extend the syntax of `\markdownSetup`:¹⁵

```
\markdownSetup{
  rendererPrototypes = {
    heading* {#1} = {#1},
  },
}
```

3.5 Idiomatic ConTeXt setup

Unlike L^AT_EX, which has high-level syntax for setting up Markdown, ConTeXt has only a few additions over the plain T_EX interface for Markdown. Since version 2.15.0, there has been a concerted effort to extend Markdown, so that it can enumerate and examine its own options, renderers, and renderer prototypes.¹⁶ This will make it easier to create and maintain new high-level interfaces for formats other than L^AT_EX, such as ConTeXt.¹⁷

3.6 Additional binary platforms in Docker

Since version 2.10.0, Markdown has been available as the `witiko/markdown` Docker image.¹⁸ In version 2.15.0, images for T_EX Live 2019–2021 are available, which makes it easy to use Markdown for continuous integration with services such as GitHub Actions:

```
name: Typeset a document
on: {push: ~}
jobs:
  typeset:
    runs-on: ubuntu-latest
    container:
      image: witiko/markdown:TL2019-historic
    steps:
      - uses: actions/checkout@v2
      - run: latexmk -lualatex document.tex
```

The `witiko/markdown` Docker image is based on the `texlive/texlive` Docker image from the Island of T_EX [1], which is only available for the `linux/amd64` platform. This is sufficient for most continuous integration services. However, to allow interactive use of `witiko/markdown`, it may be useful to add support for multi-platform builds to `texlive/texlive`.¹⁹

References

- [1] Island of T_EX. The Island of T_EX: Developing abroad, your next destination. *TUGboat* 41(2):182–184, 2020. tug.org/TUGboat/tb41-2/tb128island.pdf
- [2] D. Knuth. *Digital Typography*. No. 78 in CSLI Lecture Notes. Center for the Study of Language and Information (CSLI), 1999. The second printing (2012) contains numerous corrections.
- [3] V. Novotný. Markdown 2.10.0: L^AT_EX themes & snippets, two flavors of comments, and LuaMetaT_EX. *TUGboat* 42(2):186–193, 2021. tug.org/TUGboat/tb42-2/tb131novotny-markdown.pdf
- [4] M. Thompson. Re: Error in "cabal install pandoc", 2010. groups.google.com/g/pandoc-discuss/c/tKB4E7y6H2E/m/OiieKAuWsl4J

- ◇ Vít Novotný
Studená 453/15
Brno, 638 00
Czech Republic
`witiko (at) mail dot muni dot cz`
github.com/witiko
- ◇ Dominik Reháč
Legionárska 71
Trenčín, 911 04
Slovak Republic
`drehak (at) firemail dot cc`
github.com/drehak
- ◇ Michal Hoftich
Magdalény Rettigové 4
Praha, 116 39
Czech Republic
`michal dot h21 (at) gmail dot com`
github.com/michal-h21
- ◇ Tereza Vrabcová
V Aleji 130/30
Děčín, 405 02
Czech Republic
`vrabcova dot tereza (at) email dot cz`
github.com/xvrabcov

¹⁵ github.com/witiko/markdown/issues/121

¹⁶ github.com/witiko/markdown/issues/119

¹⁷ github.com/witiko/markdown/issues/17

¹⁸ hub.docker.com/r/witiko/markdown

¹⁹ gitlab.com/islandoftex/images/texlive/-/issues/15

The DuckBoat — Beginners' Pond: CDs, but not Compact Disks

Herr Professor Paulinho van Duck

Abstract

In this installment, Prof. van Duck will show you some tips & tricks about `tikz-cd`, a package for high-quality typesetting of commutative diagrams.

1 \TeX .SE has a new moderator!

Hi, \LaTeX friends!

An exceptional event took place last November: a \TeX .SE moderator's election, quack!

Moderators are elected for life because they are somewhat compared to Supreme Court judges, who are appointed lifelong (in some countries, at least). Therefore, their elections are quite infrequent.

Of course, the role of moderator is not a life sentence. There is also the possibility of resigning.

Our former moderator Martin Scharrer made that choice last October. Martin had been in that role since \TeX .SE left beta in 2011. He is also the author of many valuable packages, like `standalone` and `adjustbox`. We are immensely grateful to him for all his past and future contributions to the \LaTeX community.

I would also like to thank the friends who nominated themselves as candidates in the moderator election. There were a couple of well-known names but even relatively new users. I am particularly glad to have seen people from non-Western cultures, and I hope to have a woman, too, next time.

It was very hard to choose. Eventually, the winner was Werner Grundlingh! (Simply known as *Werner* on \TeX .SE: tex.stackexchange.com/users/5764/werner).

He needs no introduction for the \TeX .SE users since he is the third in the site ranking, after egreg and David Carlisle. He has always stood out for his patience, good temper, and his craving for helping others. Let me congratulate him. He has the skills to be a great mod, quack!



Unfortunately, there is also bad news. Our dear friend Brent Longborough passed away in December 2021. He gave an outstanding contribution both on \TeX .SE and with his packages and collaborations (for example, `arara`).

I like to remember his great sense of humor. He defined himself “old-ish IT geezer, young at heart.”

We will miss him!



Herr Professor Paulinho van Duck

doi.org/10.47397/tb/43-1/tb133duck-tikz-cd

As for TopAnswers \TeX , <https://topanswers.xyz/tex>, the site grows and starts appearing in browsers' search results. I suggest you try it. In particular, if your question on other sites remains unanswered, you can be sure it will be looked after on TopAnswers \TeX . Moreover, if you have problems with `beamer`, you will find the great expert samcarter answering there.

I take the occasion to thank her for her help setting the correct fonts in the example of Box 14.



If you are talking with a mathematician about CDs, you almost certainly are not referring to an old-fashioned(!) way to listen to music.

The acronym CDs stands for Commutative Diagrams. My math colleagues say they are graphs largely used in category theory. I do not know what category theory is, but I did not ask, in order to avoid a two-hour math lesson, quack!

However, judging by the number of questions about them, they are undoubtedly prevalent.

Box 1 shows a little example. It is the representation of $h \circ f = k \circ g$.

Box 1 – A commutative diagram

```
...
\usepackage{tikz-cd}
...
\begin{document}
\[
\begin{tikzcd}
A \ar[r,"f"] \ar[d,"g"] & B \\
C \ar[r,"k"] \ar[u,"h"] & D
\end{tikzcd}
\]
\end{document}
```

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ g \downarrow & & \downarrow h \\ C & \xrightarrow{k} & D \end{array}$$

Do you find it beautiful? Of course, you do!

In the following, I will show you how to create it with `tikz-cd`. There are also other packages for drawing commutative diagrams, for instance `xy`, but in my opinion `tikz-cd` is the simplest and the most customizable.

2 Quack Guide No. 7

A package for typing beautiful CDs

In a previous article of mine [1], I told you about `TikZ` matrices. Simply speaking, a `TikZ` CD is a `TikZ` matrix with some arrows added.

Let us start examining the code in Box 1.

First of all, you have to load the package. Alternatively, if you have already loaded TikZ in your preamble, you may take advantage of the `cd` library:

```
\usepackage{tikz}
\usetikzlibrary{cd}
```

Now you are ready to use the `tikzcd` environment. Please note that I have included it within `[...]` to show it as a displayed equation.

The core of your CD is typed like an ordinary `array`, with `&` to separate the cells (the vertices of the diagram) and `\\` to separate the rows. Pay attention that the last row does not end with `\\`; otherwise, you will get an undesired empty line.

The new additions are the commands `\ar`, an abbreviation for `\arrows`. As it is obvious from their name, they allow you to draw your arrows, quack!

The first parameter of `\ar` is the direction: `r`, `l`, `d`, and `u`, which stand for right, left, down, up. You can combine two or more directions. For example, `dr` means your arrow will go diagonally from the current cell, where the `\ar` is positioned, to the one down and right from it. You can also repeat a direction. For instance, `rr` will go from the current cell to the second cell right to it.

What appears between quotes are the labels of the arrows. The option `swap` makes the label be placed on the right side of the arrow, relative to its direction (left is the default).



I will describe how to build and customize your diagram with the available options in the following.

Remember that, in general, options can be set either for a single element; or for the current CD as environment options:

```
\begin{tikzcd}[\langle options \rangle]
```

or for all the CDs of your document by putting:

```
\tikzcdset{\langle options \rangle}
```

in your preamble.

See Section 2 of the package documentation [3] for further details.

2.1 About vertices

Since a TikZ CD is actually a TikZ matrix, it inherits all matrix characteristics.

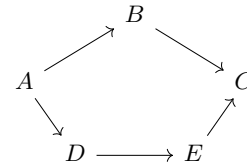
You can set the distance separating columns with `&[\langle width \rangle]` and for rows with `\\[\langle width \rangle]`.

If you would like to change these separators for all the rows/columns of your diagram, you can set them with the options `column sep/row sep`.

You may indicate either an explicit dimension or the predefined ones: `tiny`, `small`, `scriptsize`, `normal`, `large`, and `huge`.

Box 2 – Column and row separators

```
\begin{tikzcd}[column sep=4pt,
row sep=scriptsize
&&[3pt]B\ar[dr]&&[3pt]\\
A\ar[urr]\ar[dr]&&&&C\llbracket[2pt]
&D\ar[rr]&&E\ar[ur]
\end{tikzcd}
```

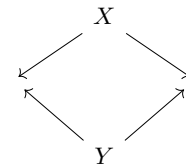


Please note how, in Box 2, the empty cells are used to position the vertices correctly. But be careful that if a cell is a target point of an arrow, it must have a text. Otherwise, you will get the error:

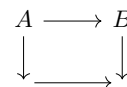
No shape named ... is known.

If you want to have an arrow pointing to an empty vertex, use `{}` for one vertex at a time, or the matrix option `nodes in empty cells`, for all the vertices at once.

```
\begin{tikzcd}
&X\ar[d]\ar[dr]\\
{}&&{}\\
&Y\ar[ul]\ar[ur]
\end{tikzcd}
```



```
\begin{tikzcd}[
every matrix/.append
style={nodes in
empty cells}
]
A\ar[r]\ar[d]&&B\ar[d]
\ar[r]&&
\end{tikzcd}
```



If you would like to customize one vertex (for example, to draw it), you can use `|[\langle options \rangle]` before the node text:

```
\begin{tikzcd}
A\ar[r]&|[\langle draw \rangle]B
\end{tikzcd}
```



If you would like to modify all the vertices of your diagram, you can use `cells={\langle options \rangle}`, which is equivalent to `every cell/.append style={\langle options \rangle}`

whereas

`every cell/.style={\options}`
replaces the default style of the vertices instead of adding options to it.

```
\begin{tikzcd}[
  cells={nodes=draw}]
  A\ar[r]&B
\end{tikzcd}
```



One useful node option is `alias=\langle cell name \rangle`. It allows you to refer to the vertex with `\langle cell name \rangle` when you are drawing your diagram.

Another way for referring to the vertices is `\langle matrix name \rangle-\langle row number \rangle-\langle column number \rangle` where `\langle matrix name \rangle` could be `\tikzcdmatrixname`

or a name you set with `every matrix/.append style={name=\langle matrix name \rangle}`

Within a path you can also use `\tikztostart` (starting point of the path) and `\tikztotarget` (target point of the path).

In Section 2.4 I will show how node naming is indispensable for drawing complex diagrams.

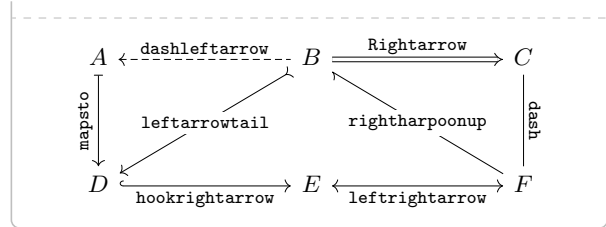
2.2 Go wild with the arrows

The package `tikz-cd` allows a huge number of arrow tips and shapes.

Box 3 shows some examples. For the complete list, see Section 1.3 of the package documentation.

Box 3 – Some arrow types

```
\begin{tikzcd}[
  labels={font=\ttfamily\scriptsize},
  column sep=7em, row sep=9ex
]
A\ar[r, dashleftarrow,
  "\text{dashleftarrow}"]
\ar[d, mapsto, "\text{mapsto}"
  {rotate=90, anchor=south}]&
B\ar[r, Rightarrow, "\text{Rightarrow}"]&
C\ar[d, dash, "\text{dash}"
  {rotate=-90, anchor=south}]\
D\ar[r, hookrightarrow,
  "\text{hookrightarrow}", swap]
\ar[ur, leftarrowtail,
  "\text{leftarrowtail}" description]&
E\ar[r, leftrightarrow,
  "\text{leftrightarrow}", swap]&
F\ar[ul, rightharpoonup,
  "\text{rightharpoonup}" description]
\end{tikzcd}
```



You can also set options for arrows, at either the environment or document level, with `arrows=\langle options \rangle` which appends options to the default, or `every arrow/.style=\langle options \rangle` which replaces the default.

Let us examine some of them (label options will be explained in the next Section).



If the arrows of `tikz-cd` are not enough for you, you may use all the types provided by TikZ, setting `arrow style=tikz`.

For example, if you would like to have stealth arrows throughout your document, you can set

```
\usetikzlibrary{arrows.meta}
\tikzcdset{
  arrow style=tikz,
  arrows={>={Stealth}}
}
```

in your preamble, and all your arrows will appear like this:



When your CD has many vertices, it may be boring or difficult to indicate the correct number of `r`, `l`, `d`, and `u` you need to reach the target point.

The arrow options `to/from=\langle argument \rangle` may help you. The argument could be in the form `\langle row number-column number \rangle`; a string of `r`, `l`, `d`, and `u`; or `\langle cell name \rangle`.

You may even use `to` and `from` together and draw your arrows at the end of the diagram.

The following code is an alternative way to produce the diagram in Box 2:

```
\begin{tikzcd}[
  column sep=3pt,
  row sep=scriptsize
]
&&[3pt]B\ar[dr]&&[3pt]
A\ar[to=1-3]&&&
|[alias=nodeC]|C\ar[from=nodeE]\[2pt]
&&D\ar[from=ul]&&
|[alias=nodeE]|E\ar[from=3-2]
\ar[from=1-3, to=nodeC]
```

```
\end{tikzcd}
```



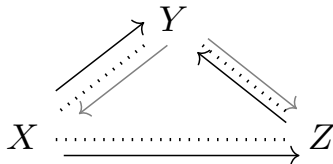
It may happen that you need to shift your arrows to better position them. You can easily do this with `shift left/right=<dimension>` (the parameter is optional).

It is also possible to use `xshift=<dimension>`, `yshift=<dimension>` or `shift={<coordinate>}`.

Box 4 shows some shiftings. The dotted lines represent the default positions. Some arrows are drawn in gray and the CD is enlarged with `\scalebox` to better visualize the positioning. For the `ampersand` replacement option see Section 2.4, Box 14.

Box 4 – Shifting

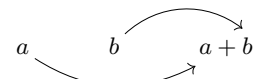
```
\scalebox{1.5}{%
\begin{tikzcd}[ampersand replacement=&]
  \&Y\ar[dr, shift left, gray]
  \ar[from=dr, shift left]
  \ar[dr, dash, dotted, thick]
  \ar[d1, xshift=1.5ex, gray]
  \ar[from=d1, yshift=1.5ex]
  \ar[d1, dash, dotted, thick]
  X\ar[rr, shift={ (2pt,-4pt) }]
  \ar[rr, dash, dotted, thick] \&\&Z
\end{tikzcd}%
}
```



You can also explicitly set a starting/ending point with `start/end anchor=` `{[<coordinate transformations>]}<anchor>`

Box 5 – Start/end anchor

```
\begin{tikzcd}
  a\ar[start anchor={ [xshift=-2pt,
  yshift=2pt] south east },
  end anchor=210, bend right=30, rr]
  & b\ar[start
  anchor={ [shift={ (-2pt,-2pt) } ] north
  east },
  end anchor=50, bend left=40, r]
  & a+b
\end{tikzcd}
```



As you see in Box 5, if you are tired of straight arrows, you can use options inherited from TikZ to bend them:

- `bend right/left=<angle>`, to curve the arrow
- `out/in=<angle>`, to set the angle at which the arrow leaves/reaches the vertices
- `loop`, possibly with `above/right/below/left` or setting the in/out angles
- `looseness=<number>` to choose the “level of bending”.

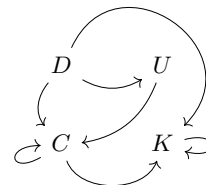
For more details, see Sections 74.3 and 74.4 of the TikZ manual [4].

Box 6 shows some of these options. Please note the environment option `bezier bounding box`. It is useful to set the correct bounding box when there is `looseness` in order to avoid unwanted empty space around your diagram. To use it just add the `bbox` TikZ library [2] in your preamble:

```
\usetikzlibrary{bbox}
```

Box 6 – Bent arrows

```
\begin{tikzcd}[bezier bounding box]
  D\ar[r, bend right]
  \ar[dr, bend left=100, looseness=2]
  \ar[d, out=230, in=130]
  & U\ar[d1, bend left]
  C\ar[r, bend right=70]
  \ar[loop, in=180, out=210, looseness=8]
  & K\ar[loop right]
\end{tikzcd}
```

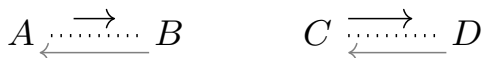


Another useful option is `shorten=<dimension>`. It allows to shorten or, if `<dimension>` is negative, elongate your arrow.

This option acts on both sides of the arrows, if you would like to shorten/elongate only one side, use the standard TikZ options `shorten >=<dimension>` or `shorten <=<dimension>`.

Box 7 – Shorten/elongate arrows

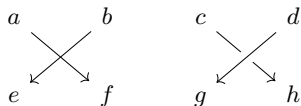
```
\scalebox{1.5}{\begin{tikzcd}[ampersand
replacement=\&]
A\ar[r, dash, dotted, thick]
\ar[r, shift left=1.5mm, shorten=.2cm]
\&B\ar[l, shift left=1.5mm, shorten=-.1cm,
gray]
\&C\ar[r, dash, dotted, thick]
\ar[r, shift left=1.5mm, shorten >=.2cm]
\&D\ar[l, shift left=1.5mm, shorten
<=-.1cm, gray]
\end{tikzcd}}
```



Finally, it is possible to use **crossing over** to show an arrow in foreground. Compare:

Box 8 – Crossing over

```
\begin{tikzcd}
a\ar[dr]&\&b\ar[dl]&
c\ar[dr]&\&d\ar[dl, crossing over]&
e&\&f&
g&\&h
\end{tikzcd}
```



2.3 Adding labels

The general syntax for labels is `"{label text}"{label options}` where the curly braces are mandatory only if `label text` or `label options` contain a comma.

You can set the label options at the environment level specifying `labels={options}` which appends the style to the default, or `every label/.style={options}` which replaces the default style.

When including them in `\tikzcdset` in your preamble, they will be valid at the document level, as usual.

In Box 3, for example, I set the label font at the environment level. Note that since the labels are typed in math mode by default, to have them in text mode you have to use `\text{label text}`.

The options `rotate={degree}` and `anchor={anchor}` are necessary to put the label along the vertical ar-

rows. If you prefer to write them over the arrow, just use `description` (label text with a white background). You can change the background color with `background color={color}` or do not have it at all using `marking`.

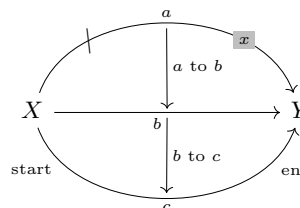
The TikZ options for positioning nodes along a path could also be used (`near start`, `near end`, and so on).

Since labels are nodes, you can even name them and use them as starting or target point of an arrow.

Please note that you can also have more than one label on the same arrow.

Box 9 – Labels

```
\begin{tikzcd}[column sep=3cm]
X\ar[r, "b"{name=B, below left},
" "{name=middle, inner sep=0pt}]
\ar[r, bend left=70, "a"{name=A,
"/" {marking, near start},
"x" {description, near end, background
color=lightgray}]
\ar[r, bend right=70, "c" {name=C, below},
"\text{start}" {very near start, swap},
"\text{end}" {very near end, swap}]
& Y
\ar[from=A, to=middle, shorten <=2pt,
"a\text{ to }b"]
\ar[from=middle, to=C, shorten =2pt,
"b\text{ to }c"]
\end{tikzcd}
```



See Section 2 of the package documentation [3] for further details.

2.4 Other tips & tricks

Sometimes your vertices are not simply letters but longer expressions. Maybe you do not like the standard positioning:

```
\begin{tikzcd}
a+b\ar[r]\ar[d]&
c\ar[d]&
d\ar[r] & e-f
\end{tikzcd}
```

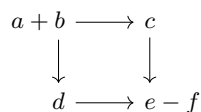
There are ways to change the node alignment and modify the arrow path.

The former may be achieved, for example, taking advantage of the TikZ possibility to set the matrix's column style.

For the latter, the appropriate coordinates can be used in a `from/to` or the path can be designed with `to path`.

Box 10 – Column style

```
\begin{tikzcd}[
  /tikz/column 1/.append
    style={nodes={anchor=base east}},
  /tikz/column 2/.append
    style={nodes={anchor=base west}}
]
|[alias=first]|a+b\ar[r]\ar[d,
  from={first.south -| second}]&
c\ar[d, to path={--
  (\tikztotarget.north -|
  \tikztostart)}]\ \
|[alias=second]|d\ar[r] & e-f
\end{tikzcd}
```



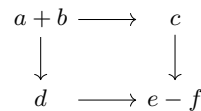
Here, as usual in TikZ, `-|` means “horizontal coordinate of the point before `-` and vertical coordinate of the point after `|`”. For further details see Section 3.2 of the package documentation [3].

If the standard alignment is OK for you, but you would like, for example, to have the same length of the horizontal arrows without calculating it manually, you can set the node width to a given dimension.

In the following, the width of the longest node text is used, taking advantage of the option `text width=width(<string>)`.

Box 11 – Node width

```
\begin{tikzcd}[
  /tikz/column 1/.append style={
    nodes={text width=width("$a+b$"), text
    centered}},
  /tikz/column 2/.append style={
    nodes={text width=width("$e-f$"), text
    centered}}
]
a+b\ar[r]\ar[d]& c\ar[d]\ \
d\ar[r] & e-f
\end{tikzcd}
```



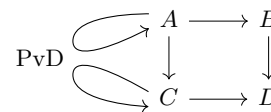
For complex diagrams, it is possible to add code after your diagram is drawn, with `remember picture and overlay` or `execute at end picture`

The difference between these two methods is that what you draw with the former is not included in the bounding box of the picture. To visualize this, compare the alignment of the following two examples.

In Box 12, the entire picture is centered, including what gets drawn at end (the “PvD” and curves).

Box 12 – Execute at end picture

```
\begin{tikzcd}[
  every matrix/.append style={name=mycd,},
  execute at end picture={
    \node[left=of \tikzcdmatrixname]
      (leftofm) {PvD};
    \draw[->] plot [smooth, tension=7]
      coordinates
        {(\tikzcdmatrixname-1-1.160)
        (leftofm.north east)
        (\tikzcdmatrixname-1-1.200)};
    \draw[->] plot [smooth, tension=7]
      coordinates
        {(mycd-2-1.160) (leftofm.south east)
        (mycd-2-1.200)};
  }
]
A \ar[r]\ar[d] & B \ar[d] \ \
C \ar[r] & D
\end{tikzcd}
```



In Box 13, only the actual CD is centered, what is added afterwards is not.

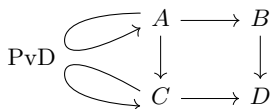
Box 13 – Remember picture and overlay

```
\begin{tikzcd}[
  every matrix/.append style={name=mycd,},
  remember picture
```

```

]
A \ar[r]\ar[d] & B \ar[d] \\
C \ar[r]      & D
\end{tikzcd}%
\begin{tikzpicture}[
  overlay, remember picture
]
\node[left=of \tikzcdmatrixname]
  (leftofm) {PvD};
\draw[->] plot [smooth, tension=7]
  coordinates
  {(\tikzcdmatrixname-1-1.160)
  (leftofm.north east)
  (\tikzcdmatrixname-1-1.200)};
\draw[->] plot [smooth, tension=7]
  coordinates
  {(mycd-2-1.160) (leftofm.south east)
  (mycd-2-1.200)};
\end{tikzpicture}

```



Choose one or the other method depending on your needs.



Last but not least, when you use a `tikzcd` in a `beamer` frame or as argument to a command, you get the error:

Single ampersand used with wrong catcode.

You could also have strange errors or undesired alignments when your vertices or labels are math matrices or text tables or, in general, any environment that uses `&`.

These problems can be easily solved using `ampersand replacement=<macro name>` and then using `<macro name>` instead of `&`.

Box 14 – Ampersand replacement

```

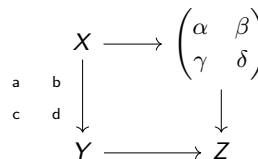
\documentclass{beamer}
\usepackage{tikz-cd}
\tikzcdset{ampersand replacement=\&}
\begin{document}
\begin{frame}
\[\
\begin{tikzcd}
X\ar[r]\ar[d,"\text{\begin{tabular}{cc}
a & b \\ c & d
\end{tabular}}"]
& \begin{pmatrix}
\alpha & \beta \\
\gamma & \delta
\end{pmatrix}
\end{tikzcd}
\]
\end{frame}
\end{document}

```

```

\end{pmatrix}\ar[d] \\
Y\ar[r] \& Z
\end{tikzcd}
\]
\end{frame}
\end{document}

```



3 Conclusions

If you have been affected by Stendhal syndrome after looking at the commutative diagrams drawn with `tikz-cd`, remember:

*For a quack math,
ask a TikZ duck!*

References

- [1] C. Maggi. The DuckBoat — Beginners' Pond: You do not need to be Neo to cope with a TikZ matrix. *TUGboat* 41(1):20–25, 2020. tug.org/TUGboat/tb41-1/tb127duck-matrix.pdf
- [2] marmotghost. *Bounding boxes for Bézier curves*. mirrors.ctan.org/graphics/pgf/contrib/tikz-bbox/pgfmanual4bbox.pdf. Package page: ctan.org/pkg/tikz-bbox.
- [3] A. Stoffel. `{tikzcd}` — *Commutative diagrams with TikZ. Version 1.0*. mirrors.ctan.org/graphics/pgf/contrib/tikz-cd/tikz-cd-doc.pdf. Package page: ctan.org/pkg/tikz-cd.
- [4] T. Tantau. *The TikZ and PGF packages. Manual for Version 3.1.9a*. mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf. Package page: ctan.org/pkg/pgf.

◇ Herr Professor Paulinho van Duck
Quack University Campus
Sempione Park Pond
Milano, Italy
[paulinho dot vanduck \(at\) gmail dot com](mailto:paulinho.vanduck@gmail.com)

Making open source textbooks, and diagrams with AIDraTex

Seth D. Bergmann

Abstract

This article describes a new paradigm for the creation of textbooks, using \LaTeX . Macros for the automatic generation of figures and diagrams are described.

1 Introduction

Textbooks have traditionally been produced by publishing companies which provide many services, and revenue, to authors. These services include:

- Editors responsible for the acquisition and production process
- Copyeditors responsible for formatting, proof-reading the text for errors and making stylistic improvements
- Graphic artists responsible for the production of technical figures and diagrams
- Sales representatives responsible for the marketing of the book

All these services, plus other corporate employees, bricks and mortar, etc., contribute to the extremely high cost of textbooks for readers.

This article proposes a new paradigm for the creation of textbooks, based on the *open source* model of software development [2]. Open source software is developed by many developers who may be at remote locations, cooperating on the Internet. In addition to distributing the software product, the developers also distribute the source documents, making it easy for others to make corrections, enhancements, or extensions to the original product. Since there is no cost for the product, there is no revenue for the developers. Some common examples of open source software: Java, Apache, and \LaTeX .

Developers who contribute to a well-known successful product are rewarded by the status associated with the product, and can perhaps leverage this status with job offers or promotions.

In applying the open source paradigm to textbooks, authors can collaborate on the generation of high-quality books. As the books become widely adopted, errata are found and corrected, and other authors are able to add sections and chapters to improve the book. The authors (usually teachers or professors, but also students) receive no direct compensation, but if the book is widely adopted, they receive recognition for their contributions, which can be a positive factor in applying for jobs, tenure, promotion, etc.

The most significant impediment to open-source textbooks is the production of high quality figures and diagrams. This article makes the claim that diagrams can be drawn with \LaTeX macros, rather than traditional drawing tools. The distinction is that I am proposing the use of a markup language as opposed to WYSIWYG tools.

The development of the graphics macros can be a difficult, and time-consuming process. However, once the macros are perfected, they can be used to draw many diagrams with a professional appearance. If a minor stylistic change is needed, (e.g., arrowheads in state diagrams need to be enlarged) all that is needed is a change to the macro that draws state diagrams, and all such diagrams are regenerated automatically.

When several authors are collaborating on a textbook, it is not necessary that they all have the expertise needed to develop new macros. This could be the responsibility of one developer, with expertise in the graphics commands, to produce new macros as needed by the other authors.

This paper lists four currently available open source textbooks (and a fifth in development) and shows some examples of the diagrams that are generated automatically.

2 Some currently available open source textbooks

I have developed four open source textbooks, which are currently being distributed by the Campbell Library at Rowan University, with a total of over 40,000 downloads worldwide. They are available at rdw.rowan.edu/oer:

Title	Downloads
Compiler Design: Theory, Tools, and Examples [3]	25,843
Computer Organization with MIPS [4]	11,739
Introduction to Computer Science with Java [6]	4,757
Computer Science Principles (Java, Python, and C++ editions) [5]	1,398
Cryptology with Bitcoin and Blockchain [in development]	0

(Number of downloads as of March 2022.)

The source files for these books, along with recently updated versions, are available from the author's web site: cs.rowan.edu/~bergmann/books.

I'd like to single out Allen Downey, at Olin College of Engineering, who has also published several open source textbooks [7]. He has been one of the first, if not the first, to promote this new paradigm, making educational resources affordable for students and school districts everywhere.

Another developer of open source textbooks is Jim Hefferon of Saint Michael's College in Vermont, who has authored books on the Theory of Computation [10] and Linear Algebra [9]. Hefferon has made extensive use of L^AT_EX graphics packages to draw figures and diagrams.

3 Diagrams

In this section I expose some of the more complex diagrams that can be drawn with special-purpose macros. I use the graphics packages DraTex and AlDraTex, developed at Ohio State University by Eitan Gurari [8] in the 1990s; they are available in T_EX Live and on CTAN (ctan.org/pkg/dratex). These packages include fairly primitive structures such as circles, lines, rectangles, and text boxes. They also include state diagrams, trees, grids, and other structures, with parameters controlling the size and appearance.¹

As with L^AT_EX, the DraTex and AlDraTex packages are extensible: new macro commands can be defined using existing macro commands. I have used this feature in developing macros for the examples shown in this section.

3.1 Charts, trees, and other diagrams with AlDraTex

The AlDraTex macros for chart and tree diagrams support diagrams with various attributes and formats, by specifying parameter values in the macro call. AlDraTex also allows the user to define other diagrams.

3.2 Chart diagrams

AlDraTex has macros which draw charts:

Pie charts may be round or oval, with shaded or painted sections, and labeled with internal or external labels.

XY charts permit graphs of two-dimensional data on an XY grid, with labeling on the axes and points, and with a continuous graph or discrete points

¹ Other graphics packages with similar capabilities include `tikz`, available on github, and `metapost`, available from the Tex User Group.

Bar charts allow variable size charts, with various properties on the bars, including 3-dimensional bars and tailored or painted bars, and labels on the bars and axes.

3.3 Tree diagrams

Trees, in various formats, may be drawn with AlDraTex. The user simply specifies the label on each node and the number of children. Nodes may be in various shapes, including circle, oval, rectangle, and text-only. The format of the tree may allow for horizontal-vertical edges or straight edges, and optional labels on the edges. Trees may be oriented vertically or horizontally.

3.4 Other diagrams

As an extensible language, with AlDraTex it is possible to define nodes of any shape for the diagrams described above, as well as macros for drawing state diagrams, such as those needed for finite automata.

3.5 Logic diagrams

The *Computer Organization* textbook [4] makes extensive use of diagrams. The primitive drawing commands of DraTex can be used to produce diagrams of logic gates — AND, OR, XOR, NOT, etc.

These gates can then be connected with each other to form higher level components such as encoders, decoders, multiplexers, adders, and arithmetic/logic units. An example of a logic diagram is shown in Figure 1.

One commonly used logic diagram is a canonical sum-of-products diagram, corresponding to a logic expression which is the logical OR of several logical ANDs. I developed a macro to draw sum-of-products diagrams with up to four variables. An example is shown in Figure 3. The user of this macro need only specify the names of the variables and the boolean true/false values for each product in the sum-of-products expression to be diagrammed.

3.6 Karnaugh maps

Another important diagram for the *Computer Organization* textbook is the Karnaugh Map (or K-Map). It is used to reduce a logic expression to its simplest sum-of-products form. The macro which draws K-Maps needs input for the positions of 1's (or don't-cares) in the map, as well as the grouping of the 1's. In theory a macro could deduce the groupings, but this feature is not currently available. The macro can draw K-Maps with three or four variables. An example of a K-Map with four variables is shown in Figure 5.

3.7 Object diagrams

The most interesting diagram macro that I have developed is for the object diagram construct commonly used in textbooks on object-oriented programming. I used this macro in the textbook *Introduction to Computer Science with Java* [6].

