TUGboat

Volume 38, Number 1 / 2017

TEX Users Group

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

Individual memberships

2017 dues for individual members are as follows:

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

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

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

Members also have the option to receive *TUGboat* and other benefits electronically, for an additional discount.

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

Journal subscriptions

TUGboat subscriptions (non-voting) are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate for 2017 is $110.

Institutional memberships

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

Trademarks

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

[printing date: March 2017]

Printed in U.S.A.

Addresses

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

Telephone

+1 503 223-9994

Fax

+1 815 301-3568

Web

`http://tug.org/`
`http://tug.org/TUGboat/`

Electronic Mail

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

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

Technical support for TEX users:
`support@tug.org`

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

. . . I sat down to dinner to find my soup which was a sort of noodle soup, turning into all sorts of curious forms and even letters of the alphabet. . . . On close investigation I found that the dough had been run through forms so as to make the different letters of the alphabet and figures, too! It was like looking in the "hell box" of a printing office, ...

Wilbur Wright, letter to sister Katherine,
Le Mans, June 28, 1908
in David McCullough,
The Wright Brothers (2015)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 38, NUMBER 1 • 2017
PORTLAND • OREGON • U.S.A.

TUGboat editorial information
This regular issue (Vol. 38, No. 1) is the first issue of the 2017 volume year.

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

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

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

The second 2017 issue will be the proceedings of the TUG'17 conference (`tug.org/tug2017`); the deadline for receipt of final papers is May 12. The third issue deadline is September 1.

The *TUGboat* style files, for use with `plain` TEX and LATEX, are available from CTAN and the *TUGboat* web site, and are included in common TEX distributions. We also accept submissions using ConTEXt. Deadlines, templates, tips for authors, and more is available at `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 TEX 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`.

TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:
`http://tug.org/instmem.html`
Thanks to all for their support!

American Mathematical Society, *Providence*, *Rhode Island*

Aware Software, Inc., *Midland Park*, *New Jersey*

Center for Computing Sciences, *Bowie*, *Maryland*

CSTUG, *Praha*, *Czech Republic*

Fermilab, *Batavia*, *Illinois*

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

Maluhy & Co., *São Paulo*, *Brazil*

Marquette University, *Milwaukee*, *Wisconsin*

Masaryk University, Faculty of Informatics, *Brno*, *Czech Republic*

MOSEK ApS, *Copenhagen*, *Denmark*

New York University, Academic Computing Facility, *New York*, *New York*

Overleaf, *London*, *UK*

River Valley Technologies, *Trivandrum*, *India*

ShareLaTeX, *United Kingdom*

Springer-Verlag Heidelberg, *Heidelberg*, *Germany*

StackExchange, *New York City*, *New York*

Stanford University, Computer Science Department, *Stanford*, *California*

Stockholm University, Department of Mathematics, *Stockholm*, *Sweden*

TNQ, *Chennai*, *India*

University College, Cork, Computer Centre, *Cork*, *Ireland*

Université Laval, *Ste-Foy*, *Québec*, *Canada*

University of Cambridge, Centre for Mathematical Sciences, *Cambridge*, *United Kingdom*

University of Ontario, Institute of Technology, *Oshawa*, *Ontario*, *Canada*

University of Oslo, Institute of Informatics, *Blindern*, *Oslo*, *Norway*

President's note

Jim Hefferon

This is my last column as President. I am delighted to announce our next President and a robust election for the next Board. I will discuss some more events in the TEX world, and I will also take a minute for an exit reflection.

Elections

This is an election year for TUG. Since there was a single candidate for President, Boris Veytsman, he will serve for the next two years. Congratulations Boris!

For the Board there are ten open seats. To every one of the candidates, let me say on behalf of the community, "Thank you for volunteering." By the time this reaches you the results will be posted at `tug.org/election/`.

Conferences

This year's TUG conference takes place in conjunction with BachoTEX, April 29–May 3, 2017. We are joining the Polish group GUST in celebrating their 25th birthday. See `tug.org/tug2017/` for more information. If you are planning to go please read "Things you always wanted to know about Bachotek" at `tug.org/tug2017/bachotex.html`.

The 11th International ConTEXt meeting will be September 11–17, 2017 in Butzbach-Maibach, Germany. See `meeting.contextgarden.net/2017/` for the full information.

One more meeting note: we've submitted an application to be a satellite conference for the 2018 International Conference of Mathematicians in Rio de Janeiro. Nothing is firm yet but it is an exciting prospect. Fingers crossed.

On the TUG site

Dave Walden has returned to doing interviews and the latest subject is Scott Pakin. I am a big admirer of Scott's work, including the Comprehensive LATEX Symbol List, and I often recommend it when answering questions. But his Visual LATEX FAQ made a bigger impression on me because the first time I saw it, I was floored. This idea is so compelling that of course the document needs to be made, but how come I never thought of anything like it? A brilliant idea just seems obvious in hindsight. As usual, Dave does a great job with an interesting subject. See `tug.org/interviews/pakin.html`.

There are also new book reviews. Boris Veytsman looks at *The Noblest Roman: A History of the Centaur Types of Bruce Rogers* by Jerry Kelly and Misha Beletsky; Dave Walden reviewed *Track Changes: A Literary History of Word Processing* by Matthew G. Kirschenbaum; and I have looked at the latest edition of George Grätzer's *More Math Into LATEX*. The link `tug.org/books/` will take you to all the book reviews, discounts, and more.

Membership

The winner of the prize for the 2016 *Members Bring Members* drive is Doug Marmion. He received a limited edition of *Manuale Zapficum, 2009: Typographic arrangements of the words by and about the work of Hermann Zapf & Gudrun Zapf von Hesse* from RIT Press. Congratulations to Doug, and thank you to everyone who participated