In an object-oriented language, such as Java or C++, the state of an object is determined by the instance variables² in the object's class. Each instance variable in an object may store either primitive data or a reference³ to another object. This is a recursive definition, and thus the object diagram, which is a visual representation of an object, is a recursive diagram. Consequently the macro which draws object diagrams is also recursive. An example of an object diagram is shown in Figure 7.

4 Examples of diagrams

Examples of diagrams which were described above are shown on the following pages. In each case the commands which generated the diagram are shown in the next figure. The full macro definitions and packages are available at cs.rowan.edu/~bergmann/books.

5 Summary

In this article I have presented a new paradigm for the production of free open-source textbooks. This paradigm has been used to produce four computer science textbooks, with a fifth book in development. The main contribution of this work is the capability of drawing figures and diagrams with \LaTeX macros. These macros are based on the extensible graphics packages `DraTex` and `AIDraTex`.

References

- [1] S.D. Bergmann. *Compiler Design: Theory, Tools, and Examples*. Wm. C. Brown Publishers, Dubuque, 1994.
- [2] S.D. Bergmann. Open source textbooks: A paradigm derived from open source software. *Publishing Research Quarterly* 30(1):1–10, March 2014.
- [3] S.D. Bergmann. *Compiler Design: Theory, Tools, and Examples*. Campbell Library at Rowan University, Glassboro, NJ, 2022. <https://rdw.rowan.edu/oer/1>
- [4] S.D. Bergmann. *Computer Organization with MIPS*. Campbell Library at Rowan University, Glassboro, NJ, 2022.

² Instance variables are also known as non-static fields in Java, or member data in C++.

³ A Java reference is similar to a C++ pointer, though there are restrictions on what can be done with a reference.

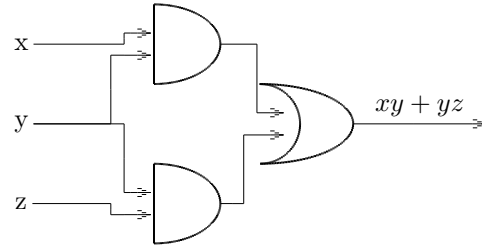


Figure 1: An implementation of the boolean function $xy + yz$ using logic gates

```
\DiagramSpec(\Inp & \Gate & \Wire)
\Diagram
  %% Inputs: Var, xLoc, yLoc
  (x,-50,30 & y,-50,0 & z,-50,-30)

  %% Gates: ID, type, inputs, xLoc, yLoc
  (And1,0,2,0,30,, &
   And2,0,2,0,-30,, & Or,1,2,50,0,,)

  %% Wires: srcID, targetID, input#
  (x, And1,1 & y, And1,2 &
   y, And2,1 & z,And2,2 &
   And1,Or,1 & And2,Or,2)
\MoveToNode(Orout,1,0.5) \Move(50,0)
\FcNode(result)
\Edge(Orout,result)
\EdgeLabel(--$xy + yz$--)
```

Figure 2: \LaTeX code used to draw Figure 1

- [5] S.D. Bergmann. *Computer Science Principles*. Campbell Library at Rowan University, Glassboro, NJ, 2022. Editions available for several programming languages.
- [6] S.D. Bergmann. *Introduction to Computer Science with Java Programming*. Campbell Library at Rowan University, Glassboro, NJ, 2022. <https://rdw.rowan.edu/oer/2>
- [7] A.B. Downey. *Python for Software Design: How to Think Like a Computer Scientist*. Cambridge University Press, New York, 2009.
- [8] E.M. Gurari. *TEX and LATEX: Drawing and Literate Programming*. McGraw-Hill, New York, 1994.
- [9] J. Hefferon. *Linear Algebra*. Orthogonal Publishing, Ann Arbor, MI, 2022. hefferon.net/linearalgebra
- [10] J. Hefferon. *Theory of Computation*. hefferon.net, 2022. hefferon.net/computation

◇ Seth D. Bergmann
Rowan University
Glassboro, NJ, USA
[bergmann \(at\) rowan dot edu](mailto:bergmann@rowan.edu)
<https://cs.rowan.edu/~bergmann/>

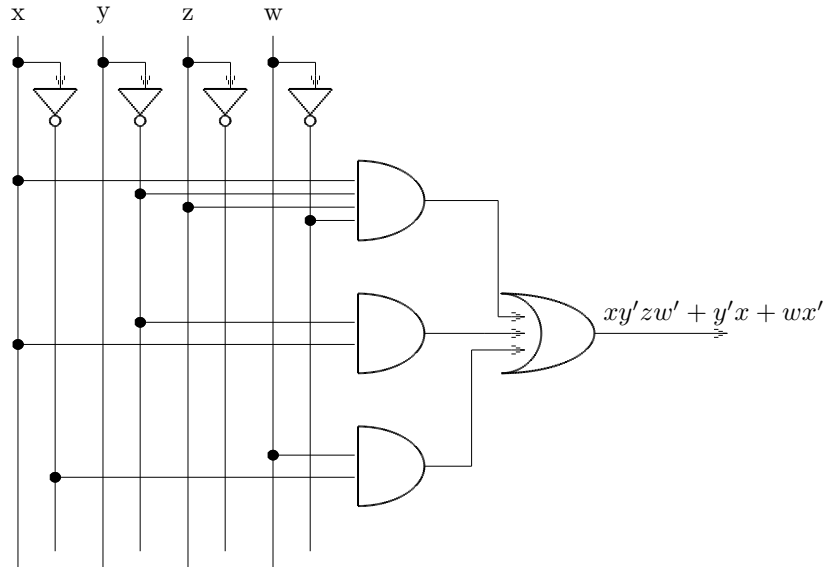


Figure 3: A sum of products logic diagram for the expression $xy'zw' + y'x + wx'$

```
\DiagramSpec(\SOP & \Ins & \Ands)
\Diagram
(4,3,xy'zw'+y'x+wx')    %% 4 variables, 3 terms
(x & y & z & w)         %% Variable names
%% Terms: #vars,
%%         var,
%%         0=negated
(4,x,1,y,0,z,1,w,0 &   %% xy'zw'
 2,y,0,x,1,0,0,0,0 &   %% y'x
 2,w,1,x,0,0,0,0,0)    %% wx'
)
```

Figure 4: L^AT_EX code used to draw Figure 3

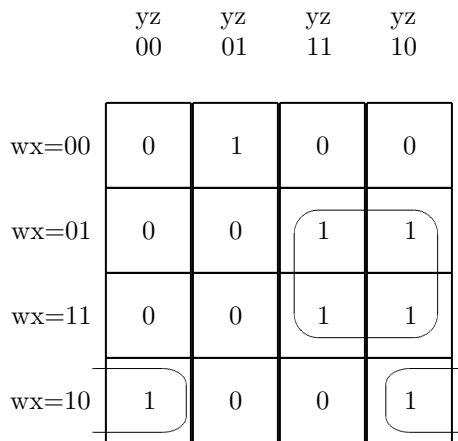


Figure 5: A K-map for the boolean expression $w'x'y'z + w'xyz + w'xyz' + wxyz + wxyz' + wx'y'z' + wx'yz'$. A 1x2 group and a 2x2 group are identified. The minimized expression is $xy + wx'z' + w'x'y'z$.

```
\KmapFourTop(0,1,0,0, 0,0,1,1, 32)
\KmapFourBot(0,0,1,1, 1,0,0,1, 32)

\Group(4,7,32,)           %% xy
\Group(h,10,32,h)        %% wx'z'
```

Figure 6: L^AT_EX code used to draw Figure 5

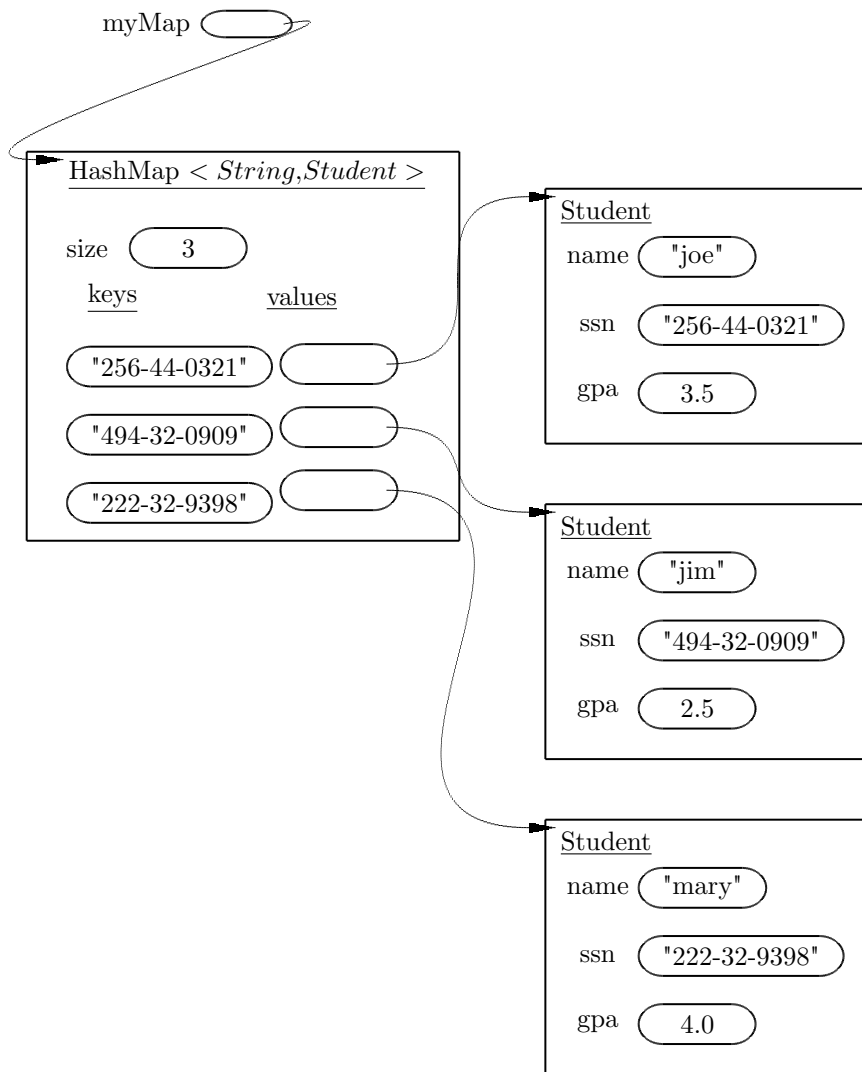


Figure 7: An object diagram showing the value of the variable `myMap` storing a reference to a map, after three entries have been added

```

\Indirect \Table <student1Prims>
{ 1, name, "joe" &
  1, ssn, "256-44-0321" &
  1, gpa, 3.5
}
\Indirect \Table <student10bjs>
{ 0, , , , , , }
\Indirect \Table <student2Prims>
{ 1, name, "jim" &
  1, ssn, "494-32-0909" &
  1, gpa, 2.5
}
\Indirect \Table <student20bjs>
{ 0, , , , , , }
\Indirect \Table <student3Prims>
{ 1, name, "mary" &
  1, ssn, "222-32-9398" &
  1, gpa, 4.0
}
\Indirect \Table <student30bjs>
{ 0, , , , , , }
\Indirect \Table <keyPrims>
{ 1, ,"256-44-0321" &
  1, ,"494-32-0909" &
  1, ,"222-32-9398" }
\Indirect \Table <keyObjs>
{ 0,0, , , , , }
\Indirect \Table <valuePrims>
{ 0, , }
\Indirect \Table <valueObjs>
{ 1, ,Student,student1Prims,
  student10bjs, , , &
  1, ,Student,student2Prims,
  student20bjs, , , &
  1, ,Student,student3Prims,
  student30bjs, , ,
}
\Obj (5, myMap,
  HashMap~$<String{,}Student>$,
  3, keyPrims, keyObjs,
  valuePrims, valueObjs)
  
```

Figure 8: L^AT_EX code used to draw Figure 7.

Automatically removing widows and orphans with `lua-widow-control`

Max Chernoff

Abstract

The `lua-widow-control` package, for plain $\text{Lua}\text{T}\text{E}\text{X}$ / $\text{Lua}\text{L}\text{A}\text{T}\text{E}\text{X}$ / $\text{Con}\text{T}\text{E}\text{X}\text{t}$ / $\text{O}\text{p}\text{T}\text{E}\text{X}$, removes widows and orphans without any user intervention. Using the power of $\text{Lua}\text{T}\text{E}\text{X}$, it does so *without* stretching any glue or shortening any pages or columns. Instead, `lua-widow-control` automatically lengthens a paragraph on a page or column where a widow or orphan would otherwise occur.

To use `lua-widow-control`, all that most users need do is place `\usepackage{lua-widow-control}` in their preamble. No further changes are required.

1 Motivation

TEX provides top-notch typesetting: even 40 years after its first release, no other program produces higher quality mathematical typesetting, and its paragraph-breaking algorithm is still state-of-the-art. However, its page breaking is not quite as sophisticated as its paragraph breaking and thus suffers from some minor issues.

Unmodified TEX has only two familiar ways of dealing with widows and orphans: it can either shorten a page by one line, or it can stretch vertical whitespace. TEX was designed for mathematical and scientific typesetting, where a typical page has multiple section headings, tables, figures, and equations. For this style of document, TEX 's default behaviour works quite well, since the slight stretching of whitespace between the various document elements is nearly imperceptible; however, for prose or other documents composed almost entirely of paragraphs, there is little vertical whitespace to stretch.

Since no ready-made and fully-automated solution to remove widows and orphans from all types of documents was available, I decided to create `lua-widow-control`.

2 What are widows and orphans?

2.1 Widows

A “widow” occurs when the majority of a paragraph is on one page or column, but the last line is on the following page or column. It not only looks quite odd for a lone line to be at the start of the page, but it makes a paragraph harder to read since the separation of a paragraph and its last line disconnects the two, causing the reader to lose context for the widowed line.

Max Chernoff

doi.org/10.47397/tb/43-1/tb133chernoff-widows

Widow	Orphan
A widow occurs when the last line of a paragraph is placed on a page separate from where it begins.	An orphan is when the first line of a paragraph occurs on the page before all of the other lines.

Figure 1: The difference between widows and orphans. If we imagine that each box is a different page, then this roughly simulates how widows and orphans appear.

2.2 Orphans

An “orphan” occurs when the first line of a paragraph is at the end of the page or column preceding the remainder of the paragraph. They are not as distracting for the reader, but they are still not ideal. Visually, widows and orphans are about equally disruptive; however, orphans tend not to decrease the legibility of a text as much as widows, so many authors choose to ignore them.

See figure 1 for a visual reference.

2.3 Broken hyphens

“Broken” hyphens occur whenever a page break occurs in a hyphenated word. These are not related to widows and orphans; however, breaking a word across two pages is at least as disruptive for the reader as widows and orphans. TEX identifies broken hyphens in the same ways as widows and orphans, so `lua-widow-control` treats broken hyphens in the same way.

3 History and etymology

The concept of widows and orphans is nearly as old as printing itself. In [13], a printers manual from 1683, we have:

Nor do good *Compositors* account it good Workmanship to begin a *Page* with a *Break-line*, unless it be a very short *Break*, and cannot be gotten in the foregoing *Page*; but if it be a long *Break*, he will let it be the *Direction-line* of the fore-going *Page*, and *Set* his *Direction* at the end of it. (p. 226)

3.1 Widows

However, the terms “widow” and “orphan” are much newer. The earliest published source that I could find referencing “widows” in typography is *Webster’s New International Dictionary* from 1934. However, no one — not even the editors of the dictionary [3] —

seems to know how it got there. Even then, the definition is somewhat different than it is now:

widow, n. c. *Print*. A short line or single word carried over from the foot of one column or page to the head of a succeeding column or page. [3]

Contrast this with the modern definition:

Typography. A short line of text (usually one consisting of one word or part of a word) which falls undesirably at the end of a paragraph, esp. one set at the top of a page or column. [16]

which includes a single lone line of any length.

3.2 Orphans

The term “orphan” is even more confusing. Its initial usage seems to have occurred some time after “widow” [3], and it is given many contradictory definitions. Most sources define an orphan as a first line at the bottom of the page and a widow as the last line at the top [2, 3, 4, 6, 9, 12, 14, 16]; however, some sources define these two terms as *exact opposites* of each other, with a widow as a first line at the bottom of the page and an orphan as the last line! [1, 3, 5, 14, 18] This usage is plain wrong; nevertheless, it is sufficiently common that you need to be careful when you see the terms “widow” and “orphan”.

3.3 Clubs

The T_EXbook never refers to “orphans” as such; rather, it refers to them as “clubs”. This term is remarkably rare: I could only find a *single* source published before *The T_EXbook* — a compilation article about the definition of “widow” — that mentions a “club line”:

The Dictionary staff informs me that they have no example of the use of the word widow in the typographical sense. [...]

Mr. Watson of the technical staff says that the Edinburgh printing houses referred to it as a “clubline”. [3, p. 4]

To my knowledge, a ‘widow’, or ‘widow-line’, is a short line, forming the end of a paragraph, which is carried over from the foot of a page or column to the top of the succeeding one. [...]

To my personal knowledge, in typographical parlance in Edinburgh, Scotland, the ‘widow’ is called a ‘club-line.’ [3, p. 23]

Both quotes above are from separate authors, and they each define a “club” like we define “widow”, not an “orphan”. In addition, they both mention that

the term is only used in Scotland. Even the extensive OED — which lists 17 full definitions and 103 subdefinitions for the noun “club” — doesn’t recognize the phrase. [15]

I spent a few hours searching through Google Books and my university library catalogue, but I could not find a single additional source. If anyone has any more information on the definition of a “club line” or why Knuth chose to use this archaic Scottish term in T_EX, please let me know!

4 Pagination in T_EX

Let’s move on to looking at how T_EX treats these widows and orphans.

4.1 Algorithm

It is tricky to understand how lua-widow-control works if you aren’t familiar with how T_EX breaks pages and columns. For a full description, you should consult Chapter 15 of *The T_EXbook* [9] (“How T_EX Makes Lines into Pages”); however, this goes into much more detail than most users require, so here is a *very* simplified summary of T_EX’s page breaking algorithm:

T_EX fills the page with lines and other objects until the next object will no longer fit. Once no more objects will fit, T_EX will align the bottom of the last line with the bottom of the page by stretching any available vertical spaces if (in L^AT_EX) `\flushbottom` is set; otherwise, it will break the page and leave the bottom empty.

However, some objects have penalties attached. Penalties encourage or discourage page breaks from occurring at specific places. For example, L^AT_EX sets a negative penalty before section headings to encourage a page break there; conversely, it sets a positive penalty after section headings to discourage breaking.

To reduce widows and orphans, T_EX sets weakly-positive penalties between the first and second lines of a paragraph to prevent orphans, and between the penultimate and final lines to prevent widows.

One important note: once T_EX begins breaking a page, it never goes back to modify any content on the page. Page breaking is a localized algorithm, without any backtracking.

4.2 Behaviour

Merely describing the algorithm doesn’t allow us to intuitively understand how T_EX deals with widows and orphans.

Due to the penalties attached to widows and orphans, T_EX tries to avoid them. Widows and orphans with small penalties attached — like L^AT_EX’s

default values of 150 — are only lightly coupled to the rest of the paragraph, while widows and orphans with large penalties — values of 10 000 or more — are treated as infinitely bad and are thus unbreakable. Intermediate values behave just as you would expect, discouraging page breaks proportional to their value.

When \TeX goes to break a page, it tries to avoid breaking at a location with a high penalty. How it does so depends on a few settings:

4.2.1 `\flushbottom` and `\normalbottom`

With the settings `\normalbottom` (Plain \TeX) or `\flushbottom` (\LaTeX), \TeX is willing to stretch any glue on the page by an amount roughly commensurate to the magnitude of the penalty: for small `\clubpenalty` and `\widowpenalty` values, \TeX will only slightly stretch the glue on the page before creating a widow or orphan; for very large penalties, \TeX will stretch the glue by a near-infinite amount.

This corresponds to the “Stretch” column in Figure 2. It is the default behaviour of Plain \TeX , and of the standard \LaTeX classes when the `twocolumn` option is given.

4.2.2 `\raggedbottom`

When `\raggedbottom` is set, \TeX won’t stretch any glue. Instead, for sufficiently-high `\clubpenalty` and `\widowpenalty` values, \TeX will shorten the page or column by one line in order to prevent the widow or orphan from being created.

This corresponds to the “Shorten” column in Figure 2 and is the default behaviour of the \LaTeX classes when the `twocolumn` option is not given.

5 `\looseness`

Before we can continue further, we need to discuss one more \TeX command: `\looseness`. The following is excerpted from Chapter 14 of [9] (“How \TeX Breaks Paragraphs into Lines”):

If you set `\looseness=1`, \TeX will try to make the current paragraph one line longer than its optimum length, provided that there is a way to choose such breakpoints without exceeding the tolerance you have specified for the badnesses of individual lines. Similarly, if you set `\looseness=2`, \TeX will try to make the paragraph two lines longer; and `\looseness=-1` causes an attempt to make it shorter. [...]

For example, you can set `\looseness=1` if you want to avoid a lonely “club line” or “widow line” on some page that does not have sufficiently flexible glue, or if you want the total number of lines in some two-column document to come out to be an even number.

It’s usually best to choose a paragraph that is already pretty “full”, i.e., one whose last line doesn’t have much white space, since such paragraphs can generally be loosened without much harm. You might also want to insert a tie between the last two words of that paragraph, so that the loosened version will not end with only one “widow word” on the orphans line; this tie will cover your tracks, so that people will find it hard to detect the fact that you have tampered with the spacing. On the other hand, \TeX can take almost any sufficiently long paragraph and stretch it a bit, without substantial harm.

The widow and orphan removal strategy suggested in the second paragraph works quite well; however, it requires manual editing each and every time a page or paragraph is rewritten or repositioned.

6 Alternate removal strategies

There have been a few previous attempts to improve upon \TeX ’s previously-discussed widow and orphan-handling abilities; however, none of these have been able to automatically remove widows and orphans without stretching any glue or shortening any pages.

The articles “Strategies against widows” by Paul Isambert [6] and “Managing forlorn paragraph lines” by Frank Mittelbach [11] both begin with comprehensive discussions of the methods of preventing widows and orphans. They agree that widows and orphans are bad and ought to be avoided; however, they differ in their solutions. *Strategies* proposes an output routine that reduces the length of facing pages by one line when necessary to remove widows and orphans, while *Managing* proposes that the author manually rewrites or adjusts `\looseness` when needed.

The post “Paragraph callback . . .” by jeremie [7] contains a file `widow-assist.lua` that automatically detects which paragraphs can be safely shortened or lengthened by one line. Mittelbach’s `widows-and-orphans` package [12] alerts the author to the pages that contain widows or orphans. Combined, these packages make it simple for the author to quickly remove widows and orphans by adjusting the values of `\looseness`; however, it still requires the author to make manual source changes after each revision.

Another article by Mittelbach [10] suggests an fully-automated solution to remove widows and orphans. This would seem to offer a complete solution; however, it requires multiple passes, an external tool, and has not yet been publicly released.

Ignore

<p><code>lua-widow-control</code> can remove most widows and orphans from a document, <i>without</i> stretching any glue or shortening any pages.</p> <p>It does so by automatically lengthening a paragraph on a page where a widow or orphan would otherwise occur. While <code>T_EX</code> breaks paragraphs into their natural length, <code>lua-widow-control</code> is breaking the paragraph 1 line longer than its natural length. <code>T_EX</code>'s paragraph is output to the page, but <code>lua-widow-control</code>'s paragraph is just stored for later. When a widow or orphan occurs, <code>lua-widow-control</code> can take over. It selects the previously-saved paragraph with the least badness; then, it replaces <code>T_EX</code>'s paragraph with its saved paragraph. This increases the text block height of the page by 1 line.</p> <p>Now, the last line of the current page can be pushed to the top of the next page. This removes the widow or the orphan without creating any additional work.</p>

```
\parskip=0pt
\clubpenalty=0
\widowpenalty=0
```

Stretch

<p><code>lua-widow-control</code> can remove most widows and orphans from a document, <i>without</i> stretching any glue or shortening any pages.</p> <p>It does so by automatically lengthening a paragraph on a page where a widow or orphan would otherwise occur. While <code>T_EX</code> breaks paragraphs into their natural length, <code>lua-widow-control</code> is breaking the paragraph 1 line longer than its natural length. <code>T_EX</code>'s paragraph is output to the page, but <code>lua-widow-control</code>'s paragraph is just stored for later. When a widow or orphan occurs, <code>lua-widow-control</code> can take over. It selects the previously-saved paragraph with the least badness; then, it replaces <code>T_EX</code>'s paragraph with its saved paragraph. This increases the text block height of the page by 1 line.</p> <p>Now, the last line of the current page can be pushed to the top of the next page.</p> <p>This removes the widow or the orphan without creating any additional work.</p>
--

```
\parskip=0pt plus 1fill
\clubpenalty=10000
\widowpenalty=10000
```

Shorten

<p><code>lua-widow-control</code> can remove most widows and orphans from a document, <i>without</i> stretching any glue or shortening any pages.</p> <p>It does so by automatically lengthening a paragraph on a page where a widow or orphan would otherwise occur. While <code>T_EX</code> breaks paragraphs into their natural length, <code>lua-widow-control</code> is breaking the paragraph 1 line longer than its natural length. <code>T_EX</code>'s paragraph is output to the page, but <code>lua-widow-control</code>'s paragraph is just stored for later. When a widow or orphan occurs, <code>lua-widow-control</code> can take over. It selects the previously-saved paragraph with the least badness; then, it replaces <code>T_EX</code>'s paragraph with its saved paragraph. This increases the text block height of the page by 1 line.</p> <p>Now, the last line of the current page can be pushed to the top of the next page.</p> <p>This removes the widow or the orphan without creating any additional work.</p>
--

```
\parskip=0pt
\clubpenalty=10000
\widowpenalty=10000
```

lua-widow-control

<p><code>lua-widow-control</code> can remove most widows and orphans from a document, <i>without</i> stretching any glue or shortening any pages.</p> <p>It does so by automatically lengthening a paragraph on a page where a widow or orphan would otherwise occur. While <code>T_EX</code> breaks paragraphs into their natural length, <code>lua-widow-control</code> is breaking the paragraph 1 line longer than its natural length. <code>T_EX</code>'s paragraph is output to the page, but <code>lua-widow-control</code>'s paragraph is just stored for later. When a widow or orphan occurs, <code>lua-widow-control</code> can take over. It selects the previously-saved paragraph with the least badness; then, it replaces <code>T_EX</code>'s paragraph with its saved paragraph. This increases the text block height of the page by 1 line.</p> <p>Now, the last line of the current page can be pushed to the top of the next page.</p> <p>This removes the widow or the orphan without creating any additional work.</p>
--

```
\usepackage{lua-widow-control}
```

Figure 2: A visual comparison of various automated widow-handling techniques.

`lua-widow-control` is essentially a combination of `widow-assist.lua` [7] and `widows-and-orphans` [12] (although its implementation is independent of both): when the `\outputpenalty` value indicates that a widow or orphan has occurred, Lua is used to find a stretchable paragraph. What `lua-widow-control` mainly adds on top of these packages is automation: it eliminates the requirement for any manual adjustments or changes to your document’s source.

7 Visual comparison

Although \TeX ’s page breaking algorithm is reasonably straightforward, it can lead to complex behaviour when widows and orphans are involved. The usual choices, when rewriting is not possible, are to ignore them, stretch some glue, or shorten the page. Figure 2 has a visual comparison of these options, which we’ll discuss in the following:

7.1 “Ignore”

As you can see, the last line of the page is on a separate page from the rest of its paragraph, creating a widow. This is usually highly distracting for the reader, so it is best avoided for the reasons previously discussed.

7.2 “Shorten”

This page did not leave any widows, but it did shorten the previous page by one line. Sometimes this is acceptable, but usually it looks bad because pages will then have different text-block heights. This can make the pages look quite uneven, especially when typesetting with columns or in a book with facing pages.

7.3 “Stretch”

This page also has no widows and it has a flush bottom margin. However, the space between each pair of paragraphs had to be stretched.

If this page had many equations, headings, and other elements with natural space between them, the stretched out space would be much less noticeable. \TeX was designed for mathematical typesetting, so it makes sense that this is its default behaviour. However, in a page with mostly text, these paragraph gaps look unsightly.

Also, this method is incompatible with grid typesetting, where all glue stretching must be quantised to the height of a line.

7.4 “lua-widow-control”

`lua-widow-control` has none of these issues: it eliminates the widows in a document while keeping a flush bottom margin and constant paragraph spacing.

To do so, `lua-widow-control` lengthened the second paragraph by one line. If you look closely, you can see that this stretched the interword spaces. This stretching is noticeable when typesetting in a narrow text block, but is mostly imperceptible with larger widths.

`lua-widow-control` automatically finds the “best” paragraph to stretch, so the increase in interword spaces should almost always be minimal.

8 Installation and standard usage

The `lua-widow-control` package was first released in October 2021. It is available in the default installations of both $\text{MiK}\TeX$ and $\text{T}\TeX$ Live, although you will need recent versions of either.

You may also download `lua-widow-control` manually from either CTAN,¹ the Con $\text{T}\TeX$ t Garden,² or GitHub,³ although it is best if you can install it through your $\text{T}\TeX$ distribution.

As its name may suggest, `lua-widow-control` requires $\text{Lua}\TeX$ ⁴ regardless of the format used. With that in mind, using `lua-widow-control` is quite simple:

Plain $\text{T}\TeX$	<code>\input lua-widow-control</code>
Op $\text{T}\TeX$	<code>\load[lua-widow-control]</code>
$\text{L}\text{A}\text{T}\text{E}\text{X}$	<code>\usepackage{lua-widow-control}</code>
Con $\text{T}\text{E}\text{X}\text{t}$	<code>\usemodule[lua-widow-control]</code>

And that’s usually enough. Most users won’t need to do anything else since `lua-widow-control` comes preconfigured and ready-to-go.

9 Options

Nevertheless, `lua-widow-control` does have a few options.

`lua-widow-control` tries very hard to have a “natural” user interface with each format, so how you set an option heavily depends on how you are running `lua-widow-control`. Also note that not every option is available in every format.

Some general guidelines:

Plain $\text{T}\TeX$ /Op $\text{T}\TeX$ Some options are set by modifying a register, while others must be set manually using `\directlua`.

$\text{L}\text{A}\text{T}\text{E}\text{X}$ Options can be set either as package options or at any point in the document with `\lwcsetup`.

Con $\text{T}\text{E}\text{X}\text{t}$ Always use `\setuplwc`.

¹ ctan.org/pkg/lua-widow-control

² [modules.contextgarden.net/cgi-bin/module.cgi/action=view/id=127](https://modules.contextgarden.net/cgi-bin/module.cgi?action=view/id=127)

³ github.com/gucci-on-fleek/lua-widow-control/releases/latest/

⁴ Or $\text{LuaMeta}\text{T}\text{E}\text{X}$ in the case of Con $\text{T}\text{E}\text{X}\text{t}$.

9.1 Disabling

You may want to disable `lua-widow-control` for certain portions of your document. You can do so with the following commands:

```
Plain TeX/OpTeX      \lwcdisable
LATeX              \lwcsetup{disable}
ConTeXt              \setuplwc[state=stop]
```

This prevents `lua-widow-control` from stretching any paragraphs that follow. If a page has earlier paragraphs where `lua-widow-control` was still enabled and a widow or orphan is detected, `lua-widow-control` will still attempt to remove the widow or orphan.

9.2 Enabling

`lua-widow-control` is enabled as soon as the package is loaded. If you have previously disabled it, you will need to re-enable it to save new paragraphs.

```
Plain TeX/OpTeX      \lwcenable
LATeX              \lwcsetup{enable}
ConTeXt              \setuplwc[state=start]
```

9.3 Automatically disabling

You may want to disable `lua-widow-control` for certain commands where stretching is undesirable such as section headings. Of course, manually disabling and then enabling `lua-widow-control` multiple times throughout a document would quickly become tedious, so `lua-widow-control` provides some options to do this automatically for you.

`lua-widow-control` automatically patches the default `LATeX`, `ConTeXt`, `Plain TeX`, `OpTeX`, `memoir`, `KOMA-Script`, and `titlesec` section commands, so you don't need to patch these. Any others, though, you'll need to patch yourself.

```
Plain TeX/OpTeX      \lwcdisablecmd{\macro}
LATeX              \lwcsetup{disablecmds={
                    <csnameone>, <csnametwo>}}
ConTeXt              \prependtoks\lwc@patch@pre
                    \to\everybefore{hook}
                    \prependtoks\lwc@patch@post
                    \to\everyafter{hook}
```

9.4 \emergencystretch

`lua-widow-control` defaults to an `\emergencystretch` value of 3 em for stretched paragraphs, but you can configure this.

`lua-widow-control` will only use the `\emergencystretch` when it cannot lengthen a paragraph in any other way, so it is fairly safe to set this to a large value. `TeX` accumulates badness when `\emergencystretch` is used [8], so it's pretty rare that a paragraph that

requires any `\emergencystretch` will actually be used on the page.

```
Plain TeX/OpTeX      \lwcemergencystretch=
                    <dimension>
LATeX              \lwcsetup{emergencystretch=
                    <dimension>}
ConTeXt              \setuplwc[emergencystretch=
                    <dimension>]
```

9.5 Penalties

You can also manually adjust the penalties that `TeX` assigns to widows and orphans. Usually, the defaults are fine, but there are a few circumstances where you may want to change them.

```
Plain TeX/OpTeX      \widowpenalty=<integer>
                    \clubpenalty=<integer>
                    \brokenpenalty=<integer>
LATeX              \lwcsetup{widowpenalty=<integer>}
                    \lwcsetup{orphanpenalty=<integer>}
                    \lwcsetup{brokenpenalty=<integer>}
ConTeXt              \setuplwc[widowpenalty=<integer>]
                    \setuplwc[orphanpenalty=<integer>]
                    \setuplwc[brokenpenalty=<integer>]
```

The value of these penalties determines how much `TeX` should attempt to stretch glue before passing the widow or orphan to `lua-widow-control`. If you set the values to 1 (default), `TeX` will stretch nothing and immediately trigger `lua-widow-control`; if you set the values to 10 000, `TeX` will stretch infinitely and `lua-widow-control` will never be triggered. If you set the value to some intermediate number, `TeX` will first attempt to stretch some glue to remove the widow or orphan; only if it fails will `lua-widow-control` come in and lengthen a paragraph. As a special case, if you set the values to 0, both `TeX` and `lua-widow-control` will completely ignore the widow or orphan.

9.6 \nobreak behaviour

When `lua-widow-control` encounters an orphan, it removes it by moving the orphaned line to the next page. The majority of the time, this is an appropriate solution. However, if the orphan is immediately preceded by a section heading (or `\nobreak/penalty 10000`), `lua-widow-control` would naïvely separate a section heading from the paragraph that follows. This is almost always undesirable, so `lua-widow-control` provides some options to configure this.

```
Plain TeX/OpTeX      \directlua{lwc.
                    nobreak_behaviour="<value>"}
LATeX              \lwcsetup{nobreak=<value>}
ConTeXt              \setuplwc[nobreak=<value>]
```

keep	split	warn
	Heading	Heading
Heading The first line text text text	The first line text text text last line.	The first line text text text last line.

Figure 3: A visual comparison of the `nobreak` option values.

The default value, `keep`, *keeps* the section heading with the orphan by moving both to the next page. The advantage to this option is that it removes the orphan and retains any `\nobreaks`; the disadvantage is that moving the section heading can create a large blank space at the end of the page.

The value `split` *splits* up the section heading and the orphan by moving the orphan to the next page while leaving the heading behind. This is usually a bad idea, but exists for the sake of flexibility.

The value `warn` causes `lua-widow-control` to give up on the page and do nothing, leaving an orphaned line. `lua-widow-control` *warns* the user so that they can manually remove the orphan.

See figure 3 for a visual reference.

9.7 Maximum cost

`lua-widow-control` ranks each paragraph on the page by how much it would “cost” to lengthen that paragraph. By default, `lua-widow-control` selects the paragraph on the page with the lowest cost; however, you can configure it to only select paragraphs below a selected cost.

If there aren’t any paragraphs below the set threshold, then `lua-widow-control` won’t remove the widow or orphan and will instead issue a warning.

```
Plain TEX/OpTEX      \lwcmaxcost=<integer>
LATEX              \lwcsetup{max-cost=<integer>}
ConTEXt            \setuplwc [maxcost=<integer>]
```

Based on my testing, `max-cost` values less than 1 000 cause completely imperceptible changes in interword spacing; values less than 5 000 are only noticeable if you are specifically trying to pick out the expanded paragraph on the page; values less than 15 000 are typically acceptable; and larger values may become distracting. `lua-widow-control` defaults to an infinite `max-cost`, although the “strict” and “balanced” modes sets the values to 5 000 and 10 000 respectively.

10 Presets

As you can see, `lua-widow-control` provides quite a few options. Luckily, there are a few presets that you

can use to set multiple options at once. These presets are a good starting point for most documents, and you can always manually override individual options.

Currently, these presets are L^AT_EX-only.

```
LATEX \lwcsetup{<preset>}
```

10.1 default

If you use `lua-widow-control` without any options, it defaults to this preset. In default mode, `lua-widow-control` takes all possible measures to remove widows and orphans and will not attempt to stretch any vertical glue. This usually removes >95% of all possible widows and orphans. The catch here is that this mode is quite aggressive, so it often leaves behind some fairly “spacey” paragraphs.

This mode is good if you want to remove (nearly) all widows and orphans from your document, without fine-tuning the results.

10.2 strict

`lua-widow-control` also offers a strict mode. This greatly restricts `lua-widow-control`’s tolerance and makes it so that it will only lengthen paragraphs where the change will be imperceptible.

The caveat with strict mode is that — depending on the document — `lua-widow-control` will be able to remove less than a third of the widows and orphans. For the widows and orphans that can’t be automatically removed, a warning will be printed to your terminal and log file so that a human can manually fix the situation.

This mode is good if you want the best possible typesetting and are willing to do some manual editing.

10.3 balanced

Balanced mode sits somewhere between default mode and strict mode. This mode first lets T_EX stretch a little glue to remove the widow or orphan; only if that fails will it then trigger `lua-widow-control`. Even then, the maximum paragraph cost is capped. Here, `lua-widow-control` can usually remove 90% of a document’s potential widows and orphans, and it does so while making a minimal visual impact.

This mode is recommended for most users who care about their document’s typography. This mode is not the default since it doesn’t remove all widows and orphans: it still requires a little manual intervention.

11 Compatibility

The `lua-widow-control` implementation is almost entirely in Lua, with only a minimal T_EX footprint. It doesn’t modify the output routine, inserts/floats,

Table 1: lua-widow-control options set by each mode.

Option	default	balanced	strict
max-cost	∞	10000	5000
emergencystretch	3em	1em	0pt
nobreak	keep	keep	warn
widowpenalty	1	500	1
orphanpenalty	1	500	1
brokenpenalty	1	500	1

`\everypar`, and it doesn't insert any whatsits. This means that it should be compatible with nearly any \TeX package, class, and format. Most changes that lua-widow-control makes are not observable on the \TeX side.

However, on the Lua side, lua-widow-control modifies much of a page's internal structure. This should not affect any \TeX code; however, it may surprise Lua code that modifies or depends on the page's low-level structure. This does not matter for Plain \TeX or \LaTeX , where even most Lua-based packages don't depend on the node list structure; nevertheless, there are a few issues with Con \TeX t.

Simple Con \TeX t documents tend to be fine, but many advanced Con \TeX t features rely heavily on Lua and can thus be disturbed by lua-widow-control. This is not a huge issue — the lua-widow-control manual is written in Con \TeX t — but lua-widow-control is inevitably more reliable with Plain \TeX and \LaTeX than with Con \TeX t.

Finally, keep in mind that adding lua-widow-control to a document will almost certainly change its page break locations.

11.1 Formats

lua-widow-control runs on all known Lua \TeX -based formats: Plain Lua \TeX , Lua \LaTeX , Con \TeX t MkIV, Con \TeX t MkXL/LMTX, and Op \TeX . Unless otherwise documented, all features should work equally well in all formats.

11.2 Columns

Since \TeX and the formats implement column breaking and page breaking through the same internal mechanisms, lua-widow-control removes widows and orphans between columns just as it does with widows and orphans between pages.

lua-widow-control is known to work with the \LaTeX class option `twocolumn` and the two-column output routine from Chapter 23 of [9].

11.3 Performance

lua-widow-control runs entirely in a single pass, without depending on any `.aux` files or the like. Thus, it shouldn't meaningfully increase compile times. Although lua-widow-control internally breaks each paragraph twice, modern computers break paragraphs near-instantaneously, so you are not likely to notice any slowdown.

11.4 ϵ - \TeX penalties

Knuth's original \TeX has three basic line penalties: `\interlinepenalty`, which is inserted between all lines; `\clubpenalty`, which is inserted after the first line; and `\widowpenalty`, which is inserted before the last line. The ϵ - \TeX extensions [20] generalize these commands with a syntax similar to `\parshape`: with `\widowpenalties` you can set the penalty between the last, second last, and n th last lines of a paragraph; `\interlinepenalties` and `\clubpenalties` behave similarly.

lua-widow-control makes no explicit attempts to support these new `-penalties` commands. Specifically, if you give a line a penalty that matches either `\widowpenalty` or `\clubpenalty`, lua-widow-control will treat the lines exactly as it would a widow or orphan. So while these commands won't break lua-widow-control, they are likely to lead to some unexpected behaviour.

12 Short last lines

When lengthening a paragraph with `\looseness`, it is common advice to insert ties (`~`) between the last few words of the paragraph to avoid overly-short last lines [9]. lua-widow-control does this automatically, but instead of using ties or `\hboxes`, it uses the `\parfillskip` parameter [9, 21]. When lengthening a paragraph (and only when lengthening a paragraph — remember, lua-widow-control doesn't interfere with \TeX 's output unless it detects a widow or orphan), lua-widow-control sets `\parfillskip` to `0pt plus 0.8\hspace`. This normally makes the last line of a paragraph be at least 20% of the overall paragraph's width, thus preventing ultra-short lines.

13 How it works

lua-widow-control uses a fairly simple algorithm to eliminate widows and orphans, but there are a few subtleties.

13.1 Setup

lua-widow-control sets the parameters `\clubpenalty`, `\widowpenalty`, and `\brokenpenalty` to sentinel values of 1. This will signal to lua-widow-control when

a widow or orphan occurs, yet it is small enough that it won't stretch any glue.

`lua-widow-control` also enables Lua \TeX 's microtypographic extensions [19]. This isn't strictly necessary; however, it significantly increases the number of paragraphs that can be acceptably "loosened".

That is all that happens on the \TeX end. The rest of `lua-widow-control` is pure Lua.

13.2 Paragraph breaking

First, `lua-widow-control` hooks into the paragraph breaking process, before any output routines or page breaking.

Before a paragraph is broken by \TeX , `lua-widow-control` grabs the unbroken paragraph. Then `lua-widow-control` breaks the paragraph one line longer than its natural length and stores it for later. It does this in the background, *without* interfering with how \TeX breaks paragraphs into their natural length.

After \TeX has broken its paragraph into its natural length, `lua-widow-control` appears again. Before the broken paragraph is added to the main vertical list, `lua-widow-control` "tags" the first and last nodes of the paragraph using a Lua \TeX attribute. These attributes associate the previously-saved lengthened paragraph with the naturally-typeset paragraph on the page.

13.3 Page breaking

`lua-widow-control` intercepts `\box255` (the `\vbox` output by \TeX) immediately before the output routine runs, after all the paragraphs have been typeset.

First, `lua-widow-control` looks at the `\outputpenalty` of the page or column. If the page was broken at a widow or orphan, the `\outputpenalty` will be equal to either the `\widowpenalty` or the `\clubpenalty`. If the `\outputpenalty` does not indicate a widow or orphan, `lua-widow-control` will stop and return `\box255` unmodified to the output routine, and \TeX continues as normal.

Otherwise, we assume that we have a widow or orphan on the page, meaning that we should lengthen the page by 1 line. We iterate through the list of saved paragraphs to find the lengthened paragraph with the least cost. After we've selected a good paragraph, we traverse through the page to find the original version of this paragraph — the one that unmodified \TeX originally typeset. Having found the original paragraph, we splice in the lengthened paragraph in place of the original.

Since the page is now 1 line longer than it was before, we pull the last line off the page to bring it back to its original length, and place that line onto the top of \TeX 's "recent contributions" list. When

the next page begins, this line will be inserted before all other paragraphs, right at the top. Now, we can return the new, widow-free page (updated `\box255`) to the output routine, which proceeds as normal.

14 Choosing the "best" paragraph

As we discussed previously, `lua-widow-control` lengthens the paragraph with the lowest cost. However, assigning a cost to each paragraph is not quite as simple as it sounds. Before we look at how `lua-widow-control` assigns costs, let's look at how \TeX scores paragraphs when breaking them naturally.

14.1 How \TeX scores paragraphs

All glue in \TeX has a certain natural size: the size that it would be in an ideal scenario. However, most glue also has stretch and shrink components so that the glue can change in size to adapt to its surroundings. For each line, \TeX individually sums the total stretch/shrink for the line and the stretch/shrink that was actually used. We define the stretch/shrink ratio r as the quotient of the stretch/shrink used and the stretch/shrink available. Then the badness b of a line is approximately defined as

$$b = 100r^3.$$

This is the badness referenced in the commonly-seen `Underfull \hbox (badness 1234)` warnings that \TeX produces.

\TeX calculates the badness for each line individually; however, we also need to assess the paragraph as a whole. To do so, \TeX defines the demerits for a whole paragraph d as approximately⁵ the sum of the squared badnesses for each line. The natural paragraph that \TeX breaks is the one that minimizes d .

One important thing to realize is that demerits grow incredibly fast: demerits are proportional to the *sixth* power of glue stretch. This means that you can expect to see extremely large demerit values, even for a relatively "good" paragraph.

14.2 Possible cost functions

Now, let's return to how `lua-widow-control` assigns costs to each paragraph. This is surprisingly more complicated than it sounds, so we'll go through a few possible cost functions first.

Here, we use c for the cost of a paragraph, d for the total demerits, and l for the number of lines (`\prevgraf`).

⁵ We ignore any additional demerits or penalties that \TeX may add.

14.2.1 The original implementation

The original implementation of `lua-widow-control` used the very simple cost function

$$c = d.$$

This cost function works reasonably well, but has one major issue: it doesn't take into account the number of lines in the paragraph. The demerits for a paragraph is the sum of the demerits for each line. This means this cost function will prefer using shorter paragraphs since they tend to have fewer demerits. However, long paragraphs tend to have much more available glue stretch, so this strategy can lead to suboptimal solutions.

14.2.2 Scaling by the number of lines

Once I realized this issue, I tried correcting it by dividing by the number of lines in the paragraph to get the average demerits instead of the total demerits:

$$c = \frac{d}{l}$$

This works better than the previous function, but still has an issue. If we have a fairly bad ten-line paragraph with total demerits $10d$ and an almost-equally bad two-line paragraph with total demerits $2d + 1$, then by this cost function, the ten-line paragraph will have a lower cost and will be chosen. This means that our page now has ten bad lines instead of two bad lines, which is not ideal.

14.2.3 Current implementation

Our first cost function, $c = dl^0$, doesn't consider the number of lines at all, while our second cost function, $c = dl^{-1}$, considers the number of lines too much. Splitting the difference between the two functions, we get the current implementation:

$$c = \frac{d}{\sqrt{l}}$$

I didn't arrive at this function through any sort of scientific testing; rather, I picked the simplest function that I could think of that satisfies the following two properties:

- Given a long paragraph and a short paragraph with different average badnesses per line, prefer the one with the least average badness.
- Given two paragraphs with equal average badnesses per line, prefer the shorter one.

15 Quantitative analysis

Let's look at some statistics for `lua-widow-control`. For testing, I downloaded the top ten books on *Project Gutenberg*,⁶ converted them to \LaTeX using `pandoc`, concatenated them into a single article

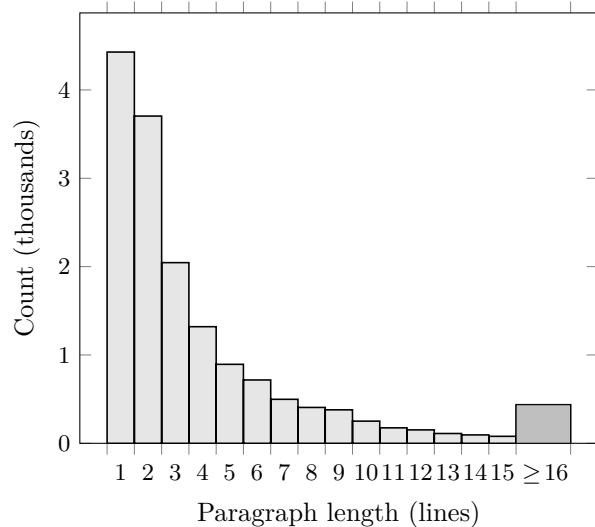


Figure 4: Histogram of natural paragraph lengths in the sample text.

file, and compiled twice. This gives us a PDF with 1381 pages, 15692 paragraphs, 61865 lines, and 399 widows and orphans (if they aren't removed).

This is a fairly challenging test: almost every third page has a widow or orphan, over half of the paragraphs have two lines or fewer, and the text block is set to the fairly wide article defaults. An average document is much less challenging for `lua-widow-control`, so we can consider this to be a worst-case scenario.

15.1 Widows and orphans removed

When we run \LaTeX with its default settings on the file, 179 (47%) of the widows and orphans are removed. When we add `lua-widow-control` with default settings, we remove 392 (98%). Switching to strict mode, we can only remove 52 (13%) of the widows and orphans. In balanced mode, we remove 348 (87%). See figure 5 for a visual comparison.

15.2 Paragraph costs

The last section showed us that `lua-widow-control` is quite effective at removing widows and orphans, so now let's look at the paragraphs that `lua-widow-control` expands. As \TeX processes a document, `lua-widow-control` is recording the costs for the naturally-broken and expanded versions of each paragraph in the document. Costs don't mean that much on their own, but a lower cost is always better.

⁶ *Frankenstein, Pride and Prejudice, Alice's Adventures in Wonderland, The Great Gatsby, The Adventures of Sherlock Holmes, Simple Sabotage Field Manual, A Tale of Two Cities, The Picture of Dorian Gray, Moby Dick, and A Doll's House.*

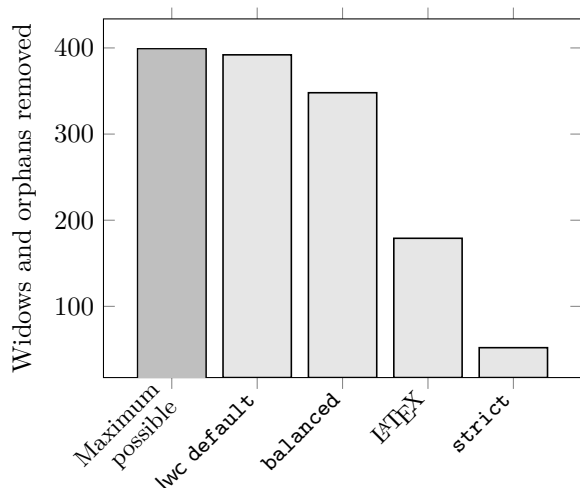


Figure 5: The number of widows and orphans removed by each method.

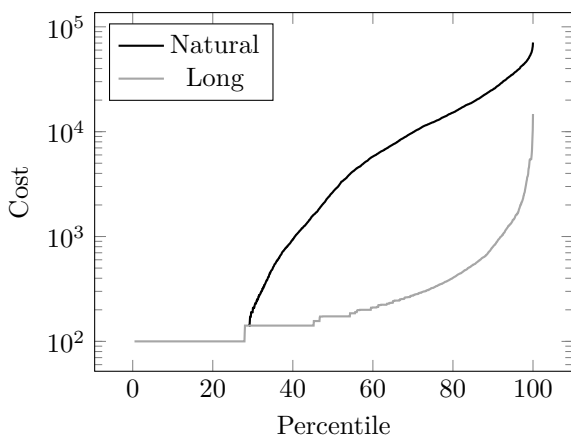


Figure 6: Paragraph costs by percentile rank for naturally-broken and one-line lengthened paragraphs.

As you can see in figure 6, the lengthened paragraphs tend to have *much* higher costs than the naturally-broken paragraphs. This is not surprising, since (as we’ve seen) a paragraph’s demerits scale with the sixth power of glue stretch, so even a small amount of glue stretch can cause a huge increase in demerits.

The empty space on the left of the “long” line is from the paragraphs that `lua-widow-control` was unable to lengthen at any cost. LuaTeX assigns these paragraphs zero demerits, so they disappear on a logarithmic plot.

15.3 Lengthening vs. shortening paragraphs

Figure 7 shows the number of paragraphs that `lua-widow-control` could potentially stretch or shrink. The one-line paragraphs are broken out separately since

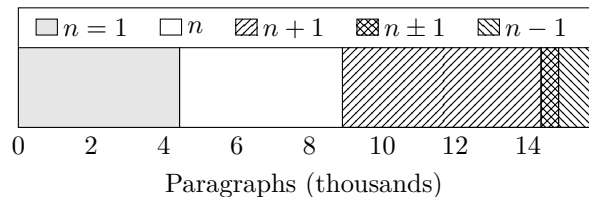


Figure 7: The number of paragraphs in the test sample that (respectively) have exactly one line, cannot be stretched or shrunk, can be only stretched by one line, can be either stretched or shrunk, and can be only shrunk.

this test sample has an anomalous number of them. Otherwise, we can see that `lua-widow-control` is capable of stretching the majority of paragraphs.

We can also see that of non-single-line paragraphs, only about 8% of paragraphs can only be shrunk (the last segment of figure 7), and this is in a document where 13% of paragraphs have at least eight lines. Most documents rarely have such long paragraphs, and it is these long paragraphs that are the easiest to shrink.

Because of this, `lua-widow-control` doesn’t even attempt to shrink paragraphs; it only stretches them.

16 Known issues

`lua-widow-control` is quite stable these days, a few issues remain:

- When a three-line paragraph is at the end of a page forming a widow, `lua-widow-control` will remove the widow; however, it will leave an orphan. This issue is inherent to any process that removes widows through paragraph expansion and is thus unavoidable. Orphans are considered to be better than widows [2], so this is still an improvement.
- Sometimes a widow or orphan cannot be eliminated because no paragraph has enough stretch. Sometimes this can be remediated by increasing `lua-widow-control`’s `\emergencystretch`; however, some pages just don’t have any suitable paragraph.

Long paragraphs with short words tend to be stretchier than short paragraphs with long words since these long paragraphs have more interword glue. Narrow columns also stretch more easily than wide columns since you need to expand a paragraph by less to make a new line.

- When running under LuaMetaTeX (ConTeXt), the log may contain many lines like “`luatex warning > tex: left parfill skip is gone`”.

These messages are completely harmless (although admittedly quite annoying).

- \TeX may warn about overfull `\vboxes` on pages where `lua-widow-control` removed a widow or orphan. This happens due to the way that `lua-widow-control` corrects for the `\prevdepth` when replacing paragraphs. It does not actually produce an overfull `vbox`, but there is a warning nevertheless. You can set `\vfuzz=2.5pt` to hide the warning.
- `lua-widow-control` only attempts to expand paragraphs on a page with a widow or orphan. A global system like in [10] would solve this; however, this is both NP-complete [17] and impossible to solve in a single pass. Very rarely would such a system remove widow or orphans that `lua-widow-control` cannot.

17 Conclusion

All this probably makes `lua-widow-control` look quite complicated, and this is true to some extent. However, this complexity is hidden from the end user: as stated at the outset, most users merely need to place `\usepackage{lua-widow-control}` in their \LaTeX document preamble, and `lua-widow-control` will remove all the troublesome widows and orphans, without needing any manual intervention.

Should you have any issues, questions, or suggestions for `lua-widow-control`, please visit the project's GitHub page: github.com/gucci-on-fleek/lua-widow-control. Any feedback is greatly appreciated!

References

- [1] G. Ambrose, P. Harris. *The Layout Book*. Advanced Level Series. Bloomsbury Academic, 2007.
- [2] R. Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, 3rd ed., 2004.
- [3] K. Brown. The typographical widow. *Bulletin of the New York Public Library* 52(1):3–25, Jan. 1948. hdl.handle.net/2027/uc1.b3310084
- [4] K. Brown. The typographical widow: Encore. *Bulletin of the New York Public Library* 52(9):458–466, Sept. 1948. hdl.handle.net/2027/uc1.b3310084
- [5] R. Hunt. *Advanced Typography: From Knowledge to Mastery*. Bloomsbury Publishing, 2020.
- [6] P. Isambert. Strategies against widows. *TUGboat* 31(1):12–17, 2010. tug.org/TUGboat/tb31-1/tb97isambert.pdf
- [7] jeremie. Paragraph callback to help with widows/orphans hand tuning, August 2017. tex.stackexchange.com/q/372062
- [8] D.E. Knuth. The new versions of \TeX and *METAFONT*. *TUGboat* 10(3):325–328, Nov. 1989. tug.org/TUGboat/tb10-3/tb25knut.pdf
- [9] D.E. Knuth. *The \TeX book*. Addison–Wesley, 2021.
- [10] F. Mittelbach. A general framework for globally optimized pagination. *Computational Intelligence* 35(2):242–284, Mar. 2018. doi.org/10.1111/coin.12165
- [11] F. Mittelbach. Managing forlorn paragraph lines (a.k.a. widows and orphans) in \LaTeX . *TUGboat* 39(3):246–251, 2018. tug.org/TUGboat/tb39-3/tb123mitt-widows.pdf
- [12] F. Mittelbach. The widows-and-orphans package, March 2021. ctan.org/pkg/widows-and-orphans
- [13] J. Moxon. *Mechanick exercises*, vol. 2, 1683. archive.org/details/mechanickexercis00moxo_0
- [14] Oxford English Dictionary. line at end of paragraph. www.oed.com/view/th/class/195380
- [15] Oxford English Dictionary. club, n., Sept. 2021. www.oed.com/view/Entry/34788
- [16] Oxford English Dictionary. widow, n., Dec. 2021. www.oed.com/view/Entry/228912
- [17] M.F. Plass. *Optimal pagination techniques for automatic typesetting systems*. Ph.D. thesis, Stanford University, 1981. tug.org/docs/plass/plass-thesis.pdf
- [18] I. Saltz. *Typography Essentials Revised and Updated*. Rockport Publishers, 2019.
- [19] Hàn Thế Thành. Micro-typographic extensions to the \TeX typesetting system. *TUGboat* 21(4):317–317, Dec. 2000. tug.org/TUGboat/tb21-4/tb69thanh.pdf
- [20] The $\mathcal{N}\mathcal{T}\mathcal{S}$ Team. The $\epsilon\text{-}\TeX$ manual, Feb. 1998. ctan.org/pkg/etex
- [21] U. Wermuth. Experiments with `\parfillskip`. *TUGboat* 39(3):276–303, 2018. tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf

◇ Max Chernoff
Calgary, Alberta
Canada
mseven at telus dot net

l3build: The beginner's guide

Joseph Wright

1 Introduction

For package authors, creating a release is a regular process, ideal for automation. There are several steps to creating a release to CTAN, for example ensuring documentation is updated, structuring an archive correctly and actually uploading the material.

Some time ago, the L^AT_EX Team extended their existing basic scripts to create an independent tool, l3build, which can cover all of those tasks. Most significantly, it included features to run comprehensive tests: this aspect was previously covered for *TUGboat* (2014, **35**:3, pp. 287–293). Here, I will give a more general overview of the tool, looking at how it can help package authors create releases in a quick and reliable manner.

2 l3build at the command line

With a modern T_EX system, l3build is available as a command at the command line/terminal. It understands $\langle targets \rangle$, $\langle options \rangle$ and $\langle arguments \rangle$.

```
l3build  $\langle target \rangle$  [ $\langle options \rangle$ ] [ $\langle arguments \rangle$ ]
```

The $\langle target \rangle$ is the task we want l3build to carry out. The most common ones are:

- check Runs one or more automated tests
- save Saves the result of one or more tests
- doc Typesets documentation
- ctan Creates a zip file ready to send to CTAN
- upload Sends a zip file to CTAN
- install Installs the package in the local texmf tree (there is also `uninstall` to reverse this)

The particular $\langle options \rangle$ which apply depend on the $\langle target \rangle$. For example, when running the `check` target, l3build will normally finish all of the tests then report the results. However, particularly when used with an automated system, one might want the tests to halt as soon as there is an error. That is available using the `--halt-on-error` option, which is also available as the one-letter version `-H`.

Some targets *require* one or more $\langle arguments \rangle$. For example, to save test results, you *have* to give the name of the test(s). Some targets take an optional $\langle argument \rangle$: `doc` is a good example, as you can limit this to a specific PDF (where your project has several PDFs, this can be useful). Finally, some targets do not need arguments at all: `install` is an example.

3 Configuration: the build.lua file

The configuration of l3build for a project is controlled by a file called `build.lua`, which should be present

in the main directory. This is a Lua file, and so *can* contain sophisticated programming. However, for a large number of use cases, the requirements are simply to set either string variables or tables of strings. That means that for many projects, the `build.lua` file will comprise just a few short lines, and requires no insight into Lua programming.

Only one line is absolutely required: one to tell l3build the name of the package. This is specified as the `module` string:

```
module = "mypkg"
```

By the way, Lua will allow us to mark strings using either single or double quotes. I favour double ones, and only use single quotes if the string itself contains a double quote, but it's purely personal preference.

The standard settings in l3build are based around using one or more `.dtx` files extracted using an `.ins` file. They also assume that the documentation is in the `.dtx` files. One common structure with larger packages is to separate out the documentation from the code, so to have a `.tex` file to typeset. This can be covered using

```
typesetfiles = {"*.tex"}
```

or if we want to specify only specific files, for example:

```
typesetfiles =
{
  "mypkg-doc.tex",
  "mypkg-example-a.tex"
}
```

Here, we are using a Lua table: these can hold a variety of data, but all we need to know here is that we can use a comma-separated list of names inside braces.

If the project we are working on doesn't use the `.dtx` format, we need to tell l3build the name(s) of our source files, and that it can skip unpacking:

```
sourcefiles = {"*.def", "*.sty"}
unpackfiles = {}
```

Or we might unpack some files that are not on the standard list, in which case we need to tell l3build to install them:

```
installfiles = {"*.def", "*.sty"}
```

The standard settings for l3build assume that all of the source files are in the same directory as the `build.lua` file. Some authors prefer a more complex structure. For example, for L^AT_EX itself there are *lots* of documentation files, so they are inside a subdirectory:

```
docfiledir = "./doc"
```

You can do the same with your source files, for example if you want your main directory to hold just `build.lua` (and probably a `README.md`):

```
sourcefiledir = "./source"
```

The system can cope with more complex layouts, for example with subdirectories. One new feature that can help with these more tricky cases is `tdsdirs`, which lets `l3build` simply copy an entire directory ‘as is’. We tell the system the name of the directory, and where it matches up with in the \TeX installation tree. For example, if we wanted to use the above source directory in its entirety, and install it into the `tex` tree, we would use

```
tdsdirs = {source = "tex"}
```

In this case, *all* of the files are used.

We will see later that there are settings that apply to tests, to creating CTAN releases, and for more advanced functions.

4 Setting up simple tests

The core mechanism for creating tests in `l3build` uses the fact that documents can write to the `.log` and extract information to verify that our code has worked. That can broadly be done in two ways: deliberately writing information to the `.log`, or using `\showoutput` or similar to place the result of some typesetting operation into the file.

What is also needed is a way to mark those parts of the `.log` that are of interest, and to normalise system-dependent information, such as paths, to make the results as portable as possible. Some of this is carried out by `l3build` itself, with the macro parts of the process implemented in the source file `regression-test.tex`. All the commands provided by the latter have all-uppercase names, to minimise the chance of clashes with normal commands.

For the case where it is possible to save a result in a macro, counter or similar, the easiest approach to testing is to write these using `\TYPEOUT`.

```
\input{regression-test}
\documentclass{article}
\usepackage{mypkg} % The package to test
\START
\TEST{A first test}{%
  \mypkfunctionA{input-tokens}%
  \outputmacro
  \TYPEOUT{\outputmacro}
}
\TEST{A second test}{%
  \mypkfunctionB
  {input-tokens}%
  {more-input-tokens}%
  \outputmacro
  \TYPEOUT{\outputmacro}
}
\END
```

Nothing before `\START` will be recorded, which makes it a good way to skip the preamble. We can skip small parts of the input using the pair `\OMIT` and `\TIMO`. The run here is stopped using `\END` as we are not interested in the typesetting of pages: this basically kills the \TeX run and saves a bit of time.

The alternative approach is to look at \TeX 's output tracing, either using a box or `\showoutput`.

```
\input{regression-test}
\documentclass{article}
\usepackage{mypkg} % The package to test
\showoutput
\begin{document}
\START
% Assume the commands produce typeset output
\mypkfunctionA{input-tokens}

\mypkfunctionB
  {input-tokens}
  {more-input-tokens}
\newpage
\OMIT
\end{document}
```

Here, we can use `\OMIT` to skip over the information at the end of a \TeX run: here we have used `\end{document}` as this allows the \LaTeX `.aux` file, *etc.*, to be created. If you are relying on information passed using this mechanism, you might need to set `checkruns = 2`

or some higher value.

The input files for tests, `.lvt` files, should be saved inside a directory `testfiles` within the project directory. The test results are then saved using

```
l3build save <names>
```

where the `<names>` are the file names of the test inputs, but with the extension omitted.

With the standard settings, tests are run using `pdf \TeX` , `X \TeX` and `Lua \TeX` , and using the \LaTeX format. Using formats other than \LaTeX is outside of the scope of this short guide, but running with multiple engines is a common requirement. To save an engine-specific test result, we use the `--engine` (or `-e`) option

```
l3build save -e<engine1>,<engine2> <names>
```

This will be needed most commonly when testing typeset output: there are fundamental differences between the three common engines. When running

```
l3build check
```

the system will use engine-specific results if they exist, and otherwise will assume that they all follow the ‘standard’ engine: this is normally `pdf \TeX` .

If you would rather just use one engine for tests, you can set

```
checkengines = {"pdftex"}
```

in your `build.lua` file. For Unicode-only work, in contrast, you might want

```
checkengines = {"xetex", "luatex"}
```

where the first entry given will then be the ‘standard’ engine.

5 Customising typesetting

There is only one command used for typesetting documentation: it can be set using the `typesetexe` setting. This is typically set to `pdflatex`: notice that this is a typesetting *command* not an *engine*.

As for tests, the number of typesetting runs can be set, using the `typesetruns` setting. More complex adjustment of the typesetting run is possible: `l3build` provides a set of basic operation functions (such as ‘run Biber’), and these can be combined to make defined workflows. This aspect requires some Lua programming and is therefore beyond the scope of this short guide.

6 Building CTAN releases

The standard settings will collect up all sources and typeset files, plus any `README.md`, and create a zip file to send to CTAN. You can also pack a TDS-ready zip: this feature is activated using the setting

```
packtdszip = true
```

Uploading to CTAN requires some settings to ‘fill out the form’ for administration. As an example, `l3build` itself has the configuration shown in Figure 1. The `[[...]]` syntax creates a multi-line string in Lua.

The information in `uploadconfig` is used by the `upload` target, which needs two key pieces of information: an email address and a release string. This will be requested by `l3build` if not given at the command line

```
l3build upload --email <email> <tag>
```

You can check that your upload is valid, without actually sending it, by using the `--dry-run` option on the command line. (This option also works for the `install` target.)

7 Advanced features

Using a mixture of Lua programming and additional variables, a wide range of effects can be achieved. These include

- Supporting plain \TeX and Con \TeX t testing
- Automatically updating version strings and copy-right in sources using the `tag` target

- Using multiple setups to run tests for different aspects of functionality
- Placing installed files in different parts of the \TeX tree
- Testing the PDFs produced by typesetting

Of these, the ability to automatically tag files is probably of the broadest interest. However, as sources files are extremely varied, this does require some Lua programming; that takes us beyond the scope of this short article. For details of this and the other more advanced features, please consult the `l3build` manual.

8 Example build.lua files

8.1 A basic project: one .dtx and one .ins

The most basic setup, following the model used by the \LaTeX Team, is to have your code and documentation in a single `.dtx` file, which has a matching `.ins` file and (probably) a `README.md`, all in the same directory. For this, the `build.lua` file can be a single line:

```
module = "mypkg"
```

That’s it: `l3build` will handle everything else based on its standard settings.

8.2 A ‘self-extracting’ .dtx file

Some people like to combine their `.ins` file into their `.dtx`; that is easy to support.¹

```
module = "mypkg"
unpackfiles {"*.dtx"}
```

8.3 Documentation separate from sources

With larger projects, you may want your documentation in one or more `.tex` files separate from the code. Assuming you also want to typeset your code, you’d go with

```
module = "mypkg"
typesetfiles {"*.dtx", "*.tex"}
```

8.4 Not using DocStrip, and non-standard file types

Not everyone wants to use DocStrip, and while it won’t hurt to leave unpacking enabled, we might well want to skip it. At the same time, we might have some non-standard file types: here some `.def` files and one `.lua` file.

```
module = "mypkg"
installfiles =
  {"*.def", "mypkg.lua", "*.sty"}
unpackfiles = {}
```

¹ I don’t recommend this structure. You are unlikely to need to send your source by email to someone, and the only real benefit of a single-source approach is for that type of ‘classical’ distribution.

```

uploadconfig = {
  author      = "The LaTeX Team",
  license     = "lpp11.3c",
  summary     = "A testing and building system for (La)TeX",
  topic       = {"macro-supp", "package-devel"},
  ctanPath    = "/macros/latex/contrib/l3build",
  repository  = "https://github.com/latex3/l3build/",
  bugtracker  = "https://github.com/latex3/l3build/issues",
  update      = true,
  description = [[
The build system supports testing and building
(La)TeX code, on Linux, macOS, and Windows
systems. The package offers:
* A unit testing system for (La)TeX code;
* A system for typesetting package documentation; and
* An automated process for creating CTAN releases.
]]
}

```

Figure 1: uploadconfig for l3build itself

8.5 Source files in different directories

Some developers like to have their sources in different directories inside their project. This likely goes with having separate files for typesetting.

```

module = "mypkg"
docfiledir = "doc"
sourcefiledir = "source"
typesetfiles = {"*.tex"}

```

9 Summary of key settings

There are a large number of more specialised settings available in l3build. Table 1 summarises some of the most commonly-used ones. There is a full list in the package documentation.

◇ Joseph Wright
Northampton, United Kingdom
joseph dot wright (at)
morningstar2.co.uk

Variable	Description
module	Name of the package
installfiles	List of files to place in the texmf tree
sourcefiles	List of sources/pre-extracted files
typesetfiles	List of sources to typeset
unpackfiles	List of .ins files to DocStrip
docfiledir	Location of typeset sources
sourcefiledir	Location of code sources
tdsdirs	Table of locations to install directly
checkengines	List of engines for test runs
checkruns	Number of (L ^A)T _E X runs for testing
typesetexe	Program to typeset documentation
typesetruns	Number of (L ^A)T _E X runs for typesetting
packtdsdir	Switch to build TDS-style zip file
uploadconfig	Table of information for uploading

Table 1: Summary of key settings

bib2gls: Standalone entries and repeated lists (a little book of poisons)

Nicola L. C. Talbot

Abstract

Most articles that describe how to use the `glossaries` package consider a single sorted list or possibly multiple lists where each list has a different set of entries (terms, symbols, abbreviations, etc.). However, some documents may instead have each term described in the main matter, with references to the term linking back to that point in the document rather than to a summary list. Alternatively (or additionally) a document may have multiple lists consisting of the same set of entries ordered in different ways.

The examples here were compiled with `glossaries` v4.49 [9], `mfirstuc` v2.07 [10], `glossaries-extra` v1.47 [8] and `bib2gls` v2.8 [6]. Some features and commands are not available in earlier versions. Also some earlier versions have bugs causing unexpected results.

1 The Book of Poisons

The Book of Poisons by Stevens and Bannon [3] is an excellent guide for crime fiction writers. It also provides a good example of a mixture of standalone entries (where each entry, in this case a toxic substance, is described in the main matter rather than in a summary list) and repeated lists in the back matter with just the name and no description (ordered by method of administration, by form, by symptoms, by toxicity and by time taken to react). It also has the more traditional lists (a bibliography and a simple glossary of medical terms) which aren't under discussion here. Finally, there's an index, which could be implemented using the standard `\index` and associated commands, but since it's possible to create the index as a by-product of using `bibgls` for the standalone entries, this will also be covered here.

The book consists of numbered chapters for each particular type of poison (household chemicals, plants, animals and so on). Each chapter is divided into unnumbered (sub) sections describing each poison, using a consistent structure listing:

Scientific Name (optional) for example 'Cantharis vericatoria' is the scientific name of cantharidin;

Other Name (optional) for example 'Spanish fly' is a colloquial term for cantharidin;

Other Similar (optional) a list of similar substances that don't have their own entries; for example, 'choline' is listed as similar to aspirin;

Toxicity a number from 1 (low) to 6 (highly toxic);¹

¹ The toxicity level relates to the amount required for a lethal dose. It's not a measure of symptoms. A level 6 toxin

Form/Deadly Parts plants have a deadly parts item (identifying which part of the plant is toxic), while other entries list the form (for example, liquid or gas) that the toxin takes;

Effects and Symptoms such as headache or nausea;

Reaction Time how long for symptoms to occur;

Antidotes and Treatments whether or not an antidote is available or if there is known treatment;

Notes (optional) some additional information, sometimes including a case history.

I don't want to bog down the examples with unrelated style code, so I'm just going to use the standard `description` environment. The entry data itself will also be significantly pared down to the following, which will correspond to glossary entry fields (custom ones will need to be defined):

name the toxin name (as it will appear in the section title);

description information about the toxin;

toxicity (custom field) a number;

method (custom field) the method of administration, a list of one or more elements from the set: breathed, injected, membrane absorption, skin absorption, smoked, swallowed.

symptom (custom field) the symptoms, which should be a list of one or more elements from any of the symptom classification subsets, including:

vital signs: bradycardia, hyperthermia, hypertension, hypotension, hypothermia, tachycardia;

head, eyes, nose, throat: bad/unusual taste, etc.

(There are too many to list individually here.)

I've omitted form (aerosol etc.) as the 'Ordered by Form' list can be achieved in the same way as the 'Ordered by Method' list. Similarly for the reaction time, which could be implemented with a numeric identifier like the toxicity.

2 Manual method

I'm going to start with an example document that doesn't use the `glossaries` package. Since we live in a digital age where some people prefer to read books on their devices, I've used the `hyperref` package [2]. The chapters are numbered in the main matter, but sections aren't numbered anywhere. The sections and subsections are too numerous to list in the table

can cause death from a pinprick amount with few symptoms, whereas a level 2 toxin requires a much higher dose to kill but can have debilitating long-term symptoms from a non-lethal dose.

of contents, but they would be useful in the PDF bookmarks. This can be achieved by setting the `tocdepth` counter to 0 (to only show chapters in the table of contents), setting the `secnumdepth` counter to 0 (to only show numbers for chapters) and using `hyperref`'s `bookmarksdepth` option to set the depth for the PDF bookmarks.

The `hyperref` package automatically creates an anchor at the start of each page where the anchor name is obtained from the formatted page number. The `\frontmatter` command resets the page counter to 0 and changes the page number format to lowercase Roman numerals. Thus, the first page of the table of contents is ‘i’ and so the anchor for that page is `page.i`. The `\mainmatter` command similarly resets the page and changes the format to arabic (the default page number format) so the anchor for the first page of the first chapter will be `page.1`.

Unfortunately, the title page (and its reverse) also use the default number format so, even though the numbering may be hidden by the empty page style, the page anchor is still created (`page.1` for the title page and `page.2` for its reverse). Since the page numbers are hidden, the simplest solution is to select a different number format that isn't used for any of the other pages. In this case, I've used the `alph` format. This means the first two pages have the anchors `page.a` and `page.b`. They're not required anywhere in the document but this prevents a conflict and ensures that any references to pages 1 or 2 in the index or glossaries (once they are added) link to the correct page.

The ‘Ordered by’ lists mostly have very narrow columns so I've used the `multicol` package [1]. As we'll see, the ‘Ordered by Toxicity’ listing rounds 4.5 down to 4, which is what Stevens and Bannon do (although their toxicity list is in the opposite order).

Some of the scientific names are New Latin names, so I've provided a semantic command to typeset them:

```
\newcommand*{\latinname}[1]{\emph{#1}}
```

This example document contains only one toxin (ammonia), but it's already lengthy as it has a large number of symptoms.

```
\documentclass{book}
\usepackage{multicol}
\usepackage[bookmarksdepth=2]{hyperref}
\title{A Little Book of Poisons}
\author{Ann Author}
\setcounter{secnumdepth}{0}
\setcounter{tocdepth}{0}
\newcommand*{\latinname}[1]{\emph{#1}}
\begin{document}
\pagenumbering{alph}\pagestyle{empty}
```

```
\maketitle
\frontmatter\pagestyle{headings}
\tableofcontents
\mainmatter
\chapter{Household Poisons}
\section{Chemicals}
\subsection{Ammonia}
\begin{description}
\item[Other] Ammonium hydroxide.
\item[Toxicity] 4.5
\item[Method] Breathed.
\item[Symptoms] Tachycardia, blindness, lip/mouth
irritation, burns, flushing, coughing, pulmonary
edema, abdominal or stomach pain, restlessness,
collapse, and pain.
\item[Description] Some information about ammonia.
\end{description}
\backmatter
\chapter{Ordered by Administration}
\begin{multicols}{3}
\section{Breathed}
\begin{itemize}
\item[] Ammonia
\end{itemize}
\end{multicols}
\chapter{Ordered by Symptoms}
\begin{multicols}{3}
\section{Vital Signs}
\subsection{Tachycardia}
\begin{itemize}
\item[] Ammonia
\end{itemize}
% Lots of other sections omitted for brevity
\end{multicols}
\chapter{Ordered by Toxicity}
\begin{multicols}{3}
\section{Toxicity Rating 6}
\section{Toxicity Rating 5}
\section{Toxicity Rating 4}
\begin{itemize}
\item[] Ammonia
\end{itemize}
\section{Toxicity Rating 3}
\section{Toxicity Rating 2}
\end{multicols}
\end{document}
```

3 Standalone entries

For this next example, I'm going to consider a cut-down version of the main matter in order to illustrate standalone entries. The simplest approach can be achieved with the base `glossaries` package, although this has limitations. For now, each entry just has a name, toxicity and description. The toxicity could be stored in one of the custom keys, such as `user1`, but I've decided to define a new key called `toxicity`:

```
\glsaddstoragekey{toxicity}{\toxicity}
```

This both defines the `toxicity` key and provides a command called `\toxicity` to access the value.

Since the descriptions are likely to be quite lengthy and may contain paragraph breaks, they are best defined with `\longnewglossaryentry`:

```
\longnewglossaryentry{ammonia}
  {name=ammonia,toxicity=4.5}
  {Some information about ammonia.}
\longnewglossaryentry{nutmeg}
  {name=nutmeg,toxicity=3}
  {Some information about nutmeg.}
\longnewglossaryentry{lsd}
  {name=LSD,toxicity=2}
  {Some information about LSD that includes
  a reference to nutmeg.}
\longnewglossaryentry{botulinum}
  {name={botulinum},toxicity=6}
  {Some information about botulism.}
```

These are all defined in the file `toxins.tex`, which needs to be input in the preamble (with either `\input` or `\loadglsentries`).

Since all the sections follow a set format, I'll define a command (`\toxin`) that simply takes a label and displays the complete section. To accommodate the mixture of sections and subsections, we have an associated command (`\toxinsection`) that can be redefined at the start of a chapter where necessary:

```
\newcommand{\toxinsection}{\section}
\newcommand*{\toxin}[1]{%
  \toxinsection{\glsentrytitlecase{#1}{name}}
  \begin{description}
    \item[Toxicity] \toxicity{#1}
    \item[Description] \glsentrydesc{#1}
  \end{description}
}
```

This converts the `name` field to title case in the section heading using `\glsentrytitlecase`, which internally uses `\capitalisewords` provided by the `mfirstuc` package.

When writing in English, words such as ‘and’ should only be capitalized when they occur at the start of the title. Since such exceptions are language dependent, they aren't implemented by default. The `mfirstuc-english` package provides the common English exceptions:

```
\usepackage{mfirstuc-english}
```

This doesn't affect the document so far, but it will later when the ‘order by symptoms’ list is added.

The main matter is now much shorter:

```
\chapter{Household Poisons}
\renewcommand{\toxinsection}{\subsection}
\section{Chemicals}
\toxin{ammonia}
\section{Food Poisoning}
\toxin{botulinum}
```

```
\chapter{Plants}
\renewcommand{\toxinsection}{\section}
\toxin{nutmeg}
\chapter{Street Drugs}
\toxin{lsd}
```

When just considering the main matter, this doesn't seem like a significant improvement to the first example. It is easier to move the sections around, but the title case-changing can't be implemented in the PDF bookmarks. So `\glsentrytitlecase` will expand to the original lowercase value in the bookmark.

It would be useful if the nutmeg reference in the LSD description had a hyperlink to the nutmeg section (created with `hyperref`). Such hyperlinks are normally achieved with the `glossaries` package using commands like `\gls`. However, the target anchor is typically in the glossary (implemented by the `glossary` style), but there isn't a glossary in this document.

The `glossaries-extra` package provides a solution where you can use `\glsxtrglossentry`:

```
\toxinsection{\glsxtrglossentry{#1}}
```

This command expands to just `\glsentryname` for the PDF bookmark, so there's no difference in this respect. However, within the document text, this command creates the `hypertarget` and displays the name in the same way as the `glossary` styles. So I can adjust the case using the `glossname` attribute:

```
\glssetcategoryattribute{general}{glossname}
  {title}
```

This only applies to the section title on the page, not in the bookmarks.

The LSD entry can now be modified to include a hyperlink to the nutmeg section:

```
\longnewglossaryentry{lsd}
  {name=LSD,toxicity=2}
  {Some information about LSD that includes
  a reference to \gls{nutmeg}.}
```

There are over two hundred toxins listed in the book. At the moment, all my definitions are stored in my `toxins.tex` file, but I could store them in a `bib` file instead. This would make it easier to share the data across multiple documents. For example, the `bib` file may include brief summaries that can be used as a description in other shorter documents as well as the long description for this catalogue of toxins. For example:

```
@entry{nutmeg,name={nutmeg},
  summary={Short description of nutmeg.},
  longdescription={Long description of nutmeg.}}
```

The document then can choose the appropriate field for the description using field aliases. I'm not going to do this here in order to keep the examples as simple as possible.

My `toxins.tex` file can easily be converted to `toxins.bib` using `convertgls2bib`:

```
convertgls2bib toxins.tex toxins.bib
```

Then I need to replace the code that inputs `toxins.tex` with:

```
\GlsXtrLoadResources[src=toxins]
```

and add the record package option.

The `\latinname` command can be provided in the preamble of the `bib` file to ensure that it's defined:

```
@preamble{
"\providecommand*\latinname}[1]{\emph{#1}}"}

```

Again, this is something that's useful if the `bib` file is shared with other documents, but isn't essential for this example as the command is already defined in the document preamble.

Normally, `bib2gls` will select entries from the `bib` file if they have records in the `aux` file (which are created with commands like `\gls` or `\glsadd`) or if they depend on selected entries; for example, nutmeg needs to be selected if LSD is selected, since the LSD entry depends on the nutmeg entry.

In this case, though, `bib2gls` doesn't select any entries because nothing creates a record. (The `\gls` command in the LSD description will only create a record if the LSD entry is selected and has its description expanded in the document, but there are no LSD records, so LSD won't be selected.)

I could instruct `bib2gls` to select all entries, but some entries may need to be omitted. For example, the publisher may decide that the print cost for the physical edition is too large, so some entries may need to be dropped.

One approach is to use `\glsadd` in the definition of `\toxin`:

```
\newcommand*\toxin}[1]{%
\glsadd{#1}% index this entry
\toxinsection{\glsxtrglossentry{#1}}
\begin{description}
\item[Toxicity] \toxicity{#1}
\item[Description] \glsentrydesc{#1}
\end{description}
}
```

This ensures that each entry listed in the book will be selected by `bib2gls`.

The problem of the case-conversion for PDF bookmarks can now be solved as there are some resource options that instruct `bib2gls` to change the case of field values:

```
name-case-change=title
```

However, this will cause `\gls{nutmeg}` to start with a capital unless the `text` field is set to the original value. This can be done with:

```
replicate-fields={name=text}
```

This will copy the value of the `name` field into the `text` field. (If the target field is already set, the default behaviour is to leave it unchanged.) Replication is always performed before case-changing, regardless of the resource option ordering. If the source field (`name` in this case) is not set, the default is to do nothing. But in this case, I want to obtain the value from the fallback if `name` is missing:

```
replicate-missing-field-action=fallback
```

This means I can now dispense with the `glossname` attribute.

4 Comma-separated list fields

So far I haven't included the method and symptoms in my entry definitions. I can define two more custom keys in the same way as for `toxicity`:

```
\glsaddstoragekey{method}{\method}
\glsaddstoragekey{symptom}{\symptom}

```

I could simply set the values to free-form text:

```
@entry{ammonia,name={ammonia},
toxicity={4.5},
description={Some information about ammonia.},
method={Breathed.},
symptom={Tachycardia, blindness, lip/mouth
irritation, burns, flushing, coughing,
pulmonary edema, abdominal or stomach pain,
restlessness, collapse, and pain.}
}
```

However, I decided to adopt a different approach. First, I created a file called `methods.bib` containing:

```
@index{breathed}
@index{injected}
@index{membraneabsorption,
name={membrane absorption}}
@index{skinabsorption,
name={skin absorption}}
@index{smoked}
@index{swallowed}

```

The method fields are all going to be comma-separated lists of the method entry labels.

The symptoms are defined in a similar way (in a file called `symptoms.bib`) but they have an unknown `topic` field, which will be ignored by `bib2gls` unless it is aliased or defined in the document (see later). For example:

```
@index{hyperthermia,
name={fever\MFUwordbreak{\slash}hyperthermia},
text={fever},
topic={vital signs}
}
@index{hypothermia,
name={low body temperature\MFUwordbreak
{\slash}hypothermia},
text={hypothermia},

```

```

topic={vital signs}
}
@indexplural{burn,topic={skin}}
@index{collapse,topic={whole body}}

```

Note that while most of these are defined using `@index` there are some defined with `@indexplural`. These entries will default to having the categories set to ‘index’ and ‘indexplural’. This will cause complications later, so all entries will be assigned to the ‘general’ category with the resource option:

```
category=general
```

The names will be converted to title case so the slash needs to be marked up as a word break using `\MFUwordbreak`, otherwise the word following the slash won’t have its case changed. You may prefer to define a string:

```
@string{SLASH="\MFUwordbreak{\slash}"}
```

and use string concatenation:

```
name={fever} # SLASH # {hyperthermia},
```

The ammonia entry can be defined as:

```
@entry{ammonia,name={ammonia},
toxicity = {4.5},
description={Some information about ammonia.},
method = {breathed},
symptom = {tachycardia,blindness,
mouthirritation,burn,flushing,
coughing,pulmonaryedema,abdominal,
restlessness,collapse,pain}
}
```

The line breaks in the comma-separated lists above can be problematic since these lists will internally be passed to `\@for` in the document, but it is possible to get `bib2gls` to strip the whitespace, if you’d rather not omit them.

There are two options, `labelify` and `labelify-list`, that can be used to strip any content that can’t occur in a label. The former is intended for fields containing a single label and the latter is for fields containing a comma-separated list of labels. Both are governed by `labelify-replace`, so the following can be used to strip any whitespace:

```
labelify-list={method,symptom},
labelify-replace={\string\s+{}}
```

This means that you can introduce extra space in the `bib` file to make it more readable. Further, since this option also automatically removes empty items, it’s also possible to replace `\and` with a comma:

```
labelify-list={method,symptom},
labelify-replace={
\string\s+and\string\s+}{,},
\string\s+{}}
```

This means that the list ‘A and B’ becomes ‘A,B’. The list ‘A, B, and C’ becomes ‘A,B,C’; the empty element is then stripped, leaving ‘A,B,C’.

So the `symptom` field can be set as:

```
symptom = {tachycardia, blindness,
mouth irritation, burn, flushing,
coughing, pulmonary edema, abdominal,
restlessness, collapse, and pain}
```

This is not only easier to read but also makes it suitable for use without the `symptoms.bib` file.

The `\glseeelist` command (provided by the base glossaries package) formats a comma-separated list of entry labels. This was designed for the use of cross-referencing with the `see` field [5], but may be used with any list of entry labels. If you want to ensure that the argument is fully expanded, use `glossaries-extra`’s `\glxtrseelist` instead (which internally uses `\glseeelist`).

The `\toxin` command can be modified to include formatted lists of symptoms, but `\glseeelist` doesn’t index so the method and symptom entries won’t be selected. In order to ensure that they are selected, `bib2gls` needs to be told that the `method` and `symptom` fields contain lists of dependent entries:

```
dependency-fields={method,symptom}
```

The `method` and `symptom` entries don’t have any targets (at the moment) so the hyperlinks need to be suppressed. Also the `name` field has had a case-conversion applied. Both problems can be fixed by redefining `\glseeitem` to just use `\glseentrytext`:

```
\renewcommand*\glseeitem[1]
{\glseentrytext{#1}}
```

If you want the first item capitalised you can redefine `\glseeefirstitem`:

```
\renewcommand*\glseeefirstitem[1]
{\Glseentrytext{#1}}
```

The separator between the last two items in the list is given by `\glseeelastsep`, which defaults to `\and`. If you want to change this to use ‘and’ instead:

```
\renewcommand*\glseeelastsep{ and }
```

If your preference is for an Oxford comma you will also need:

```
\renewcommand*\glseeelastoxfordsep{, and }
```

This may seem a bit redundant since the end result is much the same as the original field value, but the hyperlink will be added in a later example once the corresponding lists have been created.

If you want the `method` and `symptom` elements to be alphabetically ordered, then you can instruct `bib2gls` to do this with the `sort-label-list` option:

```
sort-label-list={
[method,symptom]:en:glseentryname}
```

This indicates that the `method` and `symptom` fields are comma-separated lists and that `bib2gls` should reorder these lists according to the `en sort` method (English) where the sort value is obtained by encapsulating the list element with `\glstentryname` (which `bib2gls` recognises). This ensures that the list is ordered by the displayed name rather than the label.

5 The index

Since `bib2gls` sorts by default, a convenient side-effect is that the index can easily be added at the end of the document using the `bookindex` glossary style (which doesn't show descriptions). As with the other provided glossary styles, this will create a hypertext, which will cause a conflict, but this can be switched off with the `target=false` option:

```
\printunsortedglossary[target=false,title=Index]
```

The `bookindex` style isn't loaded by default, so you'll also need to specify it explicitly:

```
\usepackage[record=nameref,
  stylemods=bookindex,
  style=bookindex]{glossaries-extra}
```

I've set the style as a package option as the other glossaries discussed later will also use this style. If you want letter groups, remember to use the `--group` (or `-g`) switch when you invoke `bib2gls`.

If any entries have the `see`, `seealso` or `alias` fields set, `\glseeitem` will need to be restored to its original value for the index. The simplest way to do this is to localise the redefinition. So instead of redefining it in the preamble, it can be redefined within a scoped context within the definition of `\toxin`. Since environments automatically add scoping, the redefinition can be placed inside the `description` environment.

Some of the entry descriptions may span multiple pages, in which case you may prefer to have a page range in the index. This can be achieved with explicit location ranges. The position of `\glsadd` also needs an adjustment. This command switches to horizontal mode (as complications can occur in certain situations otherwise), which means that the page number could be off if the section heading is moved to the start of the next page. If `\glsadd` is placed after the heading then it will cause an unwanted space before the start of the `description` environment. The best solution is to place it in the section title and use the optional argument for the bookmark.

```
\newcommand*{\toxin}[1]{%
  \toxinsection[\glstentryname{#1}]{%
    \glstrglossentry{#1}\glsadd[format=]{#1}}
  \begin{description}
    \let\glseeitem\glstentrytext
```

```
\let\glseeitem\glstentrytext
\item[Toxicity] \toxicity{#1}
\item[Method] \glstrseelist{\method{#1}}.
\item[Symptoms] \glstrseelist{\symptom{#1}}.
\item[Description]
\glstentrydesc{#1}\glsadd[format=]{#1}
\end{description}
}
```

There is a problem with the `method` and `symptom` entries. They haven't been indexed anywhere in the document so they don't have a page list. This could be solved by redefining `\glseeitem` to use `\glstext` instead of `\glstentrytext`, but this increases the complexity of the document build and could lead to lengthy page lists in the index, especially for common methods (such as swallowing, which applies to most toxins) or symptoms (there's a fairly sizable list for convulsions). Since the final version of this example will have lists of methods and symptoms, there's no need for them to appear in the index.

There are two basic approaches to removing entries from a list: put them in a different glossary or filter them when displaying the list. The first approach can be a bit tricky if all entries are being processed by a single resource command. One way would be to rename `toxins.bib` to `main.bib` and use the resource option:

```
type={same as base}
```

This will set the `type` field to the file basename, without the `bib` extension. So the entries defined in `main.bib` will be assigned to the default `main` glossary, the entries defined in `methods.bib` will be assigned to the glossary identified by the label `methods` (which will need to be defined), and similarly entries defined in `symptoms.bib` will be assigned to a user-provided `symptoms` glossary.

The second approach keeps all the entries in one glossary but uses the hook that's provided to help skip entries. This is discussed in more detail in a previous article [5], but essentially, in order to avoid problems involved in using iterative code within a tabular-like environment (which some glossary styles use), the entries are first iterated over outside of the glossary and the glossary contents are appended to an internal control sequence. There's a hook that's used in this stage which can skip the current iteration to prevent an entry from being appended.

I've defined a custom command to filter entries with empty locations because it may be useful for other lists (either in this document or placed in a package for the use of other documents):

```
\newcommand{\filteremptylocation}[1]{%
  \glstrifhasfield*{location}{#1}
```

```
{}% has location field
{\printunrtglossaryskipentry}%
}
```

The process hook will be `\let` to this command.

Although each entry in the index has a location list, it might be useful to have the entry name as a hyperlink to its section in the main part of the document. The `bookindex` style provides a command that's used to format the entry name, which takes the entry label as its argument. Again I'm defining a custom command which the style command can locally be `\let` to. This simply encapsulates the name with a hyperlink:

```
\newcommand*{\linkedbookname}[1]{%
\glshyperlink[\glossentryname{#1}]{#1}}
```

The starred form of `\printunrtglossary` has a mandatory argument where the code to initialise the hooks can be placed. This is scoped so it won't alter any subsequent lists.

```
\printunrtglossary*[target=false,title=Index]
{\let\printunrtglossaryentryprocesshook
\filteremptylocation
\let\glxtrbookindexname\linkedbookname}
```

Unfortunately the index now has terms in title case, which doesn't look quite right. I used `name-case-change` to remove the non-expandable case-changing from the PDF bookmarks, but this has now had an unwanted side-effect. To overcome this problem, I can create a field to store the bookmark title:

```
\glsaddstoragekey{bookmark}{-}{\pdfname}
```

Now, instead of copying the name to the `text` field, I copy it to this new `bookmark` field:

```
replicate-fields={name=bookmark}
```

and instead of `name-case-change` I now need to use:

```
field-case-change={bookmark=title}
```

Any formatting commands can be stripped by instructing `bib2gls` to interpret the `bookmark` field:

```
interpret-fields={bookmark}
```

Alternatively, the `\pdfstringdefDisableCommands` command from `hyperref` can be used to discard problematic tokens.

The `\toxin` command needs to be modified to use this field:

```
\toxinsection[\pdfname{#1}]
{\glxtrglossentry{#1}\glsadd[format=]{#1}}
```

The `glossname` attribute is needed again, but it has to be switched off before the index. This can be done either by scoping the attribute assignment or by undefining the attribute:

```
\glsunsetcategoryattribute{general}{glossname}
```

6 Synonyms and related terms

The scientific names, alternative names and related substances could be added in a similar way to the symptoms and methods, but it would be useful to have these terms in the index.

There are three cross-referencing fields available [5]: `see`, `seealso` and `alias`. The first two take a comma-separated list of labels. The `alias` field can only have a single label as the value.

I'm going to use the `alias` field for the scientific name, the `see` field for alternative names ('Other') and the `seealso` field for similar substances ('Related'). For example:

```
@index{cbot,
name={\latinname{Clostridium botulinum}},
alias={botulinum}}
@index{botulism,see={botulinum}}
@index{botox,see={botulinum}}
@index{lsd-long,
name={lysergic acid diethylamide},
alias={lsd}}
@index{lysergide,see={lsd}}
@index{ammoniumhydroxide,
name={ammonium hydroxide},
alias={ammonia}}
```

The scientific name for nutmeg is *Myristica fragans*:

```
@index{mfragans,
name={\latinname{Myristica fragans}},
alias={nutmeg}}
```

But some other nutmeg species are also listed:

```
@index{margentea,
name={\latinname{Myristica argentea}},
alias={nutmeg}}
@index{mmalabarcia,
name={\latinname{Myristica malabarcia}},
alias={nutmeg}}
```

This complicates things a little as they need their common name as well:

```
@index{Papuan-nutmeg,
name={Papuan nutmeg},
see={margentea}}
@index{Bombay-nutmeg,
name={Bombay nutmeg},
see={mmalabarcia}}
```

These new entries aren't referenced anywhere in the document, nor are the selected entries dependent on them, so I need to change the selection criteria to include entries that cross-reference the selected entries:

```
selection={recorded and deps and see}
```

The `\toxin` command needs to be adjusted so that it shows the other names. Entries only have fields that store dependent entries (`see`, etc.), not

the reverse. Whilst it is possible to iterate over all entries to find the synonyms, it's not very efficient.

The resource options `save-from-alias`, `save-from-see` and `save-from-seealso` provide a solution. These define, respectively, the fields `from-alias`, `from-see` and `from-seealso` that contain the required information. The `\toxin` command now needs to check if any of these fields have been defined. For example:

```
\glxtrifhasfield*{from-see}{#1}{\item[Other]
\glxtrseelist\glscurrentfieldvalue}{}
```

The `from-alias` field could be dealt with in the same way, but the common names of the other nutmeg species won't show.

A simple solution is to define a command that checks the `from-see` field that can be used to encapsulate the items in the list:

```
\newcommand{\seeitemandother}[1]{%
\glentrytext{#1}%
\glxtrifhasfield{from-see}{#1}%
{ (\glentrytext{\glscurrentfieldvalue})}%
}%
}
```

and also an analogous command `\Seeitemandother` that uses `\Glentrytext` instead, used for the first item in the list. There's a custom command to switch to these commands and display the list:

```
% \toxinitelist{label}{field}{title}
\newcommand{\toxinitelist}[3]{%
\glxtrifhasfield*{#2}{#1}%
{\formattoxinitelist{#3}
\glscurrentfieldvalue}%
}%
{ }% field not defined
}
```

I've made an inner command to make it easier to adjust the actual formatting:

```
% \formattoxinitelist{title}{label list}
\newcommand*{\formattoxinitelist}[2]{%
\item[#1] {\let\glseeitem\seeitemandother
\let\glseeefirstitem\seeitemandother
\glxtrseelist{#2}}.
}
```

The `description` environment within the `\toxin` definition can now be written in a more compact form:

```
\begin{description}
\toxinitelist{#1}{from-alias}{Scientific Name}
\toxinitelist{#1}{from-see}{Other}
\toxinitelist{#1}{from-seealso}{Related}
% other items as before
\end{description}
```

The `\booklinkname` command is now going to cause a problem in the index as these new cross-reference terms don't have a target. There are a

number of ways around this. For example, a conditional can be added to use a hyperlink only if the `toxicity` field is set:

```
\newcommand*{\linkedbookname}[1]{%
\glxtrifhasfield{toxicity}{#1}%
{\glshyperlink[\glossentryname{#1}]{#1}%
{\glossentryname{#1}}%
}
```

It would, however, be more convenient if the other names could have a hyperlink to their main entry. This could be done by consulting the `alias`, `see` and `seealso` fields in turn, which leads to a complicated set of nested conditionals and also doesn't work for Bombay nutmeg and Papuan nutmeg.

The `save-crossref-tail` resource option is useful here as it will save the tail label from a cross-reference trail in the `crossref-tail` field. This requires only one extra conditional:

```
\newcommand*{\linkedbookname}[1]{%
\glxtrifhasfield{toxicity}{#1}%
{\glshyperlink[\glossentryname{#1}]{#1}%
{%
\glxtrifhasfield{crossref-tail}{#1}%
{\glshyperlink[\glossentryname{#1}]
{\glscurrentfieldvalue}}%
{\glossentryname{#1}}%
}%
}
```

7 Order by toxicity

The `glossaries` package allows multiple glossaries. A default one is provided with the label `main`. When a new glossary is defined, an internal command is constructed from the name `glolist@⟨type⟩` where `⟨type⟩` is the glossary label. Whenever a new entry is defined, its label is appended to the glossary's internal list command's replacement text with a comma separator. It's this list that `\printunsrtglossary` iterates over.

The `glossaries-extra` package provides a command that copies an entry label to another glossary list. This means that the entry is only defined once and its `type` field is set to its original glossary (this can be considered the entry's primary glossary) but the entry will also appear in the other glossary's list. (This approach can't be used with `makeindex` or `xindy`.)

For example:

```
\newglossaryentry{sample}
{name=sample,description={}}
\newglossary*{another}{Another}
\glxtrcopytogglossary{sample}{another}
\begin{document}
\printunsrtglossaries
\end{document}
```

This will display two glossaries, both containing the sample entry. This method allows a duplicate list, which may have a different order, without the overhead of duplicate entry definitions, and it's this method that's employed with `bib2gls`'s secondary resource option.

The syntax for this option is:

```
secondary=<sort>:<field>:<type>
```

where `<sort>` indicates the sort method and `<type>` is the glossary label.

The `<field>` part is optional and indicates the field to use for sorting. The previous article [7] discussed sorting and, in particular, the system of fallbacks used to determine the value used for comparison if the `sort` field isn't set.

A by-product of the sorting function (regardless of the field used for sorting) is that it assigns the actual sorting value (possibly obtained from fallbacks, with word breaks marked, suffixes appended etc.) to the `sort` field. This means that if you want a secondary glossary, you will need to choose a different field for the sort value unless you want to reuse the same sort values from the primary sort (which would usually be redundant). The secondary sort method will store its actual sort value in the internal field `secondarysort` to avoid conflict, in the event that you need to access both values in your document.

I've used the `--group` switch to add letter groups to my index. You may recall from previous articles that this will store the group label (obtained as a by-product of sorting) in the `group` field. To avoid conflict the secondary sort function will store the group label in the `secondarygroup` internal field, and `bib2gls` will append the required redefinition to the secondary glossary's preamble so that it will automatically switch to the `secondarygroup` field.

So to have a secondary glossary ordered according to the `toxicity` field (from highest to lowest):

```
secondary=integer-reverse:toxicity:bytoxicity
```

Here I've indicated that the secondary glossary has the label `bytoxicity`. If you inspect the `glstex` file, you should find the line:

```
\provideignoredglossary*{bytoxicity}
```

This means that the glossary will be provided if you don't define it; however, it will be an 'ignored' glossary so it will use the default title and won't be picked up by `\printunsortedglossaries`.

If you want to explicitly define this glossary in the document you can add the following (before the resource command):

```
\newglossary*{bytoxicity}{Order by Toxicity}
```

I've used a numeric sort and you may recall from the previous article [7] that this will result in the

group label `glsnumbers` (which has the associated language-sensitive title 'Numbers'). This label is actually set indirectly as can be seen from an inspection of the `glstex` file:

```
\bibglsssetnumbergrouptitle{6}{6}{bytoxicity}
\bibglsssetnumbergrouptitle{4}{4}{bytoxicity}
\bibglsssetnumbergrouptitle{2}{2}{bytoxicity}
\bibglsssetnumbergrouptitle{3}{3}{bytoxicity}
\bibglsssetnumbergrouptitle{0}{0}{bytoxicity}
```

This command is provided at the start of the file:

```
\providecommand{\bibglsssetnumbergrouptitle}[1]{%
  \glstrsetgrouptitle
    {\bibglsnumbergroup#1}
    {\bibglsnumbergrouptitle#1}}
```

The group label is obtained from the control sequence `\bibglsnumbergroup` (which must fully expand). This command is also provided:

```
\providecommand{\bibglsnumbergroup}[3]{glsnumbers}
```

It's this definition that causes all entries that have been sorted numerically to be placed in the 'Numbers' group. For this example, I'd like the groups to correspond to the toxicity levels, so I need to define this command before the `glstex` file is input:

```
\newcommand{\bibglsnumbergroup}[3]{#1}
```

The corresponding title is obtained from a command that is also provided in the `glstex` file:

```
\providecommand{\bibglsssetnumbergrouptitle}[3]{%
  \protect\glsnumbersgroupname}
```

Again I can provide my own definition in the document to override this. For example:

```
\newcommand{\bibglsssetnumbergrouptitle}[3]{Toxicity
Rating #1}
```

The bookindex headers are formatted according to:

```
\glstrbookindexformatheader{<title>}
```

This defaults to a centred format. I've decided to provide a different format, but I don't want to apply it to the index as well, so the redefinition will need to be scoped.

The custom header formatting is quite simplistic for this example:

```
\newcommand{\orderbyheader}[1]{%
  \par{\raggedright\bfseries\large #1\par}}
```

This can be adjusted as required.

As with the index, the list includes entries that don't have the `toxicity` field set. These will end up with a sort value of 0 and can be filtered using a method similar to that employed for the index. However, it's simpler to get `bib2gls` to filter them: `secondary-not-match={toxicity={}}`

The glossary can be displayed in the usual way:

```
\printunsortedglossary[type=bytoxicity,
  target=false]
```

This will include the page list for each entry, which you may prefer to omit. The `nonumberlist` option can be added to suppress it. On the other hand, it can be useful to have a way to link back to the main definition. As with the index, a hyperlink can be added to the name using the same method as earlier. This is useful for the reader of the PDF version, but for the paperback version it might be helpful to just list the primary page number (rather than the complete list). For this, I've defined a custom location format command:

```
\newcommand{\mainfmt}[1]{\glxnumberformat{#1}}
```

This uses the default formatting, but I can use it to identify the principal page in my custom `\toxin` command:

```
\section[\pdfname{#1}]{\glxstrglossentry{#1}%
\glsadd[format=mainfmt]{#1}%
\glsadd[format=(]{#1}}
```

This primary location format `\mainfmt` is identified with:

```
primary-location-formats=mainfmt
```

I can instruct `bib2gls` to move the primary locations out of the normal location list and into a field called `primarylocations`:

```
save-primary-locations=remove
```

To ensure that `\printunsrtglossary` uses this field for the location list:

```
\renewcommand{\GlsXtrLocationField}
{primarylocations}
```

To prevent the primary locations from being merged with the explicit range formation:

```
bib2gls -g --retain-formats mainfmt toxinbook
```

where the document is in the file `toxinbook.tex`.

8 Order by method

The method list is superficially similar. It's not possible to use multiple secondary options in one resource command, but it's possible to have a second resource set that copies entries to another glossary. First define the new glossary:

```
\newglossary*{bymethod}{By Method}
```

Now for the second resource command:

```
\GlsXtrLoadResources[
src=methods,type=bymethod,
selection={selected before},
action=copy
]
```

This selects all the entries in `methods.bib` that were previously selected, sorts them and copies their labels to the `bymethod` glossary. Internally (that is, within `bib2gls`'s Java code) a new object representing each

entry is created with the information obtained by reparsing the `bib` file. So any modifications made by the previous resource set won't be present in this resource set (unless the modifications are repeated). This means that the `sort` field will be missing again: the value won't be retained from the previous resource set. However, from \LaTeX 's point of view, each entry is defined once.

This now provides a target for the methods, so the method list in `\toxin` can have a hyperlink for each entry:

```
\renewcommand*{\glseeitem}[1]{%
\glshyperlink[\glseentrytext{#1}]{#1}}
\renewcommand*{\glseeifirstitem}[1]{%
\glshyperlink[\Glsentrytext{#1}]{#1}}
```

So far this just lists the methods. The sub-list of relevant toxin entries needs to follow each method name. This looks like a hierarchical glossary but it has child entries with multiple parents. The structure is essentially a set of nested glossaries with an outer glossary (the `bymethod` glossary) where each element is followed by an inner glossary.

The glossary process hook can be used to create throwaway glossaries:

```
\let\printunsrtglossaryentryprocesshook
\provideignoredglossary
```

This creates a glossary with the same label as the entry, but the hook will have to be reverted before the inner glossaries are processed.

There's another hook that's used after processing and just before the glossary is displayed. This can be used to populate the throwaway glossaries and reset the process hook. First a command that does the action:

```
\newcommand{\populatemethods}{%
\renewcommand
\printunsrtglossaryentryprocesshook[1]{%
\forGlsentries[main]{\thislabel}%
{\glxtrforcsvfield*{\thislabel}{method}
{\populatedo}}}%
}
```

with a list handler:

```
\newcommand{\populatedo}[1]{%
\glxtrcopytoglossary{\thislabel}{#1}%
}
```

Then the hook needs to be assigned to this command within a scoped context:

```
\let\printunsrtglossarypredoglossary
\populatemethods
```

The glossary entry handler needs to not only display the current entry (as it does by default with `\glxtrunsrtdo`) but also follow it with an inner glossary. First the custom nested handler that takes the current entry label as the argument:

```

\newcommand{\nestedhandler}[1]{%
  \glxtrunsrtdo{#1}%
  \ifglossaryexists*{#1}%
  {%
    \printunsrtinglossary
      [type={#1},groups=false,target=false]
  }%
  \let\glxtrbookindexname\linkedbookname
  \renewcommand{\GlsXtrLocationField}
    {primarylocations}%
  }{}%
}%
{}%
}

```

This tests if there's a glossary with a label matching the entry's label and, if it exists, that glossary will be displayed (but without the title). As with the ordered by toxicity glossary, only the primary location is displayed and the name is hyperlinked to easily jump back to the main definition.

All glossaries are using the `bookindex` style, and since the method and toxin entries are all top-level entries (they don't have a parent), they will all end up with their names formatted in the same way (as a top-level entry). The method entries need to have their names formatted in the same way as the group titles to be consistent with the order by toxicity glossary.

```

\newcommand{\prenamesep}{}
\newcommand{\orderbyname}[1]{%
  \prenamesep
  \glxtrbookindexbookmark{\pdfname{#1}}
    {\glxtrbookindexbookmarkprefix#1}%
  \orderbyheader{\glossentryname{#1}}%
  \def\prenamesep{\par}%
  \par\smallskip
}

```

(Again, the style is simplistic to reduce the complexity of the example.) It's then possible to switch to this in a scoped context:

```
\let\glxtrbookindexname\orderbyname
```

9 Order by symptoms

The symptoms list is more complicated as it's divided into different categories: 'vital signs', 'head, eyes, ears, nose, throat', 'skin', 'heart' and so on. These correspond to the custom `topic` field that has so far been ignored by `bib2gls`. Note that these topics aren't listed in alphabetical order. Within each topic is a sub-list of symptoms, which is ordered alphabetically. I'm first going to start off with the topics alphabetically ordered and then make an adjustment to achieve the desired result.

The topics and their sub-lists are essentially hierarchical, so the `topic` field can be aliased to

parent. The `labelify` option can strip the spaces using the same `labelify-replace` setting used earlier.

```
field-aliases={topic=parent},
labelify={parent}
```

The parent entries (representing the topics) need to be defined, so I've created a new file called `topics.bib` that contains:

```

@index{vitalsigns,name={vital signs}}
@index{head,
  name={head, eyes, ears, nose, throat}}
@index{skin}
@index{heart}
@index{airway,
  name={airway and lungs}}
@index{gastrointestinal,
  name={gastrointestinal system}}
@index{fluids,
  name={fluids and electrolytes}}
@index{neurological,
  name={neurological system}}
@index{psychiatric}
@index{wholebody,
  name={whole body and miscellaneous symptoms}}

```

This file needs to be added to the `src` list:

```
src={toxins,methods,symptoms,topics}
```

Since child entries depend on their parent, the parent entries will automatically be selected when the child entry is selected. The topics and symptoms can be copied to a new glossary in the same way as the methods:

```

\newglossary*{bysymptoms}{By Symptoms}
\GlsXtrLoadResources[
  src={symptoms,topics},
  type=bysymptoms,
  selection={selected before},
  action=copy,
  field-aliases={topic=parent},
  labelify={parent},
  labelify-replace={{\string\s+}}{}
]

```

The process hook is similar to the hook used for the method list but the throwaway glossaries are only created for child entries:

```

\newcommand{\symptomsprocesshook}[1]{%
  \ifglshasparent{#1}%
  {\provideignoredglossary{#1}}%
  }%
}

```

The code to populate the symptom glossaries is similar to that used for the methods:

```

\newcommand{\populatesymptoms}{%
  \renewcommand
  \printunsrtinglossaryentryprocesshook[1]{%
  \forglsentries[main]{\thislabel}%
  }%
}

```



```
\glxtrforcsvfield*{\thislabel}{symptom}
{\populatedo}%
}%
}
```

The entry handler is the same one as used before (`\nestedhandler`).

The formatting of the sub-items (the symptom entries) needs adjusting. The `bookindex` style formats the sub-item names according to:

```
\glxtrbookindexsubname{(label)}
```

The default definition uses `\glxtrbookindexname` so this will need to be changed. First I need a custom sub-header to match the headers used in the toxicity and method lists:

```
\newcommand{\orderbysubheader}[1]{%
\par{\raggedright\bfseries #1\par}}
```

Again this is simplistic, to be modified as required. The custom command for child entry names is:

```
\newcommand{\orderbychild}[1]{%
\pdfbookmark[2]{\pdfname{#1}}
{\glxtrbookindexbookmarkprefix#1}%
\orderbysubheader{\glossentryname{#1}}%
\par\smallskip
}
```

I added the `mfirstuc-english` package earlier, but up until now it hasn't been needed. This package can be implemented by `bib2gls` but isn't by default. In order to ensure that the title-case word exceptions provided by that package are used by `bib2gls`, it's necessary to use the `--packages` (or `-p`) switch:

```
bib2gls -g --retain-formats mainfmt \
--packages mfirstuc-english toxinbook
```

Finally, I want to have the topics listed in the order that they are defined in the `topics.bib` file. This is quite awkward as it's not possible to apply a different sort method to each hierarchical level. However, it is possible to encapsulate the sort value (after it has been obtained from fallbacks and any other processing, such as word breaks and suffixes). The encapsulation command must take two arguments: the first is the sort value that has been determined so far, and the second is the entry's label.

It's possible to save the entry definition index using `save-definition-index`. With this setting, the definition index can be accessed with the command `\bibglsdefinitionindex`. The aim here is to define a command that finds out if an entry has a parent. If it has, then the ordinary sort value is used. If it hasn't, then the definition index is used instead. Since the sort method is alphabetical, the definition index will need to be zero-padded to ensure that it's correctly ordered. Here I've padded up to six digits, which should be ample:

```
format-integer-fields={definitionindex=\%06d}
My custom command is provided in the preamble of
topics.bib:
@preamble{"\providecommand{\topicsort}[2]{%
\ifglsashasparent{#2}{#1}
{\bibglsdefinitionindex{#2}}}"}
```

This command now needs to be specified as the sort encapsulator:

```
encapsulate-sort=topicsort
```

There's no need for this preamble to be written to the `glstex` file since it's not required in the document: `write-preamble=false`

10 Duplicate entry

Stevens and Bannon's book lists 'botulism' in the 'Household Poisons' chapter and 'botulism toxin' in the 'Biological, Chemical and Radiological Weapons' chapter. The two entries have different descriptions, so they need to be defined as two separate entries, but it would be strange to have two 'botulinum' entries listed in the index.

One solution is to change the botulinum entry to a dual entry:

```
@dualentry{botulinum,name={botulinum},
toxicity = {6},
description={Some information about foodborne
botulism.},
dualdescription={Some information about the
botulinum toxin used as a bioweapon.},
method={injected and swallowed},
symptom={blurred vision, nausea and paralysis}
}
```

This defines two linked entries. The primary entry has the label `botulinum` and the dual entry has the label `dual.botulinum`. The default is to swap the name and descriptions around in the dual and copy all the other fields. In this case I want to keep the names the same but switch descriptions, which can be done with the resource option:

```
dual-entry-map={{dualdescription},{description}}
```

Since I don't need a separate list of dual entries, it's more efficient to combine the dual into the primary list:

```
dual-sort={combine}
```

I also want to move all the dual locations over to the primary entry's location list:

```
combine-dual-locations={primary}
```

This means that the original 'botulinum' entry will appear in the index with its own locations merged with the dual locations. Since the dual entry's location list ends up empty, the filter will exclude it so it won't appear in the index. The filter is also needed in the other lists. The toxicity list:

```
\printunsrtglossary*[type=bytoxicity,
                    target=false]
{%
\let\glsxtrbookindexformatheader\orderbyheader
\let\glsxtrbookindexname\linkedbookname
\let\printunsrtglossaryentryprocesshook
\filteremptylocation
\renewcommand{\GlsXtrLocationField}
{primarylocations}%
}
```

For the symptoms and method lists, the filter needs to be in the inner glossary:

```
\newcommand{\nestedhandler}[1]{%
\glsxtrunsrtdo{#1}%
\ifglossaryexists*{#1}%
{%
\printunsrtinnerglossary[type={#1},
                        groups=false,target=false]
{%
\let\glsxtrbookindexname\linkedbookname
\let\printunsrtglossaryentryprocesshook
\filteremptylocation
\renewcommand{\GlsXtrLocationField}
{primarylocations}%
}{}%
}%
}{%
}
```

Notice that the cross-reference fields only reference the primary entry, not the dual. This means that the scientific name and other names will be missing for the dual entry. However, it's possible to adjust the definition of `\toxinitelist` so that it fetches the information from the primary entry. In order to do this it's first necessary to instruct `bib2gls` to save the label of the opposite entry for dual entries. This can be done with the `dual-field` resource option, which will save the label of the opposite entry in the dual field. This means that the `dual.botulinum` entry will have the `dual` field set to `botulinum` and the `botulinum` entry will have the `dual` field set to `dual.botulinum`.

```
\newcommand{\toxinitelist}[3]{%
\glsxtrifhasfield*{#2}{#1}%
{%
\formattoxinitelist{#3}
{\glscurrentfieldvalue}%
}%
{%
\glsxtrifhasfield*{dual}{#1}%
{% is a dual entry
\glsxtrifhasfield*{#2}
{\glscurrentfieldvalue}%
}%
\formattoxinitelist{#3}
{\glscurrentfieldvalue}%
}
```

```
}%
}{%
}%
}{% not a dual entry
}%
}
```

The complete document and bib files can be downloaded [4]. I've used the `uelem` package and `hyperref`'s `hidelinks` option, so the hyperlinks show up underlined, to make them visible in the printed figures here. The key command for that:

```
\renewcommand{\glsxtrhyperlink}[2]{%
\hyperlink{#1}{\uline{#2}}}
```

The standalone entries are shown in figure 1 (household poisons — ammonia and the primary botulinum entry), figure 2 (plants — nutmeg), figure 3 (street drugs — LSD) and figure 4 (biochemical warfare — dual botulinum entry).

Figure 5 shows the toxicity list, figure 6 shows the methods list, figure 7 shows the first page of the symptoms list, and figure 8 shows the index. Finally, figure 9 shows the PDF bookmarks (in Okular).

References

- [1] F. Mittelbach. The multicol package, 2021. ctan.org/pkg/multicol.
 - [2] S. Rahtz, H. Oberdiek, The L^AT_EX3 Project. The `hyperref` package, 2021. ctan.org/pkg/hyperref.
 - [3] S. Stevens, A. Bannon. *Book of Poisons: A guide for writers*. Howdunit. Writer's Digest Books, 1st ed., 2007.
 - [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. `bib2gls`: Command line application to convert .bib files to `glossaries-extra.sty` resource files, 2021. ctan.org/pkg/bib2gls.
 - [7] N. Talbot. `bib2gls`: sorting. *TUGboat* 42(2), 2021. tug.org/TUGboat/tb42-2/tb129talbot-sorting.pdf.
 - [8] N. Talbot. The `glossaries-extra` package, 2021. ctan.org/pkg/glossaries-extra.
 - [9] N. Talbot. The `glossaries` package, 2021. ctan.org/pkg/glossaries.
 - [10] N. Talbot. The `mfirstuc` package, 2021. ctan.org/pkg/mfirstuc.
- ◇ 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>

Chapter 1

Household Poisons

Chemicals

Ammonia

Scientific Name Ammonium hydroxide.

Toxicity 4.5

Method Breathed.

Symptoms Abdominal or stomach pain, blindness, burns, collapse, coughing, flushing, mouth irritation, pain, pulmonary edema, restlessness, and tachycardia.

Description Some information about ammonia.

Food Poisoning

Botulinum

Scientific Name *Clostridium botulinum*.

Other Botox and botulism.

Toxicity 6

Method Injected and swallowed.

Symptoms Blurred or double vision, nausea, and paralysis.

Description Some information about foodborne botulism.

Figure 1: Standalone entries: Household Poisons

Chapter 2

Plants

Nutmeg

Scientific Name *Myristica argentea* (Papuan nutmeg), *Myristica fragans*, and *Myristica malabarica* (Bombay nutmeg).

Toxicity 3

Method Injected and swallowed.

Symptoms Anxiety, blurred or double vision, convulsions, dehydration, dry mouth, euphoria, fever, flushing, hallucinations, irregular heartbeat, nausea, pain, psychosis, and tachycardia.

Description Some information about nutmeg.

Figure 2: Standalone entries: Plants

Chapter 3

Street Drugs

LSD

Scientific Name Lysergic acid diethylamide.

Other Lysergide.

Toxicity 2

Method Injected and swallowed.

Symptoms Coma, confusion, convulsions, excitement, hallucinations, psychosis, and spasms.

Description Some information about LSD that includes a reference to nutmeg.

Figure 3: Standalone entries: Street Drugs

Chapter 4

Biochemical Warfare

Botulinum

Scientific Name *Clostridium botulinum*.

Other Botox and botulism.

Toxicity 6

Method Injected and swallowed.

Symptoms Blurred or double vision, nausea, and paralysis.

Description Some information about the botulinum toxin used as a bioweapon.

Figure 4: Standalone entries: Biochemical

Order by Toxicity

Toxicity Rating 6

Botulinum, 1, 7

Toxicity Rating 4

Ammonia, 1

Toxicity Rating 3

Nutmeg, 3

Toxicity Rating 2

LSD, 5

Figure 5: Order by toxicity

By Method

Breathed

Ammonia, 1

Injected

Botulinum, 1, 7
LSD, 5

Nutmeg, 3

Swallowed

Botulinum, 1, 7
LSD, 5
Nutmeg, 3

Figure 6: Order by method

By Symptoms

Vital Signs

Fever/Hyperthermia

[Nutmeg](#), 3

Tachycardia/Rapid Heartbeat or Pulse

[Ammonia](#), 1

[Nutmeg](#), 3

Head, Eyes, Ears, Nose, Throat

Blindness

[Ammonia](#), 1

Blurred or Double Vision

[Botulinum](#), 1, 7

[Nutmeg](#), 3

Dry Mouth

[Nutmeg](#), 3

Lip/Mouth Irritation

[Ammonia](#), 1

Skin

Burns

[Ammonia](#), 1

Flushing/Turning Red

[Ammonia](#), 1

[Nutmeg](#), 3

Heart

Irregular Heartbeat

[Nutmeg](#), 3

Airway and Lungs

Coughing

[Ammonia](#), 1

Pulmonary Edema

[Ammonia](#), 1

Gastrointestinal System

Abdominal or Stomach Pain

[Ammonia](#), 1

Figure 7: Order by symptom

▼ Household Poisons	1
▼ Chemicals	1
Ammonia	1
▼ Food Poisoning	1
Botulinum	1
▼ Plants	3
Nutmeg	3
▼ Street Drugs	5
LSD	5
▼ Biochemical Warfare	7
Botulinum	7
▼ Order by Toxicity	9
Toxicity Rating 6	9
Toxicity Rating 4	9
Toxicity Rating 3	9
Toxicity Rating 2	9
▼ By Method	11
Breathed	11
Injected	11
Swallowed	11
▼ By Symptoms	13
▶ Vital Signs	13
▶ Head, Eyes, Ears, Nose, Throat	13
▶ Skin	13
▶ Heart	13
▶ Airway and Lungs	13
▶ Gastrointestinal System	13
▶ Fluids and Electrolytes	13
▶ Neurological System	14
▶ Psychiatric	14
▶ Whole Body and Miscellaneous	14
▶ Index	15

Figure 9: PDF bookmarks

Index

A	
ammonia , 1	lysergic acid diethylamide , <i>see</i> LSD
ammonium hydroxide , <i>see</i> ammonia	lysergide , <i>see</i> LSD
B	
Bombay nutmeg , <i>see</i> Myristica malabarica	
botox , <i>see</i> botulinum	
botulinum , 1, 7	
botulism , <i>see</i> botulinum	
C	
Clostridium botulinum , <i>see</i> botulinum	
L	
LSD , 5	Papuan nutmeg , <i>see</i> Myristica argentea
	Myristica argentea , <i>see</i> nutmeg
	Myristica fragans , <i>see</i> nutmeg
	Myristica malabarica , <i>see</i> nutmeg
M	
	nutmeg , 3, 5
N	
P	

Figure 8: Index

Transparent file I/O using the original \TeX program and the plain \TeX format

Udo Wermuth

Abstract

Research papers demonstrate that it is possible to use a \TeX file to distribute malware to a victim's system. Although it seems that no report has been published about a virus of this kind in a real attack, the potential danger to abuse a \TeX source file to transport unfriendly code exists. This article explains an idea to make \TeX 's file I/O more transparent and develops requirements to turn the idea into \TeX macros. Their application in a \TeX file received from an untrusted source identifies all file names used for I/O operations. But the macros demand concentrated work with numerous text inputs and a non-beginner's knowledge of \TeX . Furthermore, users should be patient, curious, and courageous.

1 Introduction

The usual input to \TeX is a plain text file containing a few control sequences to instruct the program how to format the document. Through its macro capabilities \TeX allows an author to increase the number of recognized control sequences, tailoring them to the needs of the text. But \TeX does not forbid writing a macro like `"\def\useless{\useless}"` which generates an endless loop when `\useless` appears in the text. (Such endless loops are inherent for a macro expansion language [9, p.659].) Similarly, some control sequences implemented directly in the \TeX program — these are named *primitives* — must be used with care. For example, the simple `"\openout0=\jobname\bye"` truncates the file name to which `\jobname` expands, plus extension `.tex`, with zero bytes. As this is usually the file that \TeX processes as the main file in the current run its original contents are gone.

Thus it's easy to waste CPU cycles by executing `\useless`. On a modern multiuser system the single-threaded \TeX program occupies at most one CPU and a reasonably configured \TeX system doesn't require much main memory. So other users are hardly affected in their own work unless many \TeX programs run `\useless` in parallel. To produce a file that should be loaded by `\input` in a co-worker's \TeX source file with the above `\openout` statement is a bad joke and might become a disaster if there is no backup of a laboriously created main file. (To protect yourself in such a case from this bad joke set your main file temporarily to read-only, for example, under Unix-like systems with `chmod u-w`.)

These examples raise the question: how brave or careful must one be to typeset a \TeX file received from a friendly joker, a well-known silly person, an inexperienced beginner, a person known only by name, or an unknown individual who makes files available for downloading on the Internet. Is it possible that the \TeX run of this plain text file results in a damaged or, worse, virus-infected system?

Unfortunately the answer is: Be careful! A \TeX run using a specific prepared plain text file might delete important files, read private data, or infect your local system with a computer virus.

Published attacks. The thesis [13] uses \LaTeX and GNU Emacs to show in a feasibility study that a plain text file can contain code that spreads itself to other plain text files. In [1, 2] an $\varepsilon\text{-}\text{\TeX}$ source includes instructions to create during the compilation a JScript file in a certain directory. The execution of this file infects computers running MS Windows — the \TeX source contains an absolute path that's only valid for this operating system (OS).

The attacks are possible as \TeX contains commands to read from and write to any file. Some implementations of \TeX restrict which directories are permitted for \TeX 's I/O primitives. Of course, every OS should protect itself and mechanisms are usually in effect for ordinary users. But what can be done if the user runs \TeX with system administrator rights? Or when the system administrators of a multiuser system that provides a \TeX service configured the system in a way that private information is accessible to users without a need to know [12]?

I found no report of any real attack in which someone was the victim of a \TeX source file transporting a virus. This risk seems to be very small. But we can assume that some users have coded an endless loop and a few users have deleted an important file with an inappropriate file name for an `\openout`.

Is \TeX an insecure program? No, definitely not. Both published attacks need supporting tools: the programmable GNU Emacs or a JScript file placed in an auto-start directory. Similar to an email, \TeX source can be abused to transport malicious code. We avoid clicking on a link in an email sent by an unknown person and we must be cautious if we execute a \TeX file received from an untrusted source. Sure, \TeX could be more verbose with file names. But it doesn't help to learn which file was deleted and it's very cumbersome if \TeX asks every time for the user's permission to process a file, as we will see.

It's somewhat pointless to ask today why \TeX wasn't programmed with a more restricted access to files. I only provide three observations. First,

at the time \TeX was designed, this program tried to achieve new inconceivable advancements in typesetting. The limits of the available computers were touched; for example, memory had to be conserved. Second, Don Knuth’s intention, when he began the design, was to create a tool for his secretary and himself [9, p. 606; 10, p. 63]. There was no reason for mistrust, i.e., bad jokes were not expected. Third, the original \TeX was reimplemented as $\text{\TeX}82$ and at that time portability was a major concern [8, p. 254]. As file names are highly OS-dependent \TeX ’s code cannot cover all possibilities and must be carefully customized through a *change file* [8, pp. 123–124].

Implementors often transfer \TeX ’s archaic default file system into a nearly unrestricted model for the target OS. But excluding absolute paths or paths containing the short-cut for the parent directory (i.e., “./”) inhibit the attacks of [1, 2]. The recommendation of [7, §511], to use portable file names built only from letters and digits may be too restrictive, yet reminds us to think about simplicity.

Other risks. Modern \TeX implementations, not the original one that is used in this article, activate a communications mechanism to the OS; this feature uses the stream number 18 in `\write` statements. That such a communication makes the life easier for viruses and their developers or *crackers* (to name them in accordance with [15]) has been known for a long time (see [11, p. 454, no. 3]). Thus, the `\write18` feature is often disabled by default and must be explicitly switched on by the user.

A cracker might hide the use of a `\write18`. Therefore, always distrust tricky code without appropriate comments. For example, a single search for `\write18` fails with this obfuscated code; see [14].

```
\lccode‘e’=‘r’\lccode‘q’=‘w’\lccode‘r’=‘t’\lccode‘u’=‘i’
\lccode‘w’=‘e’\let\ea=\expandafter\lowercase{\ea
\global\ea\let\ea\trouble\csname qeurw\endcsn%
ame}\newcount\maker\maker=9 \multiply\maker by2
\immediate\trouble\maker{echo === GOTCHA ===}
```

All computer users know that all operating systems require regular updating to reduce the risk of a cracker getting into a system through security holes. Additional risks exist that stem from the installation of a distribution (see, for example, [16]) or that are given through the tools of the OS which are required to process a \TeX source file and \TeX ’s output; see section 10. From all I know, these risks are much higher than the danger coming from a plain text file containing \TeX commands.

Unfriendly code can lurk everywhere. Even if you compile carefully inspected source code yourself, malicious code can be present [18].

Protection by inspection. The abovementioned articles about possible attacks need several lines of \TeX code so a look at the source file might reveal the presence of instructions for a virus. But a cracker might try to hide the coded malware. Thus the \TeX files one gets from an unknown or untrusted source must either be executed in a restricted environment or be the subject of a thorough visual inspection.

A journal or proceedings editor receives numerous source files and it’s unlikely that all authors are known by the editor. On the other hand, the authors want to have their articles published and not be accused of spreading bad code. Nevertheless, an author might be a victim and unknowingly send out a \TeX file transporting code for a virus.

Although it’s a significant effort, editors should perform a visual inspection as part of the editorial work. I assume that they review text and code in most cases. Besides security, other reasons make this necessary as not all authors are willing to follow the instructions of the journal; some prefer to cheat. For example, look at the report [4] about problems with the length of submitted papers.

Protection by macros. This article describes a set of macros for the original \TeX engine with the plain \TeX format to make the file I/O operations more *transparent*. By this I mean that a user controls which files are processed when \TeX executes `\input`, `\openin`, or `\openout`. The macros don’t detect instructions for a virus or state that a file shouldn’t be processed; they only report which file names occur and give the user a chance to change them. But they accomplish more: The instrumented source file cannot stealthily bypass their reporting.

One goal of the macros is to produce an identical DVI file compared to a run without the macros if the original source is error-free. Section 3 discusses why this goal cannot be reached for all plain \TeX source files; a few eccentric constructions might fail.

Of course, the macros need a few resources. Besides memory space for the macros and other control sequences, the macro package declares five token registers. Thus, one cannot use the macros in the unlikely case that a source file requires more than 238 non-scratch token registers. Sure, the dreadful “ \TeX capacity exceeded” error message occurs earlier if the macros are used. But this is merely a theoretical problem as modern \TeX installations set \TeX ’s compile-time constants so high that it’s doubtful that an error-free source reaches \TeX ’s limits even if the macros of this article are active.

Usefulness of the macros. Above I wrote that the risk to become a victim of a virus that enters a

system via a plain \TeX file is very small. Nevertheless it might be an interesting intellectual pastime to see how to protect a system with macros against malicious code. Moreover, such macros may reassure people and increase confidence in \TeX 's security.

A cracker might be aware that these macros exist and avoid conspicuous actions if they are present. Or, say, the code contains a test so that it gets executed only on Sundays and thus a check that runs on a Thursday doesn't detect it. Clearly the macros cannot help to protect a system if they are not active during all executions of a source file.

Although I think a cracker cannot circumvent the macros if the user follows all usage instructions carefully, everyone uses the macros at one's own risk.

2 Primitives requiring file names

With the procedure `scan_file_name` [7, §526] \TeX scans in a system-independent way file names. Although file names are highly system dependent, this aspect is handled in other sections of the program. Here I use the convention that a file name consists of an optional path, the main part of the file name, and an optional extension. The path is a sequence of directories with a slash after each directory name; a period separates main part and extension. Spaces are forbidden in file names. A single period in the path, i.e., `./`, stands for the current directory, and `../` represents the parent directory.

The above-mentioned procedure is used in the implementation of four primitives: `\input` in §537, `\font` in §1257, `\openin` in §1275, and `\openout` in §1351.

The primitive `\font` is somewhat special in this list. \TeX expects a file name but replaces any extension with `tfm` (§563) as it reads for `\font` only files containing \TeX font metric (TFM) data. It checks that the contents of the file with the constructed name obey the specifications of TFM files (§562).

Although this sounds simple it might be very hard to determine which font \TeX loads. Above it was shown that the flexibility of \TeX can be abused to hide what the code will do. File names are no exception, as the following input proves.

```
\def\gobble#1{r}\lccode'z='f \lowercase{\edef
\word{zont}}\let\something\futurelet
\expandafter\expandafter\expandafter\let
\expandafter\expandafter\expandafter\futurelet
\expandafter\csname\word\endcsname\def
\lookatnext#1{\romannumeral100\romannumeral1000
\gobble\the\the\count18.\the#1}\futurelet\next
\lookatnext\linepenalty\let\futurelet\something
\lccode'z='z \show\next
```

What does `\show\next` in the last line display?

I don't see any way to abuse the primitive `\font` to read a file that isn't a TFM file.

Three main primitives. The primitives `\input`, `\openin`, and `\openout` use the complete file name that they receive. They append the extension `.tex` if \TeX doesn't find one [6, pp.25, 217, 226]. With `\input` and `\output`, \TeX prompts for a new file name if the file cannot be found or opened for writing, respectively [7, §530, §537, §1374]. The primitive `\openin` never asks the user to enter a new file name [8, p.325, no.582 of \TeX 's error log]. When \TeX asks for another file name, the good news is that it displays first “! I can't find file” or “! I can't write on file” followed by the file name that it had scanned. Thus, even if the file name was entered in an obfuscated manner now the user sees the name.

3 Expected problems

Primitives and macros behave differently in a \TeX run. If the three file I/O primitives are replaced by macros, under what circumstances does this influence the typesetting? Sure, a source file might test these command names and produce a different DVI file if one of them is a macro. In this case I only care about the result obtained with file I/O macros.

One important difference lies in the ability of macros to expand. The primitives `\openout` and `\openin` are allowed in an `\edef` (or `\xdef`, `\write`, etc.) so the macros should be accepted too. Thus the macros must either contain only expandable tokens and be quite simple or stop the expansion early.

The primitive `\input` is a special case as its acceptance in an `\edef` depends on the contents of the file that is input. \TeX usually throws an error, as it treats the end of a file that's input similar to an outer macro [6, p.206]. But \TeX accepts a file that ends with the primitive `\noexpand`. Thus, the macro `\input` must be completely expanded and do its work. But if this macro, say, sets a Boolean flag from false to true, \TeX runs into an error if `\input` is executed in an `\edef`. This is completely independent of the contents of the file that gets input.

This is expected, as `\input`'s expansion is null but \TeX starts to read from the file [6, p.214]. Thus, use of `\expandafter` will also give different results. For example, `\expandafter\show\input hello` displays “the letter H” if the file `hello.tex` contains the text “Hello TeX!”. But a macro for `\input` expands just one level and \TeX displays its first token, i.e., `\show` inactivates this token. (Our macro will start with `\begingroup`; so any control sequence between `\expandafter` and `\input` that reads at least one argument and doesn't open an unclosed

group gives an error.) Similar problems exist with the macros for `\openin` and `\openout`.

This “contents dependency” for the acceptance of the primitive `\input` makes it possible to place it between `\csname` and `\endcsname`. `TeX` allows this if the file that’s input expands to character tokens only; `\openin` and `\openout` are always rejected. For example, the statement `\csname\input hello\endcsname` is a valid construction. Usually a macro fails in this scenario if it isn’t very simple.

A similar situation occurs with the application of a prefix, `\number`, etc., to the primitive `\input`. The first token of the file that’s input must accept this command or `TeX` displays an error; `\openin` and `\openout` don’t accept such commands.

A reader might agree with me in finding some of these constructions weird and classify them as bad programming practice. Nevertheless the macros will address the four problems: the “`\csname` problem”, the “`\edef` problem”, the “`\expandafter` problem”, and the “apply problem”. Some can be solved interactively, others require a change of the source. The important point is: Be alert if a source file uses one of these unusual constructions and check the code carefully to convince yourself that it is required.

Note: The discussion concentrates on plain `TeX` but, for example, *TUGboat* uses its own macro package in which the command `\input` becomes a macro. Now, `TeX` always throws errors for the `\edef` and `\csname` problems but not for `\global` as the macro absorbs it; `\long`, `\number`, etc., give errors. Macros with at least three arguments in the `\expandafter` problem hinder `\input`.

Privacy. Let’s state it frankly: It’s not possible to hide the fact that file I/O primitives are replaced by macros. This doesn’t mean that all macros must be made public but it means that I decided not to change, for example, `\meaning`, so a cracker can look at the macro `\input`. Thus, a cracker knows which control word was given the original meaning of the primitive as it is called in the macro.

The important question is, what can a cracker do with this information? It’s suspicious to input a file without using the macro. A user sees on the terminal that `TeX` inputs a file except if `\batchmode` is active. My advice: Stop the execution if this happens without the approval through the procedure of the macro described in section 4. Thus the first statements of the macro package are

```
\let\batchmode=\scrollmode
\let\nonstopmode=\scrollmode
```

to make sure that no file can be input without a message on the terminal.

Udo Wermuth

I deactivate `\nonstopmode` too in order to assure that `TeX` stops if it cannot find a file as I decided to let `\input` scan all file names with a trick that makes `TeX` prompt for a new file name. Then the user has the chance to check which file gets processed and to change the file name if necessary or to end the run. In a second step the file name is given to the primitive whose name occurs in the source to process the file, if the run wasn’t canceled.

Another source file might redefine the primitives used in our macros and then they might not do what is intended. This problem gets solved in the usual manner: The used primitives are copied to new control words with a unique start sequence. I use the string “TRIO” for these copies and “TrIO” for all private macros. For example, instead of the primitive `\begingroup` I use `\TRIObegingroup`. The source might use the prefix TRIO too, for example,

```
\def\TRIObegingroup{% open three groups
  \begingroup \begingroup \begingroup}
```

(how likely is this?) and our own macro must get a new name, for example, `\TRIOxObegingroup`.

Security. The primitives `\openin` and `\openout` are not as verbose as `\input`. They operate on a file without stating the file name on the terminal (or in the `log` file). The control words that save the meaning of these primitives must not be made public. Otherwise an evil-doer circumvents the macros and applies the original primitives under their new name.

Fortunately, none of our public macros require the control words with the original meaning of these two primitives as `\input` is executed first. As mentioned above the file name is read with a trick to make `TeX` ask for a new file name. The user must enter a special file name that in a next step contains control words that have received via `\let` the meaning of either `\openin` or `\openout`. Therefore these control words can be given what I call a *password-protected* name.

A password-protected name contains a string of at least six letters in upper- and lowercase and with one letter from the first third of the alphabet and another from the last third. If the six letters form neither an English word nor a word in the language of the user it is very unlikely that this control word can be guessed or computed by a cracker. (Six letters define the minimum; use more if you like. Shorter passwords might be discovered with `TeX` through a brute force attack.) For example, I use in this text the name `\TRIOaAmNzZopenin` in a `\let` assignment to save the meaning of the primitive `\openin`. Note, “aAmNzZ” is a placeholder that must be changed

by the user if the macros are used. First, it's the default that a cracker knows; second, it's much too simple to make a good password.

The macros contain several passwords and some are applied more than once. For example, every used \TeX primitive has not only a copy with the prefix “TRIO” but also one with the prefix “TRIOhHJqsS” built with the password “hHJqsS” — again this is a placeholder which must be changed before the macro package is used. During the run a check procedure gets occasionally called to assure that both control words have the same meaning. At the start we define

```
\let\TRIOhHJqsSifx\TRIOifx
\let\TRIOhHJqsSelse\TRIOelse
\let\TRIOhHJqsSfi\TRIOfi
... % many more \let assignments
\def\TrIOhHJqsSstop#1{\TRIOhHJqsSerrmessage{TrIO
ALERT !!! Don't trust the source (#1)}}
\def\TrIODjQwWcheck{% check that macros are OK
\TRIOhHJqsSifx\TRIOhHJqsSifx\TRIOifx
\TRIOhHJqsSelse\TrIOhHJqsSstop{\TRIOifx}%
\TRIOhHJqsSfi % \TRIOifx is OK
\TRIOifx\TRIOhHJqsSelse\TRIOelse
\TRIOhHJqsSelse\TrIOhHJqsSstop{\TRIOelse}%
\TRIOhHJqsSfi % \TRIOelse is OK
\TRIOifx\TRIOhHJqsSfi\TRIOfi
\TRIOelse\TrIOhHJqsSstop{\TRIOfi}%
\TRIOhHJqsSfi % \TRIOfi is OK
... }% many more \ifx tests
```

An undetectable problem. As mentioned above the macros for $\backslash\text{openin}$ and $\backslash\text{openout}$ input a special file. Changes in the category codes (or *catcodes*) of used characters might change what the file shall accomplish. Thus, I decided to reset all letters and some symbols to their default catcodes before the macros of the special file are executed. This — as well as other decisions like the use of $\backslash\text{count255}$ — requires executing the code of the macros most of the time inside a group. Sure, $\backslash\text{input}$ should not load the file inside a group. But $\backslash\text{openin}$ and $\backslash\text{openout}$ act globally and can be placed inside a group.

In order to keep such changes local to the group they must not be prefixed by $\backslash\text{global}$. The problem occurs if the source sets $\backslash\text{globaldefs}=1$ because then every assignment, prefixed by $\backslash\text{global}$ or not, becomes global. Code like this is ok:

```
\begingroup\globaldefs=1 \input hello \endgroup
```

Our macro $\backslash\text{input}$ sets $\backslash\text{globaldefs}=0$, executes its code, and sets $\backslash\text{globaldefs}=1$. The first assignment to $\backslash\text{globaldefs}$ inside the macro, inside the group, is always global. Thus a problem occurs if $\backslash\text{globaldefs}$ was set to -1 before the above group as then $\backslash\text{globaldefs}$ is restored as 0 rather than -1 after $\backslash\text{endgroup}$. Similarly the code $\backslash\text{globaldefs}=1$

$\backslash\text{begingroup} \backslash\text{input} \text{hello} \backslash\text{endgroup}$ restores 0 not 1 for $\backslash\text{globaldefs}$ after $\backslash\text{endgroup}$.

A positive $\backslash\text{globaldefs}$ is rare, and when it does occur it is usually in the good case above. But the problem that arises from the two bad cases can be neither solved nor detected. The macros can only report that $\backslash\text{globaldefs}$ is positive. The user must then carefully check the source to understand why this seldom-used integer parameter was set.

4 The macro $\backslash\text{input}$

Do we need to make $\backslash\text{input}$ more transparent, as it writes the received file name to the terminal if $\backslash\text{batchmode}$ is inactive? It's easy to miss one file in a flood of output on the terminal. I prefer to check which files are input and I want to have the control to redirect the request. It is crucial for success to check which files are input. For example, a user must never allow that a source inputs any of the files of the macro package and continues the run.

The trick. How does the macro force \TeX to ask for a new file name? A nonexistent path is placed in front of the given file name. For example, I define $\backslash\text{def}\backslash\text{TrIONosubdir}\{\text{nosubdir}/\}$ where *nosubdir/* must not exist as a directory in the current directory. Next, the macro changes $\backslash\text{input} \text{hello}$ into $\backslash\text{TRIOinput}\backslash\text{TrIONosubdir} \text{hello}$.

This works fine as long as the file name doesn't start with “./” as this might undo in some \TeX implementations the “*nosubdir/*” and the remaining path points to a file in the current directory that carries the same name as the file that should be found in the parent directory. In such a case an existing file is input without asking the user. The user sees on the terminal that \TeX inputs a file without approval; stop the run and nothing dangerous can happen. Next the replacement text of $\backslash\text{TrIONosubdir}$ should be changed to, for example, two nonexistent directories “*nosubdir/nosubdir/*” before a new run is started. It is unusual for the main source to input a file from the parent directory. Be alert if this happens; stop the execution if that still happens after two nonexistent directories are used. The code tries to cope with the definition of $\backslash\text{TrIONosubdir}$.

The macro. This is the main macro:

```
\def\input{% add nonexistent subdir; raise error
\TRIObegingroup % next line works in \edef too
\TRIOafterassignment\TRIOnoexpand\TrIOempty
\TRIOdef\TrIOSkipXXXVIinsTrIOfixedef\TrIOempty
{}{\TrIOempty \TrIOhandleglobaldefs
\TRIOglobal\TRIOlet
\TrIOSkipXXXVIinsTrIOfixedef=\TRIOundefined
\TrIOcountiocmd \TrIOmessage{<<<>}}
```

```
\TrIOsetcatcodes \TrIOinput TrIOinput.tex
\TrIOinputmessage \TrIOendgroup
\TrIOinput \TrIONosubdir}% and file name: error
```

The first line (`\TrIObegingroup`) makes \TeX stop if the “apply problem” occurs or if the macro is expanded inside a `\csname/\endcsname` structure. Line 2 switches off the application of a token held by the primitive `\afterassignment`; see section 5. The tricky code works in an `\edef` too; see below. The definition of an undefined control word catches the expansion of `\input` in an `\edef`. The macro `\TrIOhandleglobaldefs` handles the `\globaldefs` problem described in the previous section. All these technical parts are discussed in a moment.

The important parts: `\TrIOcountiocmd`, catcode changes in `\TrIOsetcatcodes`, the `TrIOinput` file, `\TrIOinputmessage`, and the last line’s trick.

The first macro counts the number of times one of the three file I/O primitives is called.

```
\def\TrIOcnt{0 } \countdef\TrIOcount=255
\def\TrIOcountiocmd{% increment \TrIOcnt
\TrIOcount=\TrIOcnt \TrIOadvance\TrIOcount by 1
\TrIOxdef\TrIOcnt{\TrIONumber\TrIOcount
\TrIOspace}}
```

Together with information written to the terminal and the log file a simplified procedure for repeated execution of the source can be realized; see section 9.

```
\def\TrIOmessage{\TrIOimmediate\TrIOwrite16 }
\def\TrIOinputmessage{% what happens; what to do
\TrIOmessage{\TrIO >>> ( \TrIOcnt) Line
\TrIOthe\TrIOinputlineno: input}%
\TrIOmessage{>>> enter shown file name without
'\TrIONosubdir'.} \TrIOmessage{<<<<}}
```

The catcode changes were mentioned in section 3. The macro prepares to load `TrIOinput.tex`.

```
\def\TrIOsetcatcodes{% establish a few \catcodes
\TrIOedef\TrIONext{\TrIOthe\TrIOcatcode'\%}%
\TrIOcatcode'\%=12 \TrIOlet\%=\TrIOcatcode
%\'\=0 \%\'=12 \%\'\=12 \%\'\1=12 \%\'\2=12 }
```

These catcodes are fixed and build the base for the catcode changes in the file `TrIOinput.tex`:

```
\%\0=12 \%\3=12 ... \%\9=12 \%\a=11 \%\b=11
... \%\z=11 \%\A=11 \%\B=11 ... \%\Z=11
%\%=\TrIONext \TrIODjQwWcheck
```

Handling `\globaldefs`. The macro that checks the setting of `\globaldefs` clears it if it is positive as explained earlier. This macro de- and reactivates `\afterassignment` in case it holds a token: The macro `\TrIOSuspendafterassignment` blocks the application of this token after an assignment and the macro `\TrIOinitafterassignment` restores the default behavior. Finally, the macro defines the macro `\TrIOendgroup` that resets the integer parameter

`\globaldefs` if necessary after it closes the group opened in the first line of `\input`.

```
\def\TrIOhandleglobaldefs{% inform about
%\globaldefs>0 and switch to \globaldefs=0
\TrIOifnum\TrIOglobaldefs>0 \TrIOmessage
{TrIO Info: globaldefs is >0 (I/O)}%
\TrIOafterassignment\TrIOSuspendafterassignment
\TrIOglobaldefs=0 % only this is global
\TrIOdef\TrIOendgroup{\TrIOendgroup
\TrIOafterassignment\TrIOinitafterassignment
\TrIOglobaldefs=1 }%
\TrIOelse \TrIOSuspendafterassignment
\TrIOglobaldefs=0 \TrIOdef\TrIOendgroup{%
\TrIOendgroup\TrIOinitafterassignment}%
\TrIOfi}
```

A variant. To address some of the problems discussed in the previous section a second macro for `\input` is coded. It carries a password-protected name, `\TrIOcCkPxXinput`, to avoid its unnoticed use. It differs from the macro shown in two respects:

1. The message states “INPUT” instead of “input” to identify itself to the user.
2. In front of `\TrIOinput` in the last line the macro `\TrIOcCkPxXtransfer` appears.

The variant is called if the source file contains `\TrIOcCkPxXmove`. The user must enter this macro into the source to fix some of the discussed problems.

```
\def\TrIOcCkPxXmove#1\input{% transfer tokens
\def\TrIOcCkPxXtransfer{#1}\TrIOcCkPxXinput}
```

Use this macro only if you are convinced that a `\csname`, `\expandafter`, or “prefix” is required and the source cannot extract the password in the name.

An example. Most macros of this article are bundled in the file `TrIOmacros.tex`. This file is input in the first line of the source file that should be checked.

```
\input TrIOmacros
\batchmode \input hello \errorstopmode
\csname \input hello \endcsname
\TrIOcCkPxXmove\global\input hello
\expandafter\show\input hello
\edef\csone{\input hello } \show\csone \bye
```

When this file is executed \TeX displays the messages of the macro followed by an error:

```
<<<
(TrIOinput.tex)
TrIO >>> ( 1 ) Line 2: input
>>> enter shown file name without 'nosubdir/'.
<<<
! I can't find file 'nosubdir/hello.tex'.
1.2 \batchmode \input hello
\errorstopmode
Please type another input file name:
```

This is the normal case: First, a user should check that `TrIOinput.tex` was input, then the macro re-

ports that the first I/O command was found in line 2 and that this command is `\input`, and finally the macro displays what to do next.

We enter “hello” as the new file name. \TeX shows in the next line that it inputs `hello.tex`. But then an error message pops up.

```
(hello.tex)
! Missing \endcsname inserted.
<to be read again>
      \TRIObegingroup
\input ->\TRIObegingroup
      \TRIOafterassignment...
1.3 \csname \input
      hello \endcsname
?
This error message signals the \csname/\endcsname
problem. The answer to the question mark is to type
“42”, then to insert the correct code, i.e.,  $\text{\I\csname}$ ,
at the next prompt. Finally, enter the file name.
? 42
\input ... \TRIOendgroup
      \TRIOinput \TRIONosubdir
1.3 \csname \input
      hello \endcsname
?  $\text{\I\csname}$ 
! I can't find file 'nosubdir/hello.tex'.
1.3 \csname \input hello
      \endcsname
Please type another input file name: hello
(hello.tex)
```

In this example the `\csname` problem was fixed successfully. But, for example, the code `\csname AA \input hello \endcsname` would create a different typesetting result compared to the original source. Check carefully if the macro `\TrIOcCkPxXmove` can be inserted, if the contents of the file can be typed in, or if the source file should be rejected.

The next line represents such an insertion by the user. Now the “normal” case occurs except that the word “INPUT” signals the use of the macro. (Note the “2” as the “42” skipped the counting.)

```
<<<
(TrIOinput.tex)
TrIO >>> ( 2 ) Line 4: INPUT
>>> enter shown file name without 'nosubdir/'.
<<<
! I can't find file 'nosubdir/hello.tex'.
1.4 \TrIOcCkPxXmove\global\input hello
Please type another input file name: hello
(hello.tex)
```

without the macro \TeX reports “! You can't use a prefix with ‘\begingroup’.” and the fix is to enter “42” and “ \I\global ”. The apply problem can always be solved in this way.

Next, an error message appears as the contents of `hello.tex` doesn't start with an assignment; it's an error in the original source: “! You can't use a prefix with ‘the letter H’.”

```
After pressing RETURN  $\TeX$  displays
> \TRIObegingroup=\begingroup.
\input ->\TRIObegingroup
      \TRIOafterassignment...
1.5 \expandafter\show\input
      hello
```

which is not an error message but the result of the primitive `\show`. Nevertheless the macro `\input` lost its first token. Without intervention \TeX will display an error message as soon as it reads the corresponding `\endgroup`. This time the interactive fix is to type “41” followed by “ $\text{\I\expandafter\show}$ ”.

With a macro that reads arguments instead of the non-typesetting command `\show` such a fix is not possible. Edit the source and use `\TrIOcCkPxXmove` except in cases like `\expandafter{\input hello }`, in which the `\expandafter` should be deleted.

The macros in `TrIOfixes.tex` are designed in a way that all errors in the original source produce errors in the instrumented file, although the error messages and/or recovery might be different. An erroneous source might lead to an instrumented source in which it is impossible to recover from an error during the execution.

The last line in the above source gives an example of such an error. In the original source \TeX displays “Runaway definition?” but the instrumented source shows first “! Undefined control sequence.”

```
! Undefined control sequence.
\input ... \TRIOSkipXXXVIinsTrIOfixedef
      \TRIO...
1.6 \edef\csone{\input
      hello }\show\csone\bye
```

The name of the undefined control sequence informs the user what to do: Skip 36 tokens and insert then `\TRIOfixedef`. Doing so and after entering “hello” \TeX displays the original error message.

```
? 36
\input ... \TRIOendgroup
      \TRIOinput \TRIONosubdir
1.6 \edef\csone{\input
      hello }\show\csone\bye
?  $\text{\I\TRIOfixedef}$ 
! I can't find file 'nosubdir/hello.tex'.
1.6 \edef\csone{\input hello
      }\show\csone\bye
Please type another input file name: hello
(hello.tex)
Runaway definition?
->\TRIObegingroup \TRIOafterassignment \ETC.
! File ended while scanning definition of \csone
```

```
<inserted text>
    }
1.6 \edef\csone{\input hello
    }\show\csone\bye
```

Next TeX complains about too many closing curly braces as in the original source.

The `\show\csone` displays:

```
\TRIObegingroup \TRIOafterassignment \TrIOempty
\TRIOdef \TrIOempty {} \TRIOendgroup Hello TeX!
because of the trick in line 2 and this definition
\def\TrIOfixedef{% fix \edef problem for \input
\TRIONoexpand\TrIOempty{} \TRIOendgroup}
```

so that `\csone` contains more material than in the source file; a prefix or `\number`, etc., now gives a new error if the original accepts this in front of `\csone`.

Summary: A user can fix the apply problem interactively, but not always the `\csname` and the `\expandafter` problems; one can try to fix them in the source. The `\edef` problem must be fixed interactively but the defined macro has additional tokens.

5 Macro for `\afterassignment`

Next, let's look at the support macros that we need to handle the primitive `\afterassignment`. This primitive stores a single token that isn't expanded [6, p. 215]; thus it can hold an undefined macro and execute it after it was defined. To reproduce this behavior the macro must store the token in a token register and not via a `\let` assignment. On the other hand, a curly brace cannot be placed in a token register; this requires `\let`. To distinguish these cases the macro sets a flag. (`\afterassignment` cannot appear in a `\csname/\endcsname` construction or with a prefix like `\global`.)

```
\newif\ifTrIOSavedtoken % true: token is stored
\newif\ifTrIOblockafterassignment % true: don't
    % insert a token after an assignment
\newif\ifTrIOusetokenlist % true: use token reg
\newtoks\TrIOtoken % the token register
```

A second difficulty is that `\afterassignment` can be used in an `\edef` or `\xdef` but the macro would fail if it is fully expanded. Therefore a second token register is declared to stop the expansion.

```
\let\TRIOafterassignment=\afterassignment
\newtoks\TrIOtrafterassignment % stops expansion
% the replacement of the primitive
\def\afterassignment{% \edef expands one level
\the\TrIOtrafterassignment}
\TrIOtrafterassignment={\TRIOafterassignment}
```

For the rest of the article — and already in the code just above — I omit the initial “TRIO” if a primitive is meant and no macro replaces it. For example, above I wrote `\the` instead of `\TRIOthe`,

but I will still write `\TRIOinput` since the `\input` primitive has been replaced by a macro.

The main macro blocks the usual work of the primitive `\afterassignment` and then fetches via `\futurelet` the token that should be stored. Two of the other three user macros were shown earlier. One sets the flag to block `\afterassignment`, the second removes this block. The third uses the original primitive to call our own insertion macro.

```
\def\TRIOafterassignment{% first save a token
\begingroup\endgroup % stop \global
\TRIOglobaldefs \TrIOSavedtokentrue
\futurelet\TrIOSavedtoken\TRIOchecktoken}
% user commands for those who know the macros
\def\TRIOSuspendafterassignment{% switch off
\TRIOblockafterassignmenttrue}
\def\TRIOresumeafterassignment{% switch on
\TRIOblockafterassignmentfalse % remove block
\TRIOinitafterassignment}
\def\TRIOinitafterassignment{% init exec macro
\TRIOafterassignment\TRIOAFTERASSIGNMENT}
```

Again `\globaldefs` must be checked. This is similar to the procedure used for `\input` but here no group must be closed so `\TRIOresetglobaldefs` is defined. It's called when a token must be stored.

```
\def\TRIOglobaldefs{% inform about \globaldefs>0
% and switch to \globaldefs=0 for the macros
\ifnum\globaldefs>0 \TRIOmessage{\TRIO Info:
globaldefs is >0 (store)}%
\TRIOafterassignment\TRIOSuspendafterassignment
\globaldefs=0 \def\TRIOresetglobaldefs{%
\TRIOblockafterassignmentfalse
\TRIOafterassignment\TRIOinitafterassignment
\globaldefs=1 }%
\else\ifnum\globaldefs<0 % no group, do a reset
\TRIOafterassignment\TRIOSuspendafterassignment
\globaldefs=0 \def\TRIOresetglobaldefs{%
\TRIOblockafterassignmentfalse
\TRIOafterassignment\TRIOinitafterassignment
\globaldefs=-1 }%
\else \TRIOSuspendafterassignment % switch off
\def\TRIOresetglobaldefs{% and switch on again
\TRIOresumeafterassignment}%
\fi\fi}
```

The next macro determines the type of the token and stores it either in a token register or via a `\let` assignment.

```
\def\TRIOchecktoken{% check token, store a macro
\ifcat\noexpand\TrIOSavedtoken\relax
\let\TrIONext=\TrIOstoresavedtoken % a macro
\else % otherwise remove token from the input
\let\TrIONext=\TRIOremovesavedtoken
\fi \TrIONext}
\def\TRIOstoresavedtoken#1{% #1: cs in token reg
\let\TrIONext=\undefined \TRIOusetokenlisttrue
\TrIOtoken={#1}\TRIOresetglobaldefs}
```

```
\def\TrIOremovesavedtoken{% remove a token
\let\TrIONext=\undefined \TrIOusetokenlistfalse
\TRIOafterassignment\TrIOresetglobaldefs
\let\TrIOSavedtoken=}
```

The application macros just test the flags.

```
\def\TrIOAFTERASSIGNMENT{% use the stored token
\ifTrIOblockafterassignment% true nothing to do
\else % otherwise output token if one is saved
\ifTrIOSavedtoken \ifnum\globaldefs>0
\TrIOmessage{\TrIO Info: globaldefs is 1
  (apply)}\globaldefs=0 % clear it
\TrIOSavedtokenfalse \globaldefs=1 % & reset
\else \TrIOSavedtokenfalse \fi
\expandafter\expandafter % get rid of
\expandafter\TrIOoutputtoken % the 2 \fi
\expandafter\fi % with 3+1 \expandafter
\fi}
\def\TrIOoutputtoken{% output token (check type)
\ifTrIOusetokenlist % true: use token reg
\expandafter\the\expandafter\TrIOtoken
\else % otherwise use the saved token
\expandafter\TrIOSavedtoken
\fi}% no need to change \ifTrIOusetokenlist
```

6 Macro for `\openin`

Let's repeat what we already know about `\openin`. It's nicer than `\input` as it can't occur in a `\csname/\endcsname` construction. Moreover, it can't be prefixed by `\global` as the equals sign here does not mean an assignment is performed; it's an association between a stream number and a file name. This association acts globally so that we can execute `\openin` inside a group. To solve the `\expandafter` problem in the source just delete this token. But `\openin` might be part of an `\edef`. Thus, the technique of the previous section is applied for `\openin` too.

But `\openin` is also much more unpleasant than `\input`. It operates without stating the file name on the terminal or in the log file. Thus, the control word that saves the meaning of the primitive must not be made public. Otherwise an evil-doer could circumvent the macro and apply the original primitive under its new name. Therefore the copy of the primitive is assigned a password-protected name: `\TRIOaAmNzZopenin`.

The macro `\openin` first reads the stream number, next a test is made to see if the optional equals sign follows, and third `\TRIOinput` is called with the trick so that TeX asks for a new file name. But this time the user enters two file names. First, a generic file name—for `\openin` it's by default `openin`—and then the file name that should be processed by the primitive `\openin`. The file `openin.tex` contains several password-protected macros that do the important work. Please remember: A user must never

allow `TrIOmacros.tex` or any other file of this package, such as `openin.tex`, to be processed by the original source.

All aspects of the following macros are either well-known or have been discussed.

```
\newtoks\TrIOtropolen % token register for \edef
\def\openin{\the\TrIOtropolen}% expand one level
\TrIOtropolen={\TrIOopenin}% call the main macro
\def\TrIOopenin{\begingroup
\TrIOhandleglobaldefs \TrIOcountiocmd
\TRIOafterassignment\TrIOopenin \TrIOcount=}
\def\TrIOopenin{% remove an optional =
\TRIOafterassignment\TrIOOPENIN
\let\TrIONxt=}
\def\TrIOOPENIN{%% add nonexistent directory
\TrIOmessage{<<<}% first: the instructions
\TrIOmessage{\TrIO >>> (\TrIOcnt) Line
\the\inputlineno: openin \the\TrIOcount}%
\TrIOmessage{>>> If you accept that the file
(without \TrIONosubdir) is read}%
\TrIOmessage{>>> enter 'openin' and
follow the instructions.}\TrIOmessage{<<<}%
\ifx=\TrIONxt \def\TrIONxt{%%
\fi % otherwise \TrIONxt <> '='; so keep it
\TrIOsetcatcodes % required for openin.tex
\TRIOinput\TrIONosubdir\TrIONxt}
```

In `openin.tex`, private information is used: a kind of signature that it is the user's `openin.tex` and not one by a cracker. A user should change the text to make it unique for each installation. But of course, use only characters whose category codes are known, i.e., set in the list. As `\TrIONext` becomes undefined in the macro `\TRIOaAmNzZopenin` the message stays private.

```
\%'0=12 \%'3=12 ... \%'9=12 \%'a=11 ...
\%'z=11 \%'A=11 ... \%'Z=11 \%'>=12
\%'{=1 \%'\'=2 \%'%= \TrIONext \TrIODjQwWcheck
\TRIOgGkptpausing=1 \def\TrIONext{My message
Enter 1> return 2> file name}\TRIOgGkptpausing0
\TRIOaAmNzZopenin
```

The mentioned macro prompts for the file name and calls the password-protected primitive using the stream number stored in the register `\TrIOcount`.

```
\def\TRIOaAmNzZopenin{% get file name from user
\read16 to \FilenameOPENIN
\TRIOaAmNzZopenin\number\TrIOcount=
\FilenameOPENIN
\let\FilenameOPENIN=\undefined
\let\TrIONext=\undefined
\let\TrIONxt=\undefined
\TrIOendgroup}% see \TrIOhandleglobaldefs
```

7 Macros for `\openout` and `\immediate`

The macros to replace the primitive `\openout` are very similar to the ones used for `\openin`; and the

file `openout.tex` is similar to `openin.tex`. The only aspect not yet discussed is the “prefix” `\immediate`.

\TeX allows an `\immediate` everywhere without raising an error for the next token. This is in contrast to, for example, the prefix `\long` that, after expansion of the next token, requires a definition primitive (`\def`, etc.) or another prefix (`\global`, etc.). Although `\immediate` never complains, it influences the next token after expansion only if it is one of `\openout`, `\write`, or `\closeout`.

The way `\immediate` operates means that the macro that replaces the primitive cannot simply set a flag that signals that it was seen. For example, the sequence “`\immediate\begingroup\openout`” would then faultily apply `\immediate` to `\openout`. Can we just test if the macro `\openout` follows the macro `\immediate`? I decided to put an identification primitive at the start of `\openout` so that `\TrIOopenout` doesn’t start with `\begingroup` but with the sequence “`\TRIOimmediate\begingroup`”.

The macros for `\immediate`. As indicated above, the first part of the macros uses the known structure. Only the last line of `\TrIOimmediate` contains a new technique (or trick).

```
\newif\ifTrIOimoo % true: \immediate\openout
\newtoks\TrIOtrimmediate % token reg. for \edef
\def\immediate{\the\TrIOtrimmediate}% one level
\TrIOtrimmediate={\TrIOimmediate}% expansion
\def\TrIOimmediate{% expand the following token
\begingroup \TrIOhandleglobaldefs
\TRIOafterassignment\TrIOIMMEDIATE
\TrIOcount='x}% the trick; explanation follows
```

\TeX treats the alphabetic constant ‘x’ like a number and digests a space after such a number [7, §442]. To check if a space follows, tokens are expanded (§443) but \TeX doesn’t add anything to the alphabetic constant. Thus \TeX assigns the value 120 to `\TrIOcount` after it determines the first token of the expansion of the token that follows `\immediate`. Only if this first token is `\TRIOimmediate` does the source contain `\openout` as an interim next token for `\immediate` during the expansion.

```
\def\TrIOIMMEDIATE#1{% #1: a token; it’s tested
\ifx#1\TrIOimmediate % true: macro \openout
\global\TrIOimootrue % follows; set flag
\else \global\TrIOimoofalse \fi \TrIOendgroup
\TrIOimmediate#1}% apply the primitive
```

A cracker might set the flag (either directly or via `\TRIOimmediate` as the names aren’t protected) to confuse the user. The next `\openout` will use the flag even if no `\immediate` precedes it. Stop the execution if `TrIOMacros` reports “immediate openout” but the source file seems to have no `\immediate` in front of `\openout`. Then check the source carefully.

The macros for `\openout`. As written above, the macros for `\openout` are so similar that they aren’t shown here in detail. Besides the wording “openout” instead of “openin” and “created” instead of “read” in the messages there are two differences:

1. `\TrIOopenout` starts with `\TRIOimmediate`;
2. the first message in `\TrIOOPENOUT` contains now “`\ifTrIOimoo immediate \fi`” in front of the string “openout”.

A new password-protected macro is called in `openout.tex`; it makes use of the new flag. Otherwise `openout.tex` is identical to `openin.tex`.

```
\def\TrIObBlOyYopenout{% get file name from user
\read16 to \FilenameOPENOUT
\ifTrIOimoo \global\TrIOimoofalse
\let\TrIOnext=\TRIOimmediate % use \immediate
\else \let\TrIOnext=\relax \fi
\TrIOnext\TrIObBlOyYopenout
\number\TrIOcount=\FilenameOPENOUT
\let\FilenameOPENOUT=\undefined
\let\TrIOnext=\undefined
\let\TrIOnext=\undefined \TrIOendgroup}
```

8 The virus example

The following instructions are a modified version of the code containing the virus shown in [1] and [2]. This badly formatted, comment-free but obfuscated code should alert everyone who sees it. (I changed the original source so that it can be executed under plain \TeX . Moreover, the original file names and in one case the contents of a file were changed.)

1. `\input TrIOMacros` % new 1st line; see below
2. `\newif\ifcontinue \continuetrue`
3. `\def\uncat{\def\do##1{\c'##1=12 }\dospecials`
4. `\do\^^M\do*}\def\nice{\endlinechar=\^^M`
5. `\uncat}\def\readline#1to#2{\begingroup\nice`
6. `\global\read#1to#2\endgroup}%`
7. `{\newwrite\w\let\c\catcode\c'*13\def`
8. `*{\afterassignment\d\count255"}\def\d{%`
9. `\expandafter\c\the\count255=12}{*OD\def%`
10. `\a#1^^M{\immediate\write\w{#1}}\c'^^M5%`
11. `\newread\r\openin\r=\jobname`
12. `\immediate\openout\w=../justafile.tex`
13. `\loop\ifeof\r\continuetrue\fi\ifcontinue`
14. `\readline\r to\l\expandafter\al\repeat`
15. `\immediate\closeout`
16. `\w\closein\r}{*7E*24*25*26*7B*7D\immediate`
17. `\openout\w gotcha.tex \c'[1\c']2\c'\@`
18. `\newlinechar\^^J\endlinechar-1*5C@immediate`
19. `@write@w[What have I done?}@immediate`
20. `@closeout@w}}%`
21. `\bye`

As in the example of section 4 the file got a new first line “`\input TrIOMacros`”. Next we run \TeX on this file, which I call `danger.tex`. \TeX quickly

stops to display a message. (Some lines are broken for *TUGboat*'s column width, and the identifying password in the name for `\pausing` was deleted.)

```
<<<
TrIO >>> ( 1 ) Line 11: openin 0
>>> If you accept that the file (without
      nosubdir/) is read
>>> enter 'openin' and follow the instructions.
<<<
! I can't find file 'nosubdir/danger.tex'.
<to be read again>
      \begingroup
\TrIOImmediate ->\begingroup
      \TrIOhandlegl...
```

```
1.12 \immediate
      \openout\w=./justafilename.tex
Please type another input file name:
```

Don't get confused by the shown source lines. \TeX detects that it has the complete file name only after seeing the `\immediate` in line 12. The "TrIO >>>" line shows the number of the I/O command, the line number in which it was found, and the command itself. The first file I/O is in line 11 and the command is `\openin` with stream number 0. After the instructions \TeX displays the file name that it read plus the nonexistent subdirectory that our macros added. Here the source looks for the file `danger.tex`, i.e., itself. Although I find it weird for a file to read itself, this process is harmless compared to a file that wants to destroy itself. So I continue; that is, I enter "openin", press return, check my private message, press return, and enter the file name.

```
Please type another input file name: openin
(openin.tex
Enter 1> return 2> file name)\TRIO...pausing0=>
\FilenameOPENIN=danger
```

Next, \TeX stops again. As expected it is the second file I/O command and this time it's `\immediate \openout` with stream number 0. The source wants to write a file in the parent directory. This is very strange and shouldn't be allowed. I prefer to create a subdirectory `trioo/` and to redirect all output files to this directory. Of course, the user must remember which files are placed in this subdirectory if the source wants to read one of them again.

```
<<<
TrIO >>> ( 2 ) Line 12: immediate openout 0
>>> If you accept that the file (without
      nosubdir/) is created
>>> enter 'openout' and follow the instructions.
<<<
! I can't find file 'nosubdir../justafilename.tex'.
1.12 \immediate\openout\w=./justafilename.tex
```

```
Please type another input file name: openout
```

```
(openout.tex
Enter 1> return 2> file name)\TRIO...pausing0=>
\FilenameOPENOUT=trioo/justafilename
```

Maybe you directly saw in the source that a path contains two periods. To avoid the case that \TeX inputs an existing file `justafilename.tex` in the current directory, add in front of `\input TrIOmacros \let\twonosubdirs=y` to have `\def\TrIONosubdir{nosubdir/nosubdir/}` as explained earlier.

The third stop is similar to the second except one should check that `\immediate` occurs at the end of line 16. Again I use the output directory `trioo`.

```
<<<
TrIO >>> ( 3 ) Line 17: immediate openout 0
>>> If you accept that the file (without
      nosubdir/) is created
>>> enter 'openout' and follow the instructions.
<<<
! I can't find file 'nosubdir/gotcha.tex'.
1.17 \openout\w=gotcha.tex
```

```
      \c'[1\c']2\c'\@0
Please type another input file name: openout
(openout.tex
Enter 1> return 2> file name)\TRIO...pausing0=>
\FilenameOPENOUT=trioo/gotcha
```

At the end of the run the user should check the files in the subdirectory `trioo`. This reveals that `justafilename.tex` is a copy of `danger.tex`.

9 Repeated executions

Although the macros work well, a user needs to concentrate during the stop-and-go operation and thus it's easy to make mistakes. A run is ruined if the user enters, for example, the file name instead of `openout` at a stop for `\openout`. No harm to the system is done as \TeX reads the file; the creation of a file is only possible through the file `openout.tex`.

As soon as one manages to finish a successful run the package provides macros to avoid the input of file names in subsequent runs if the I/O commands and the file names aren't changed from run to run. These macros use the I/O commands with the file names entered in the successful run in exactly the order they occurred previously. A run is deemed successful if and only if \TeX doesn't report an error that was interactively fixed. To activate the macros for repeated executions a user has to do the following.

1. Copy the `.log` file of the successful run. For example, copy `danger.log` to `danger.trio`.
2. Run a `sed` command on the copied log file. Use `TrIOlineno.sed` (or `TrIOextract.sed`) to create another \TeX file called `TrIONames.tex`. For example, enter: `sed -f TrIOlineno.sed danger.trio > TrIONames.tex`.

3. Change the first line of the instrumented source file; replace `TrIOmacros` by `TrIOauto`.

The log file contains in the lines that start with “`TrIO >>> ...`”, “`! I can't find file ...`”, and “`FilenameOPEN...`” all the data needed to create a case statement in `TeX`, in which for each sequence number the line number, the I/O command, and the file name can be combined to do the file I/O automatically; the uppercase form of “input” is thereby changed to “`TrIOcCkPxXtransfer TrIOinput`”.

The difference between the two `sed` files is that in one the new line number and the line number of the successful run are compared. This exact replication of the successful run might be too strict if the user has to edit the text but doesn't change the sequence of I/O commands. A user can create a new `TrIONames.tex` by using `TrIOextract.sed` instead of `TrIOlineno.sed` in step 2 of the above list.

The case statement is placed in a password-protected macro stored in `TrIONames.tex`. Here is the structure of this file from the run of section 8.

```
\def\TrIOeEMnvVfilenames{% use files of prev run
\ifcase\TrIOcnt \iffalse % a technicality
\else\TrIOstop{case ( \TrIOcnt) in auto}\fi
\or\ifnum\TrIOcount=11 % case 1
\def\TrIOiocmd{\TrIOaAmNzZopenin 0}%
\TrIOenvopen \def\TrIOfile{danger}%
\TrIOmessage{TrIO >>> ( 1 ) Line 11:
openin 0 \TrIOfile}%
\else \TrIOstop{case ( \TrIOcnt) in auto}\fi
\or\ifnum\TrIOcount=12 % case 2
...
\else
\TrIOstop{unknown case ( \TrIOcnt) in auto}%
\fi \TrIOfFlouUexecute}
```

The macro `\TrIOenvopen` provides some definitions for an “environment” to end the current group for `\openin` and `\openout`. For `\input` the group must end before it gets active.

```
\def\TrIOenvopen{\let\TrIOleft=\relax
\let\TrIOright=\TrIOendgroup}
\def\TrIOenvinput{\let\TrIOleft=\TrIOendgroup
\let\TrIOright=\relax}
```

The new macros. The file `TrIOauto.tex` contains simplified macros for `\input`, `\openin`, and `\openout`. It uses the file `TrIOopen.tex` to load and write the files in `TrIONames.tex`. The new file `TrIOopen.tex` is like `openin.tex` or `openout.tex` except that it doesn't contain a personal message and that it calls `\TrIOeEMnvVfilenames`, not the password-protected copies of `\openin` or `\openout`.

The macro for `\input` no longer writes terminal messages with `\TrIOmessage`; this also applies to all other file I/O macros in `TrIOauto.tex`.

```
\def\input{\begingroup \TrIOhandleglobaldefs
\TrIOcountiocmd \TrIOsetcatcodes
\TrIOcount=\inputlineno % see \TrIOfilenames
\let\TrIONxt==% needed in \TrIOexecute
\TrIOinput TrIOopen.tex }
```

The variant with a password-protected name, `\TrIOcCkPxXinput`, isn't needed anymore because the macro `\TrIOcCkPxXmove`, which might still occur in the source, now calls `\input`.

For `\openin`, two of the four macros are unchanged. In `\TrIOopenin` the line number is saved (as in `\input`) so that it becomes available in the macro `\TrIOeEMnvVfilenames`. The other changes in this set of macros are similar to the changes seen in the new macro `\input`.

```
\def\openin{\the\TrIOtropenin}
\TrIOtropenin={\TrIOopenin}
\def\TrIOopenin{\begingroup
\TrIOhandleglobaldefs \TrIOcountiocmd
\xdef\TrIONext{\TrIOcount=\the\inputlineno}%
\TrIOafterassignment\TrIOopenIn \TrIOcount=}
\def\TrIOopenIn{\TrIOafterassignment\TrIOOPENIN
\global\let\TrIONxt=}
\def\TrIOOPENIN{\TrIONext \TrIOsetcatcodes
\TrIOinput TrIOopen.tex }
```

The macros for `\openout` and `\immediate` receive drastic changes: `\openout` becomes identical to `\openin` and `\immediate` isn't replaced by a macro.

The execution macro. The last line in the macro of `TrIONames.tex`, i.e., in `\TrIOeEMnvVfilenames`, calls a password-protected macro that executes the stored file I/O command.

```
\def\TrIOfFlouUexecute{% prepare I/O execution
\ifx=\TrIONxt \gdef\TrIONext{TrIO_}%
\else \gdef\TrIONext{TrIO_\TrIONxt}\fi
\TrIOafterassignment\TrIOfFlouUdoiocmd % exec
\font\unused=\TrIONext}% remove file name
```

The last line might be a surprise. Why do we need a `\font` command here? Now that the file name from the input isn't used for an I/O command the source contains an unread file name. I decided to read and display the file name so that a user can check that the file name agrees with the one used in `TrIONames.tex`. It's possible that a cracker codes something like “`\input\myfile`” and changes file names in `\myfile` from run to run. Although our macros use a name that was approved they can still help the user to identify such sources.

Thus the file name should be displayed. But with `\input` and the trick the user must enter another file name, for example, `null`. To reduce this to a simple `return` I apply the primitive `\font` and a prefix for the file name to avoid loading a TFM file if the file name is, for example, called `cmr10.tex`.

\TeX raises an error message that shows the file name without extension; see section 2. After a quick check that the main parts of the known file name and the shown one without `TrIO_` agree, the user continues the run by pressing return. Next the I/O command is executed; as mentioned earlier, `\input` outside the group, `\openin` and `\openout` inside the group.

```
\def\TrIOffLoudoicmd{% execute the I/O command
\let\TrIOnext=\undefined
\TrIOresumeafterassignment
\ifx\TrIOright\relax \expandafter\TrIOleft
\expandafter\TrIOicmd \expandafter\TrIOfile
\else \TrIOicmd\TrIOfile\TrIOright \fi}
```

For example, \TeX 's first message for the source `danger.tex` of section 8 with `TrIOauto.tex` is:

```
(TrIOopen.tex
TrIO >>> ( 1 ) Line 11: openin 0 danger
)
! Font \TRIOnused=TrIO_danger not loadable:
Metric (TFM) file not found.
<to be read again>
\immediate
1.12 \immediate
\openout\w=./justafile.tex
```

Although it is quite unusual the source might contain something like “`\input file.tex at`” and then \TeX interprets the “`at`” as a keyword if the input `file.tex` is treated as the name of a font. In such a case the user should change the source and place the “`at`” in curly braces; treat the keyword “`scaled`” in the same way. With `TrIOauto.tex` the repeated execution isn't a big problem.

10 Treatment of `\special`

The previous sections introduce macros that allow a user to control which external files \TeX reads and writes. But by default \TeX writes data to two other files: the log file and the DVI file.

The log file is a plain text file like the \TeX source. It is neither interpreted nor compiled.

The DVI file is a binary file that must be interpreted by a device driver. Most of its content is determined by the encoding of the text which \TeX has to typeset. But \TeX also contains the primitive `\special` that is able to write any data to the DVI file. The device drivers must know what to do with this data.

Some device drivers support a `\special` string being executed as a shell command; this scenario has the same risks as the `\write18`. Or the device driver may interpret data as PostScript instructions. PostScript code can delete files, spread a virus, or hide private data inside the PostScript file—later the author can extract this information if the user

returns its output; see [5, chap.4]. The macros of this article cannot control the actions of shell scripts or PostScript code.

It is strongly recommended to activate the security options of the device driver if a DVI file from an untrusted source is processed even if the source was compiled by oneself. For example, use `-safer` in `xdvi` [3] and `-R2` for the DVI-to-PostScript translator `dvips` [17].

Macros for `\special`. By default the macros assume that the user configures the device drivers to protect the system. That is, `TrIOmacros.tex` and `TrIOauto.tex` keep the primitive `\special` active.

But the macros offer a way to look at the data contained in a `\special` without touching the primitive. \TeX puts a marker for the `\special` and the associated token list into a so-called *whatsit* [6, p. 226] that appears in the box that \TeX ships out. \TeX writes all token lists into the log file (sometimes in an abbreviated form, see [7, §292]) with:

```
\tracingoutput=1
\showboxdepth=10000 \showboxbreadth=10000
```

The log file might now become very large! The user must search or extract the data to check what the unknown token lists contain. For example,

```
grep -e'^\.\.\.*\special' <logfile>
```

extracts the beginning of the token lists of all specials in the log file (*logfile*).

Of course, the source might set the above integer parameters to other values and we disable this by assigning `\tracinglostchars` via `\let` to the three parameters. But a source file that, for example, relies on the fact that one of the values of the three integer parameters has its default value—0, 3, or 5, respectively—might now produce unintended output. Again an unusual case; reject the source.

Besides the possibilities of keeping the primitive untouched in \TeX or tracing `\special`'s actions, the package offers to deactivate `\special` and to trace all complete token lists in the log file.

```
\def\TrIOwlog{\TRIOnimmediate\write1 }
\def\special{\TrIOwlog{<<< TrIO >>>
Line \the\inputlineno: special}\TrIOwlog}
```

A user starts the described tracing via either `\let\disablespecial=n` or `y` before reading one of `TrIOmacros.tex` or `TrIOauto.tex`, with or without executing the primitive `\special`.

11 Final remarks

The shown code snippets introduce all password-protected names, in total eight. The package consists of ten main files and to change these passwords

in all of them is therefore a laborious job. To automate this task I added two more files: a `sed` file to change the passwords and a shell script to apply the `sed` file to the ten files. Remember: It's crucial that each installation has its own passwords.

Before files of your run are returned to the author (1) delete the new first line and all inserted macros `\TrIOcCKpxXmove` in the source; (2) check the `log` file for tracing output containing password-protected macro names; (3) look at the DVI output to avoid the unlikely case that it contains information about the new macros.

I described scenarios in which the macros fail but remember these are all exotic cases—the author is playing tricks on you. That's why I wrote to inspect or reject the source file. I assume a cracker avoids these exotic cases; no one wants to attract attention to one's harmful code.

If you want to use the macros and you provide a macro package to authors think about code like

```
\let\TeX@input=\input \let\globaldefs=\undefined
\def\input{\begingroup\def\undefinedinput{}}%
\endgroup\TeX@input}
```

so that then error-free sources avoid most problems.

Can the program T_EX adopt these ideas? No. We can't deactivate `\batchmode` or stop the run to reenter a file name for `\input` without violating the TRIP test [9, p. 572]. But it's okay to exclude certain paths and to reenter names of certain files. Only when a file with such an excluded path occurs is the user asked to enter a new name or reenter the then-accepted file name that appeared in the T_EX file.

References

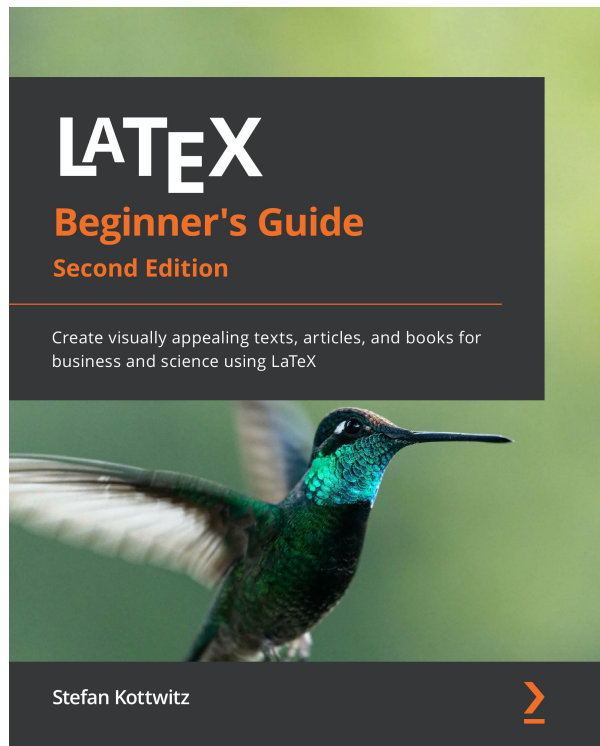
- [1] Stephen Checkoway, Hovav Shacham, and Eric Rescorla, “Are Text-Only Data Formats Safe? Or, Use This L^AT_EX Class File To Pwn Your Computer”, *Proceedings of LEET '10, USENIX* (2010), 8 pp. usenix.org/legacy/events/leet10/tech/full_papers/Checkoway.pdf
- [2] Stephen Checkoway, Hovav Shacham, and Eric Rescorla, “Don't take L^AT_EX files from strangers”, *LOGIN*: **35**:1 (2010), 17–22. usenix.org/system/files/login/articles/73506-checkoway.pdf
- [3] Eric Cooper, Bob Scheifler, Mark Eichin, Paul Vojta, et al., *Xdvi man page*, Xdvi version 22.87.04, February 29, 2020, 34 pp. tug.org/texlive/Contents/live/texmf-dist/doc/man/man1/xdvi.man1.pdf
- [4] Wouter Duivesteijn, Sibylle Hess, Xin Du, “How to cheat the page limit”, *WIRES Data Mining and Knowledge Discovery* 2020;10:e1361, 9 pp. doi.org/10.1002/widm.1361
- [5] Markus Dürmuth, *Novel Classes of Side Channels and Covert Channels*, Ph.D. thesis, Saarland University, Saarbrücken (2009), 146 pp. publikationen.sulb.uni-saarland.de/bitstream/20.500.11880/26018/1/Dissertation_1920_Duer_Mark_2009.pdf
- [6] Donald E. Knuth, *The T_EXbook*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984.
- [7] Donald E. Knuth, *T_EX: The Program*, Volume B of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986.
- [8] Donald E. Knuth, *Literate Programming*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 27, 1992.
- [9] Donald E. Knuth, *Digital Typography*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 78, 1999.
- [10] Donald E. Knuth, *Companion to the Papers of Donald Knuth*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 202, 2011.
- [11] Joachim Lammarsch, “VM/CMS site report”, *TUGboat* **11**:3 (1990), 454–455. tug.org/TUGboat/tb11-3/tb29site.pdf
- [12] Guilhem Lacombe, Kseniia Masalygina, Anass Tahiri, Carole Adam, Cédric Lauradoux, “Can You Accept L^AT_EX Files from Strangers? Ten Years Later”, arXiv:2102.00856v1 [cs.CR], 2021, 10 pp. arxiv.org/abs/2102.00856
- [13] Keith Allen McMillan, *A platform independent computer virus*, M.Sc. thesis, University of Wisconsin, Milwaukee (1994), ix+28 pp. <ftp://coast.cs.purdue.edu/pub/doc/viruses/KeithMcMillan-PlatformIndependantVirus.ps>
- [14] Scott Pakin, reply to “Malicious commands in L^AT_EX”, `comp.text.tex`, August 7, 2008. groups.google.com/g/comp.text.tex/c/epWW3eV9udw
- [15] Eric S. Raymond with Guy L. Steele Jr., eds., *The New Hacker's Dictionary*, 3rd ed., Cambridge, Massachusetts: MIT Press, 1996. catb.org/esr/jargon/
- [16] Red Hat Customer Portal: *RHSA-2012:0137 – Security Advisory*, 15 February 2012. access.redhat.com/errata/RHSA-2012:0137
- [17] Tomas Rokicki, *Dvips: A DVI-to-PostScript Translator*, version 2021.1, February 2021, 62 pp. ctan.org/pkg/dvips
- [18] Ken Thompson, “Reflections on Trusting Trust”, *CACM* **27**:8 (1984), 761–763. dl.acm.org/doi/pdf/10.1145/358198.358210

◇ Udo Wermuth
Dietzenbach, Germany
[u dot wermuth \(at\) icloud dot com](mailto:u_dot_wermuth@icloud.com)

Book review: *L^AT_EX Beginner's Guide*, second edition, by Stefan Kottwitz

Sarah Lang

Stefan Kottwitz, *L^AT_EX Beginner's Guide*, second edition. Packt, Birmingham, UK, 2021, 354 pp., softcover, US\$39.99, ISBN 9781801078658.



1 Introduction

Not unlike Boris Veytsman in his review of the 2011 version of *L^AT_EX Beginner's Guide*, I was initially skeptical.¹ Fortunately, like him, I was positively impressed by how Kottwitz's book was organized. Veytsman emphasized the hands-on approach Packt Publishers encourage ('learning by doing') and how it shuns 'boring theory', thus accommodating the impatient reader.

It has been a hallmark of L^AT_EX resources to front-load theory. This made sense in earlier days when the usual goal was to teach a select number of technology-minded students who wished to become 'superusers' eventually and already knew this from the beginning. However, as L^AT_EX is making its way into more mainstream popularity, probably fueled by an increasing number of low entrance barrier resources and, notably, the online editor Overleaf, we can always use more resources which speak to users who never intended to become 'superusers'.

¹ tug.org/TUGboat/tb32-2/tb101reviews-kottwitz.pdf

Or at least, who don't want to invest the considerable amount of time needed for a 'superuser education' in a technology to which they are not yet sure they want to commit.

Kottwitz's book is a resource for such prospective L^AT_EX users. It is an ideal crash course for self-study. This book gives them all the basics they need to get started and then make an informed decision whether they want to continue their L^AT_EX journey. For a beginner, this book answers the crucial question of how to get started using L^AT_EX and thus serves as a much-needed guide to the jungle of resources out there.

The book is organized in such a way that you are all set to get started already after the first chapter. The author's experience in writing introductory resources is obvious from how well-structured the book is for a beginner's needs. It empowers new users to solve their own problems.² Never does the book just throw a list at readers of all the possibilities L^AT_EX theoretically offers.

This lack of completeness might be considered a fault by some but it takes courage to leave out unnecessary detail. Offering truly accessible entry-level resources is no mean feat. Leaving out detail which is, for the time being, unnecessary for the beginner corresponds to the principle of didactic reduction.³ It functions somewhat like a sun visor, helping learners to focus on what's most important now, while blocking out potentially distracting information until the learner is ready to deal with it.

The chapter structure of *L^AT_EX Beginner's Guide* is driven by what users might need rather than L^AT_EX functionalities. I have described this as a 'buffet-like' approach in a blog post: Take what you need and leave what you don't.⁴ At a time where attention spans, especially for reading physical books, have dropped drastically and few people have the time or desire to sit down and deeply think about a new skill they are learning, a buffet-like approach to teaching is a blessing for the already overloaded minds of prospective new users. It is also somewhat at odds with the mindset of excellence often associated with (L^A)T_EX and its hero, Donald Knuth.

Still, I am convinced that the community should not look down upon gatekeeping-free beginners' resources. They are the heralds of a very welcome

² On empowerment: tug.org/TUGboat/tb41-2/tb128schmoelzer-empowerment.pdf

³ tug.org/TUGboat/tb41-2/tb128lang-didactic.pdf

⁴ latex-ninja.com/2021/10/27/how-to-get-started-using-latex-for-academic-writing-a-book-review-of-s-kottwitz-latex-beginners-guide-2nd-ed-packt-2021/

development: the wider adoption of L^AT_EX. Any resource that can help garner enthusiasm for L^AT_EX in a new generation of users is beneficial for all. And if the L^AT_EX community wants to stay around for future generations, there is no way other than adapting to the needs of newbies today. Kottwitz’s book contributes to that.

Here is the table of contents for the book:

- 1 *Getting Started with L^AT_EX*
- 2 *Formatting Text and Creating Macros*
- 3 *Designing Pages*
- 4 *Creating Lists*
- 5 *Including Images*
- 6 *Creating Tables*
- 7 *Using Cross-References*
- 8 *Listing Contents and References*
- 9 *Writing Math Formulas*
- 10 *Using Fonts*
- 11 *Developing Large Documents*
- 12 *Enhancing Your Documents Further*
- 13 *Troubleshooting*
- 14 *Using Online Resources*

You are truly ready to go after Chapter 2, which is just 64 pages. Chapter 2 is available as a free preview.⁵

This book is neither documentation nor reference. It is not ‘complete’ in any way. But, as I have argued before, it doesn’t have to be. Furthermore, one should judge a book not only by general standards and expectations but also by the goals it sets for itself. This book aims to be a beginner’s guide and fulfils this aim exceptionally well.

The fact that it avoids unnecessary detail also has another benefit: The more detailed a book, the faster it goes out of date. By only including the essentials, Kottwitz’s book can hopefully remain a trusted beginner’s resource for some time to come. In contrast to many other resources, it does not spend a great number of pages on installation. This also is probably a testimony of how the book is ‘modern’. Nowadays, it makes more sense to leave this amount of detailed information which easily goes out of date to the Internet. Here, users can usually find the information they need without too much hassle.

⁵ www.packtpub.com/product/latex-beginner-s-guide-second-edition/9781801078658

The reference-like style common to many other L^AT_EX resources is reminiscent of a time when it was not easy to web-search things and all the necessary information needed to be included in a book. This approach to teaching, common to computer books in general, will likely soon be a relic of the past. Kottwitz’s book is a good example for what L^AT_EX teaching can look like going forward.

Advanced users probably won’t gain much from this book but they are not the target audience. The ideal user is a L^AT_EX newbie with an interest in using L^AT_EX for academic writing. Starting out, it would have been very helpful to me to have a guide like this where I, for instance, could have looked up with ease how to change fonts if my professors were particularly picky about that. The book does not require that one immediately reads all of it. It’s perfectly sufficient to read the first chapter and then come back to the individual chapters when the need for the material presented there arises. I find that this is a practical and useful approach for beginners. If they put down the book to get in some practice after the first chapter, they have the added benefit of understanding L^AT_EX from their own experience before they dive into the following chapters. This might enhance their understanding of the more advanced topics presented there.

Kottwitz’s choice of topics which seemed remarkably ‘modern’ to the 2011 reviewer of the first edition are, of course, not all that modern any more today. I personally would even go so far as to say that the choice of ‘modern’ topics combined with the hands-on teaching approach is exactly what is needed to communicate L^AT_EX as a valuable skill to audiences who might not have previously considered themselves ‘techie enough’ for using this technology. Since this is a matter very close to my heart, I think this book is a crucial resource.

Acknowledgement: I received a free reviewer’s copy of this book in order for me to write a review about it on my blog.

◇ Sarah Lang
 Centre for Information Modelling
 University of Graz
 Elisabethstraße 59/III
 8010 Graz, Austria
[sarah dot lang \(at\) uni-graz dot at](mailto:sarah.dot.lang(at)uni-graz.dot.at)
<https://latex-ninja.com/>
 ORCID 0000-0002-4618-9481



The Treasure Chest

These are the new packages posted to CTAN (ctan.org) from October 2021–April 2022. Descriptions are based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. More information about any package can be found at ctan.org/pkg/pkgname.

A few entries which the editors subjectively believe to be especially notable are starred (*); of course, this is not intended to slight the other contributions.

We hope this column helps people access the vast amount of material available through CTAN and the distributions. See also ctan.org/topic. Comments are welcome, as always.

◇ Karl Berry
<https://tug.org/TUGboat/Chest>

biblio

- *`citation-style-language` in `biblio/bibtex/contrib`
 Bibliography formatting with the XML-based Citation Style Language.
- `ieejtran` in `biblio/bibtex/contrib`
 Unofficial BIB_TE_X style for publications of the Institute of Electrical Engineers of Japan.
- `jieeetran` in `biblio/bibtex/contrib`
 Citing Japanese articles in IEEE format.
- `pbibtex-manual` in `biblio/pbibtex`
 Documentation for Japanese (u)pBIB_TE_X.

fonts

- `andika` in `fonts`
 Fonts for beginning readers from SIL.
- `concmath-otf` in `fonts`
 OpenType math font for the Concrete design.
- `hamnosys` in `fonts`
 Font for sign languages.
- `talos` in `fonts/greek`
 Greek cult font from the 1980s.
- `vtex-nonfree` in `fonts`
 URW Classico and URW Garamond extended for Vietnamese.
- `xcharter-math` in `fonts`
 OpenType math companion for the XCharter text fonts.

fonts/utilities

- `hep-font` in `fonts/utilities`
 Latin Modern with extensions from Computer Modern and similar designs.
- `hep-math-font` in `fonts/utilities`
 Extended Greek and sans-serif math.

graphics

- `bodeplot` in `graphics/pgf/contrib`
 Making Bode, Nyquist, Nichols plots with `gnuplot` or `pgfplots`.
- `byrne` in `graphics/metapost/contrib/macros`
 Typeset geometric proofs in the style of Oliver Byrne's 1847 edition of Euclid's *Elements*, in MetaPost.
- `hexboard` in `graphics/pgf/contrib`
 Draw hex boards and games.
- `kinematikz` in `graphics/pgf/contrib`
 Design kinematic chains and mechanisms.
- `liftarm` in `graphics/pgf/contrib`
 Parameterized lift arms.
- `messagepassing` in `graphics/pgf/contrib`
 Communication protocol diagrams.
- `pgf-interference` in `graphics/pgf/contrib`
 Interference patterns.
- `pst-hsb` in `graphics/pstricks/contrib`
 Curves with continuous colors, in PSTricks.
- `robotarm` in `graphics/pgf/contrib`
 Parameterized 2D robot arms.

indexing

- `hsindex` in `indexing`
 Alternative to `xindy` and `makeindex`, in Haskell.

info

- `kaytannollista-latexia` in `info`
 Practical manual for L^AT_EX, in Finnish.
- `knuth-hint` in `info`
 (C)WEB sources from T_EX Live in HINT format. See ctan.org/pkg/hitex about the engine.
- `latex-for-undergraduates` in `info`
 Tutorial aimed at undergraduates, with an introduction to L^AT_EX Workshop in Visual Studio Code.
- `mathalphabets` in `info`
 Introduction to mathematical alphabets.

macros/latex/contrib

- `altsubsup` in `macros/latex/contrib`
 Writing sub/superscripts with square brackets and custom formatting.

[macros/latex/contrib/altsubsup](https://ctan.org/pkg/altsubsup)

- `annotate-equations` in `macros/latex/contrib`
Annotate math equations using `TikZ`.
- `atendofenv` in `macros/latex/contrib`
Add custom symbol at end of an environment.
- `bfh-ci` in `macros/latex/contrib`
Bern University of Applied Sciences design.
- `bmstu` in `macros/latex/contrib`
Bauman Moscow State Technical Univ. support.
- `ccred` in `macros/latex/contrib`
Inserting definite articles for `\cref` references.
- `clistmap` in `macros/latex/contrib`
Map and iterate over \LaTeX 3 clists.
- `codebox` in `macros/latex/contrib`
Highlighted source code in a fancy box.
- `commonunicode` in `macros/latex/contrib`
List of Unicode symbols with typeset output.
- `coop-writing` in `macros/latex/contrib`
Support for cooperative writing.
- `create-theorem` in `macros/latex/contrib`
Multilingual theorem-like environments.
- `dbshow` in `macros/latex/contrib`
Store and display data with custom filters, orders, and styles.
- `formal-grammar` in `macros/latex/contrib`
Typeset formal grammars (BNF).
- `functional` in `macros/latex/contrib`
 \LaTeX 2_ε interface for \LaTeX 3 programming.
- `grading-scheme` in `macros/latex/contrib`
Typeset grading schemes in tabular format.
- `handoutwithnotes` in `macros/latex/contrib`
Notes next to scaled slides via `pgfpages`.
- `hep-acronym` in `macros/latex/contrib`
Acronym extension for glossaries.
- `hep-bibliography` in `macros/latex/contrib`
Extend `BIB \LaTeX` with all the fields from `inspirehep.net` and more.
- `hep-float` in `macros/latex/contrib`
Convenience package for float placement.
- `hep-math` in `macros/latex/contrib`
Extended math macros.
- `hep-text` in `macros/latex/contrib`
Extensions for lists and text.
- `hep-title` in `macros/latex/contrib`
Title page extensions: preprint, affiliation, etc.
- `hvpymntex` in `macros/latex/contrib`
Automatically run `pygmentex` from `TeX` for syntax highlighting.
- `jmsdelim` in `macros/latex/contrib`
Bottom-up compositional delimiter sizing.
- `kanbun` in `macros/latex/contrib`
Typeset `kanbun-kundoku` with support for `kanbun` annotations.
- `llncs` in `macros/latex/contrib`
Document class and bibliography style for Lecture Notes in Computer Science (LNCS).
- `njustthesis` in `macros/latex/contrib`
Thesis template for Nanjing University of Science and Technology.
- `njuvisual` in `macros/latex/contrib`
Display logos related to Nanjing Univ.
- `numerica-plus` in `macros/latex/contrib`
Iterate functions, find fixed points, zeros, extremas, and more.
- `numerica-tables` in `macros/latex/contrib`
Multi-column tables of mathematical functions.
- `pascaltriangle` in `macros/latex/contrib`
Draw beautiful Pascal (Yang Hui) triangles.
- `proflycee` in `macros/latex/contrib`
Support for French high school mathematics teachers.
- `rbt-mathnotes` in `macros/latex/contrib`
Rebecca Turner's personal macros and styles for math notes.
- `seu-ml-assign` in `macros/latex/contrib`
Template for Southeast University Machine Learning assignments.
- `sillypage` in `macros/latex/contrib`
John Cleese's silly walk as page numbering style.
- `simplenodes` in `macros/latex/contrib`
Simple nodes and linking in `TikZ`.
- `snaptodo` in `macros/latex/contrib`
Put notes on closer side, and not overlapping.
- `termsim` in `macros/latex/contrib`
Simulate Windows 10, Ubuntu, and Mac terminals with various color themes.
- `unbtex` in `macros/latex/contrib`
Theses at University of Brasilia.
- `wrapfig2` in `macros/latex/contrib`
Wrap text around figures, extension of `wrapfig`.
- `yb-book` in `macros/latex/contrib`
Template for Y.B.-branded books.
- `zref-clever` in `macros/latex/contrib`
Clever \LaTeX cross-references based on `zref`.
- `zref-vario` in `macros/latex/contrib`
Combine `varioref` and `zref-clever`.
-
- m/l/c/beamer-contrib/themes**
- `beamertheme-arguelles` in `m/l/c/b-c/themes`
Emphasizing simplicity and readability.
-
- macros/latex/contrib/biblatex-contrib**
- `biblatex-readbb1` in `m/l/c/biblatex-contrib`
Process a `.bb1` file created by Biber.

macros/latex-dev/base

***latex-lab-dev** in `macros/latex-dev/base`
 L^AT_EX features in development. Currently
 includes the new command `\DocumentMetadata`.

macros/luatex/generic

luaaddplot in `macros/luatex/generic`
 Process data files as they are read by `\addplot`.

macros/luatex/latex

autopuncitems in `macros/luatex/latex`
 Automatically punctuate lists.

datestamp in `macros/luatex/latex`
 Static datestamps via `.aux` files.

letgut in `macros/luatex/latex`
 Class for *La Lettre Gutenberg*.

linebreaker in `macros/luatex/latex`
 Preventing overfull boxes by automatically
 increasing `\tolerance` and `\emergencystretch`.

luacensor in `macros/luatex/latex`
 Securely redact information using Lua.

yamlvars in `macros/luatex/latex`
 YAML parser (Lua package `tinyyaml`) and
 support functions to make L^AT_EX definitions
 using YAML.

support

luafindfont in **support**
 Lua script to search for fonts in the Lua_TE_X
 font database.

texlogfilter in **support**
 Reduce engine output or log files to warnings
 and errors.

texlogsieve in **support**
 Process log files, including merging wrapped
 lines.

systems

***hitex** in `systems/doc`
 New T_EX engine by Martin Ruckert especially
 for mobile devices.

T_EX Live 2022 news

Karl Berry

T_EX Live 2022 was released online on April 3, 2020.
 The T_EX Collection DVD is in process.

As new versions of packages are uploaded to
 CTAN, they are imported into TL, and available over
 the Internet via the `tlmgr` program. See the TL web
 site and documentation for more.

The major update in 2022 is Hi_TE_X, the new
 T_EX engine by Martin Ruckert. It generates its own
 HINT output format, intended for use on mobile
 devices. Martin has written a manual and several
 articles about the project; see ctan.org/pkg/hitex
 for links.

As always, in this year's release there are also
 pervasive updates to hundreds of packages and pro-
 grams. For a list of major changes, please see tug.org/texlive/bugs.html. For this note, I wanted
 to summarize some of the known problems in current
 TL; the same web page has more details.

Windows binaries in TL'22 are still 32-bit. We
 expect to switch to 64-bit binaries for Windows in
 2023, and we cannot provide both simultaneously—
 so be forewarned.

Lua_TE_X, unlike all other engines, does not look in a
 given `-output-directory` for input files. We expect
 this to be fixed for next year.

Some Lua-related formats are unsharable. The
 Lua_LA_TE_X, Con_TE_Xt, and Op_TE_X `.fmt` files cannot
 be shared across different architectures (32-bit/64-bit
 and/or BigEndian/LittleEndian). This is not new
 this year, but was only discovered and reported rel-
 atively recently. No decision has been made about
 changing this for Lua_LA_TE_X; Con_TE_Xt and Op_TE_X
 are not expected to ever change this, by decision of
 their authors.

On macOS Monterey 12 (the latest release at this
 writing), `install-tl` comes up as a black window,
 due to Apple's intentional breakage of the `wish` pro-
 gram (Tcl/Tk). The solution, other than installing a
 working Tcl/Tk, is to run `install-tl -gui text`,
 which is now (post-release) the default.

On Windows, `install-tl` may output the cryptic
 message `fail bad gmtime` (repeatedly). It's annoy-
 ing but harmless; just ignore it. A fix is in the works.

◇ Karl Berry
[karl \(at\) freefriends dot org](mailto:karl@freefriends.org)
<https://tug.org/texlive>

Die \TeX nische Komödie 4/2021–1/2022

Die \TeX nische Komödie is the journal of DANTE e.V., the German-language \TeX user group (dante.de). Non-technical items are omitted.

Die \TeX nische Komödie 4/2021

VOLKER RW SCHAA, Protokoll der 63. Mitgliederversammlung von DANTE am 18. September in Saarbrücken [Protocol of the 63. General Meeting of DANTE e.V. on September 18 2021 in Saarbrücken (remote)]; pp. 6–11

MARCEL KAPFER, Bericht zur Herbsttagung von DANTE e.V. 2021 in Neuland [Report of the General Meeting Autumn 2021]; pp. 11–14

This report summarizes the course of the 2021 autumn meeting of DANTE e.V. Due to the COVID-19 pandemic the meeting was completely virtual.

HERBERT VOSS, Ganz- und doppelseitige Gleitumgebungen [Full page and double page float environments]; pp. 15–46

Introduction of the `hvfloat` package that allows full page and double page float environments. Published in English, in slightly different form, in *TUGboat* 42:3.

UWE ZIEGENHAGEN, Dymo-Aufkleber mit \LaTeX gestalten [Creating Dymo labels with \LaTeX]; pp. 46–48

Short tutorial on how to create Dymo labels with the help of \LaTeX .

WALTER ENTENMANN, Einbetten von Statistik R-Code in \LaTeX [Embedding R code in \LaTeX]; pp. 48–64

Introduction of Sweave and knitR to embed R code in \LaTeX .

HENNING HRABAN RAMM, Der erweiterte Orbit [The extended orbit: News from the Con \TeX t Meeting]; pp. 65–69

Protocol of the 15. Con \TeX t meeting in Belgium.

HENNING HRABAN RAMM, Ein neuer Motor für Con \TeX t [A new engine for Con \TeX t]; pp. 69–72
Moving Con \TeX t MkIV to LMTX.

FRANK MITTELBACH, \LaTeX News, issue 33, Juni 2021 [\LaTeX News, issue 33, June 2021]; pp. 72–85

German translation of this \LaTeX news installment, published in *TUGboat* 42:2 (and on the \LaTeX web site: latex-project.org/news).

JÜRGEN FENN, Neue Pakete auf CTAN [New packages on CTAN]; pp. 86–91

An overview of new packages on CTAN.

CHRISTINE RÖMER, Newsletter zur Typografie [Newsletters on typography]; pp. 92–93

Introduction of two newsletters on typography.

Die \TeX nische Komödie 1/2022

MARTIN SIEVERS AND MATHIAS MAGDOWSKI, Einladung zur Frühjahrstagung 2022 und 64. Mitgliederversammlung von DANTE e.V. in Magdeburg [Invitation to the Spring Meeting and 64. General Meeting of DANTE e.V. in Magdeburg]; pp. 6–9

The DANTE General Meeting will take place in hybrid form (online and in person if coronavirus rules allow) from June 22nd to June 25th, 2022.

\TeX MEETING ERLANGEN, Nachruf: Walter Schmidt (1960–2021) [Obituary: Walter Schmidt (1960–2021)]; pp. 10–11

Walter Schmidt, known for his work on fonts, passed in 2021.

ROLF NIEPRASCHK, Tabellen mit dem \LaTeX -Paket `tabularray` [Tables using the \LaTeX package `tabularray`]; pp. 12–17

This article shows the usage of the `tabularray` package.

ADELHEID BONNETSMÜLLER, Having Fun with \LaTeX : Eine tolle Masche [Having Fun with \LaTeX : An amazing Mesh]; pp. 18–29

This article describes, how to typeset knitting meshes with \LaTeX .

RALF MISPELHORN, Erstellung eines Kalenders [Creating a calendar]; pp. 29–32

How to create a visually pleasing calendar with \LaTeX .

TOBIAS HILBRICHT, Lokale Seitenzähler innerhalb eines Dokuments [Local page counters within a document]; pp. 32–38

In this article we explain how to use local page counters within a document, allowing for independent page counting.

RAINER-MARIA FRITSCH, VSCodium – Eine Entwicklungsumgebung [VSCodium – A development environment]; pp. 38–49

VSCodium is a new development engine based on VSCode that is suitable for \LaTeX documents and much more.

HERBERT VOSS, Schriften für X_{\LaTeX} und $\text{Lua}\LaTeX$ [Fonts for X_{\LaTeX} and $\text{Lua}\LaTeX$]; pp. 49–57

An overview on using fonts with X_{\LaTeX} and $\text{Lua}\LaTeX$.

HENNING HRABAN RAMM, ConTeXt kurz notiert [ConTeXt News]; pp. 57–60

A short overview of what is happening in the ConTeXt world.

THOMAS A. SCHMITZ, Präsentationen in XML [Presentations with XML]; pp. 60–73

How to create a presentation using XML.

FRANK MITTELBACH, L^AT_EX News — Issue 34, November 2021; pp. 75–88

German translation of this L^AT_EX news installment, published in *TUGboat* 42:3 (and on the L^AT_EX web site: latex-project.org/news).

JÜRGEN FENN, Neue Pakete auf CTAN [New packages on CTAN]; pp. 88–92

List of new packages on CTAN.

[Received from Uwe Ziegenhagen.]

Zpravodaj 2021/1–4

Zpravodaj is the journal of $\mathcal{C}\mathcal{S}$ TUG, the T_EX user group oriented mainly but not entirely to the Czech and Slovak languages. The full issue can be downloaded at cstug.cz/bulletin.

PETR SOJKA, Úvodník [Introductory word]; pp. 1–2

Go forth and participate in $\mathcal{C}\mathcal{S}$ TUG to make the bright future of T_EX & Friends a reality! *You can!*

VÍT NOVOTNÝ, Overleaf: Kolaborativní webový editor L^AT_EXu [Overleaf: Collaborative online L^AT_EX editor]; pp. 3–8

The president of TUG named the collaborative online editor Overleaf “one of the several most important changes in the T_EX world for the last years”. In this article, I introduce Overleaf and describe its key functions and planned features.

PETR OLŠÁK, T_EX in a nutshell; pp. 9–55

Nowadays, many users discover T_EX through high-level formats that hide the complexity of typesetting behind a facade of a friendly markup language. However, all except the simplest of typesetting tasks require that users can understand what happens under the hood and know how they can influence the algorithms of T_EX when needed.

In this article, the author introduces the foundations of most high-level T_EX formats, which will help the readers with their day-to-day work with T_EX as well as their more difficult typesetting tasks.

The readers are first introduced to the program T_EX and its extensions. Then, they learn about the different processors of T_EX and their modes. Finally, the readers learn about the registers and primitive commands of T_EX as well as the macros of the plain T_EX format. The word of the day is brevity as the exposition spans less than forty pages: Excellent reading material for an otherwise uneventful train ride!

The author has previously written three books about T_EX, has developed the OpT_EX format, maintains a dozen package on the CTAN archive, and has taught a university course about T_EX for over twenty years.

DONALD KNUTH, The T_EX tuneup of 2021; pp. 56–62

Published in *TUGboat* 42:1.

BARBARA BEETON, Debugging L^AT_EX files; pp. 63–75

Published in *TUGboat* 38:2.

VÍT NOVOTNÝ, Markdown 2.10.0: L^AT_EX themes & snippets; pp. 76–82

Published in *TUGboat* 42:2.

DOMINIK REHÁK, Priama sadzba dokumentov rôznych formátov v T_EXu pomocou nástroja Pandoc [Direct typesetting of various document formats in T_EX using the Pandoc utility]; pp. 83–92

The Markdown T_EX package allows authors to typeset documents in the Markdown language and maintain control over how the documents will look. However, the package doesn’t provide support for document formats other than Markdown. In contrast, the Pandoc tool enables the conversion between dozens of document formats including T_EX and Markdown, but provides only rudimentary control over styling.

This article elaborates on the possibility of typesetting various text formats directly in T_EX by adding support for Pandoc’s intermediate document representation into the Markdown package. I focus mainly on the intermediate representations of Markdown and Pandoc as well as the differences between them, which my upcoming implementation will have to overcome. At the end of my article, I present the planned user interface for T_EX.

PETER WILSON, It might work XI; pp. 93–104

This paper shows several ways how to create a miniature book printed on just a single sheet of paper. Some L^AT_EX solutions are given.

[Received from Vít Novotný.]

TUG Annual General Meeting Procedures

1 Purpose

The Annual General Meeting (AGM) of TUG shall be conducted according to these procedures.

2 Time and place

The AGM shall ordinarily take place during the annual conference, or as specified in the TUG Bylaws, articles III.2 and III.3.

In the event that the AGM does not take place during the annual conference, the Board will either specify an alternate time and place, or give notice, with explanation, that there will be no meeting.

Notice of the meeting shall be circulated to members no less than thirty (30) calendar days prior to the meeting.

The AGM shall preferably take place in person. If necessary, the meeting will be conducted entirely online. A hybrid meeting may be conducted with both in-person and online components, if feasible.

3 Preparation for the meeting

Questions for consideration at the AGM may be submitted beforehand by sending them to the Secretary via the Board. All questions shall be acknowledged. The Secretary shall prepare responses to questions, in cooperation with the Board, to be reported (and discussed, if desired) at the meeting. Questions received less than seven (7) calendar days before the AGM may not allow sufficient time for necessary research, so answers may not be available at the AGM. In such cases, answers will be prepared after the AGM and communicated according to the followup procedure below.

Availability of President, Secretary, and Treasurer should be determined beforehand. Alternates will be designated by the President, or Vice-President in the President's absence, in consultation with the full Board.

At the meeting, all Board members attending must check in with the Secretary or designated alternate, in case a situation arises that requires Board attention or action.

4 Conduct of the meeting

Any interested party may attend the meeting.

The TUG President or designated alternate shall conduct the meeting.

The Secretary or designated alternate shall record the proceedings, with a backup audio recording made for verification if at all possible.

The length of the meeting shall be scheduled for one hour, or other period estimated to be appropriate, based on known business to be reported and questions received ahead of time.

5 Business to be transacted

The following reports shall be presented.

- A report on the status of TUG, business transacted during the year, and topics under current consideration shall be presented by the presiding officer.
- A financial report as of the most recently completed month shall be presented by the Treasurer, or in the absence of the Treasurer, by a designated alternate.
- Questions submitted to the Secretary in advance of the AGM shall be presented by the Secretary or designated alternate along with the response from the Board; discussion may follow.

If time permits, questions may be raised by attendees. When raising a question, the attendee must begin by stating their name and whether they are or are not a member. If a question can be answered, an appropriate officer (including the presiding officer) or Board member shall do so; if it cannot be answered, it shall be recorded for research and a later answer to be delivered according to the followup procedure outlined below.

A binding vote at the AGM may be held only by prior decision of the Board, published before the meeting, since the establishment of a quorum and proper credentials of AGM participants (especially at remote meetings) requires substantial effort.

6 Followup

The Board shall review promptly the recorded proceedings and confirm that they accurately represent what took place.

A report of the AGM shall appear in the next available issue of *TUGboat*.

Questions raised and not answered shall be discussed by the Board, researched if necessary, and answers to general questions reported

- supplementary to the AGM report if the answer is available in time;
- by the President or another Board member in a regular report to members either by (e)mail or in an official *TUGboat* column.

Answers to personal questions shall be communicated privately as appropriate.

(Adopted: 26 December 2021)

TUG financial statements for 2021

Karl Berry, TUG treasurer

The financial statements for 2021 have been reviewed by the TUG board but have not been audited. The totals may vary slightly due to rounding. As a US tax-exempt organization, TUG's annual information returns are publicly available on our web site: <https://tug.org/tax-exempt>.

Revenue (income) highlights

Membership dues revenue was up in 2021 compared to 2020; we ended the year with 1,210 paid members, 21 more than in 2020. The 2021 online conference had a net gain of about \$2,600, due to generous donations and few expenses. General contributions were nearly doubled to about \$21,300—thank you! Donations were the primary factor in 2021 income being up about 11%.

Other highlights; the bottom line

TUGboat production cost was up a little, due to page count and increased expenses. Members postage and delivery was down about 1/3, due to fewer special mailings. Other categories remained about the same.

Our bottom line for 2021 was positive (slightly), \$565, for the first time since 2015.

Balance sheet highlights

TUG's end-of-year asset total was steady (down less than 1%) in 2021 compared to 2020.

Committed Funds are reserved for designated projects: L^AT_EX, CTAN, MacT_EX, the T_EX development fund, and others (<https://tug.org/donate>). TUG charges no overhead to administer these funds.

The Prepaid Member Income category is member dues that were paid in earlier years for the current year (and beyond). The 2021 portion of this liability was converted into regular Membership Dues in January of 2021. The payroll liabilities are for 2021 state and federal taxes due January 15, 2022.

Upcoming

For 2022, we enabled general payments through PayPal; this is both less expensive for us and faster to process. Our previous method of paying directly through our web site is still available.

We have not changed any rates or fees for 2022, despite increased costs. Worldwide support from members and donations are what allow us to continue, so thank you! As always, we welcome ideas to attract new members.

◇ Karl Berry, TUG treasurer
<https://tug.org/tax-exempt>

TUG 12/31/2021 (vs. 2020) Revenue, Expense

	<u>Dec 31, 21</u>	<u>Dec 31, 20</u>
ORDINARY INCOME/EXPENSE		
Income		
Membership Dues	79,320	76,030
Product Sales	4,423	3,761
Contributions Income	21,311	11,830
Annual Conference	2,636	3,721
Interest Income	184	1,430
Advertising Income	565	305
Total Income	<u>108,440</u>	<u>97,078</u>
Cost of Goods Sold		
TUGboat Prod/Mailing	(22,053)	(20,312)
TUGboat Crossref	(275)	
Software Prod/Mailing	(2,391)	(2,256)
Members Postage/Delivery	(1,827)	(2,759)
Lucida Sales to B&H	(1,675)	(1,525)
Member Renewal	(372)	(356)
Total COGS	<u>(28,593)</u>	<u>(27,208)</u>
Gross Profit	79,847	69,870
Expense		
Contributions made by TUG	(2,000)	(2,000)
Office Overhead	(12,924)	(12,830)
Payroll Expense	(64,274)	(64,135)
Interest Expense	(84)	
Total Expense	<u>(79,282)</u>	<u>(78,695)</u>
Net Ordinary Income	565	(9,095)
OTHER INCOME/EXPENSE		
Prior year adjustment		1,475
NET INCOME	<u>565</u>	<u>(7,620)</u>

TUG 12/31/2021 (vs. 2020) Balance Sheet

	<u>Dec 31, 21</u>	<u>Dec 31, 20</u>
ASSETS		
Current Assets		
Total Checking/Savings	173,602	174,197
Accounts Receivable	275	395
Total Current Assets	<u>173,997</u>	<u>174,472</u>
LIABILITIES & EQUITY		
Current Liabilities		
Committed Funds	55,655	57,652
Administrative Services	1,445	1,447
Prepaid Member Income	10,075	9,185
Payroll Liabilities	1,280	1,211
Total Current Liabilities	<u>68,455</u>	<u>69,495</u>
Equity		
Unrestricted	104,977	112,596
Net Income	565	(7,620)
Total Equity	<u>105,542</u>	<u>104,977</u>
TOTAL LIABILITIES & EQUITY	<u>173,997</u>	<u>174,472</u>

T_EX 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](mailto:typesetting@dangerouscurve.org)
Typesetting for over 40 years, we have experience in production typography, graphic design, font design, and computer science, to name a few things. One of us co-authored, designed, and illustrated a T_EX book (*T_EX for the Impatient*).

We can:

- convert your documents to L^AT_EX 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 T_EX-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](mailto:amyh@texnology.com)
Web: www.texnology.com

Full time L^AT_EX 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 L^AT_EX implementation for e-publishing or print books and journals, or specialized projects.
- Data visualization, database publishing.
- L^AT_EX 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](mailto:info@typotexnica.it)
Web: www.typotexnica.it

Our skills: layout of books, journals, articles; creation of L^AT_EX 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](mailto:david.latchman@texnical-designs.com)
Web: www.texnical-designs.com

L^AT_EX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized L^AT_EX packages and classes to meet your needs. Contact us to discuss your project or visit the website for further details.

L^AT_EX Typesetting

Email: [enquiries \(at\) latextypesetting.com](mailto:enquiries@latextypesetting.com)
Web: latextypesetting.com

L^AT_EX Typesetting has been in business since 2013 and is run by Vel, the developer behind LaTeXTemplates.com. The primary focus of the service is on creating high quality L^AT_EX templates and typesetting for business purposes, but individual clients are welcome too.

I pride myself on a strong attention to detail, friendly communication, high code quality with extensive commenting and an understanding of your business needs. I can also help you with automated document production using L^AT_EX. I'm a scientist, designer and software developer, so no matter your field, I've got you covered.

I invite you to review the extensive collection of past work at the Showcase latextypesetting.com/showcase. Submit an enquiry for a free quote!

Monsurate, Rajiv

Web: www.rajivmonsurate.com
latexwithstyle.com

I offer: design of books and journals for print and online layouts with L^AT_EX and CSS; production of books and journals for any layout with publish-ready PDF, HTML and XML from L^AT_EX (bypassing any publishers' processes); custom development of L^AT_EX packages with documentation; copyediting and proofreading for English; training in L^AT_EX for authors, publishers and typesetters.

I have over two decades of experience in academic publishing, helping authors, publishers and typesetters use L^AT_EX. I've built typesetting and conversion systems with L^AT_EX and provided T_EX support for a major publisher.

Sofka, Michael

Email: [michael.sofka \(at\) gmail.com](mailto:michael.sofka@gmail.com)

Professional T_EX and L^AT_EX consulting and programming services. I offer 30 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in T_EX and L^AT_EX: Automated document conversion; Programming in Perl, Python, C, R and other languages; Writing and customizing macro packages in T_EX or L^AT_EX, `knitr`.

If you have a specialized T_EX or L^AT_EX need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

Veytsman, Boris

132 Warbler Ln.
Brisbane, CA 94005
+1 703-915-2406
Email: [borisv \(at\) lk.net](mailto:borisv@lk.net)
Web: www.borisv.lk.net

T_EX and L^AT_EX consulting, training, typesetting and seminars. Integration with databases, automated document preparation, custom L^AT_EX packages, conversions (Word, OpenOffice etc.) and much more.

I have about two decades of experience in T_EX and three decades of experience in teaching & training. I have authored more than forty packages on CTAN as well as Perl packages on CPAN and R packages on CRAN, published papers in T_EX-related journals, and conducted several workshops on T_EX and related subjects. Among my customers have been Google, US Treasury, FAO UN, Israel Journal of Mathematics, Annals of Mathematics, Res Philosophica, Philosophers' Imprint, No Starch Press, US Army Corps of Engineers, ACM, and many others.

We recently expanded our staff and operations to provide copy-editing, cleaning and troubleshooting of

T_EX manuscripts as well as typesetting of books, papers & journals, including multilingual copy with non-Latin scripts, and more.

Warde, Jake

90 Resaca Ave.
Box 452
Forest Knolls, CA 94933
+1 650-468-1393
Email: [jwarde \(at\) wardepub.com](mailto:jwarde@wardepub.com)
Web: myprojectnotebook.com


I have been in academic publishing for 30+ years. I was a Linguistics major at Stanford in the mid-1970s, then started a publishing career. I knew about T_EX from editors at Addison-Wesley who were using it to publish beautifully set math and computer science books.

Long story short, I started using L^AT_EX for exploratory projects (see website above). I have completed typesetting projects for several journal articles. I have also explored the use of multiple languages in documents extensively. I have a strong developmental editing background in STEM subjects. If you need assistance getting your manuscript set in T_EX I can help. And if I cannot help I'll let you know right away.

Hello from T_EXnology!

Macro writing, Design, Data Visualization, E-Publishing, Innovations, and more

We've been writing macro files and teaching L^AT_EX for more than 30 years.

Now offering L^AT_EX 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 L^AT_EX projects, with perhaps some ideas for projects you'd like to develop:

WWW.T_EXNOLOGY.COM

Calendar

2022

- | | |
|---|---|
| <p>Jun 4 Lecture: Confessions of a Type Designer, Donald Tarallo, Museum of Printing, Haverhill, Massachusetts.
museumofprinting.org/calendar</p> <p>Jun 8–10 Grapholinguistics in the 21st century—From graphemes to knowledge, Paliseau, France.
grafematik2022.sciencesconf.org</p> <p>Jun 8–
Jul 13 TypeParis22, intensive type design program, Paris, France. typeparis.com</p> <p>Jun 15 TUG 2022, deadline for presentation proposals</p> <p>Jun 20–22 International Society for the History and Theory of Intellectual Property (ISHTIP), 13th Annual Workshop, “Machines of Law and Intellectual Property as Legal Machinery”, University of Gothenburg, Sweden.
www.ishtip.org/?p=1210</p> <p>Jun 20–22 Twentieth International Conference on New Directions in the Humanities, “Data, Media, Knowledge: Re-Considering Interdisciplinarity and the Digital Humanities”, University of the Aegean, Rhodes, Greece.
thehumanities.com/2022-conference</p> <p>Jun 23–25 DANTE 2022 Sommertagung and 64th meeting (hybrid), Otto-von-Guericke Universität, Magdeburg, Germany.
dante.de/veranstaltungen/dante2022</p> | <p>Jul 10 TUG 2022, deadline for preprints for program</p> <p>Jul 11–15 SHARP 2022, “Power of the written word”, Society for the History of Authorship, Reading & Publishing, University of Amsterdam, The Netherlands
sharpweb.org/main/conferences</p> <hr/> <p>TUG 2022 online
Presentations covering the T_EX world</p> <p>Jul 22–24 The 43rd annual meeting of the T_EX Users Group.
tug.org/tug2022</p> <hr/> <p>Jul 25–29 Digital Humanities 2022, Alliance of Digital Humanities Organizations, “Responding to Asian Diversity”, Tokyo, Japan, and Online.
dh2022.adho.org</p> <p>Jul 31 <i>TUGboat</i> 43:2 (proceedings issue), submission deadline.</p> <p>Aug 8–11 SIGGRAPH 2022, Vancouver, Canada.
s2022.siggraph.org</p> <p>Sep 12–18 16th International ConT_EXt Meeting, Dreifelden, Germany.
meeting.contextgarden.net/2022</p> <p>Oct 15 <i>TUGboat</i> 42:3, submission deadline.</p> |
|---|---|

Owing to the COVID-19 pandemic, schedules may change. Check the websites for details.

Status as of 15 April 2022

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568, email: office@tug.org). 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.

Introductory

- 7 Jacques André, Patrick Bideault, Denis Bitouzé, Michel Bovani, Thierry Bouche, Maxime Chupin, Daniel Flipo, Yvon Henel / The last decade at GUTenberg
 - response to article by Just; background and activities of GUTenberg
- 4 Barbara Beeton / Editorial comments
 - typography and TUGboat news
- 6 Jonathan Fine / Robin Fairbairns and UK TUG
 - personal remembrance of Robin
- 3 Boris Veytsman / From the president
 - on the use of inventions for war and peace

Intermediate

- 23 Seth Bergmann / Making open source textbooks, and diagrams with AIDraTex
 - collaborative textbooks and graphics with [A1]DraTex.sty
- 75 Karl Berry / The treasure chest
 - new CTAN packages, October 2021–April 2022
- 10 Vít Novotný, Dominik Reháč, Michal Hoftich, Tereza Vrabcová / Markdown 2.15.0: What's new?
 - new features both for Markdown writers and coders of templates and solutions

Intermediate Plus

- 16 Carla Maggi / The DuckBoat — Beginners' Pond: CDs, but not Compact Disks
 - making commutative diagrams with the tikz-cd package
- 40 Joseph Wright / 13build: The beginner's guide
 - testing, preparing, installing, and uploading releases of packages

Advanced

- 28 Max Chernoff / Automatically removing widows and orphans with lua-widow-control
 - automated removal of widow and orphan lines, without stretching, in all Lua formats
- 44 Nicola Talbot / bib2gls: standalone entries and repeated lists (a little book of poisons)
 - glossary definitions in documents; reordering glossary entries in multiple ways
- 59 Udo Wermuth / Transparent file I/O using the original T_EX program and the plain T_EX format
 - require user confirmation on `\input` and other operations

Reports and notices

- 2 Institutional members
- 73 Sarah Lang / Book reviews: *L^AT_EX Beginner's Guide*, second edition, by Stefan Kottwitz
 - review of this introduction for today's new L^AT_EX users
- 77 Karl Berry / T_EX Live 2022 news
 - brief summary of known issues in the TL'22 release
- 78 From other T_EX journals: *Die T_EXnische Komödie* 4/2021–1/2022; *Zpravodaj* 2021/1–4
- 80 TUG Bylaws committee / TUG Annual General Meeting Procedures
- 81 Karl Berry / TUG financial statements for 2021
- 82 T_EX consulting and production services
- 83 T_EXnology Inc.
- 84 Calendar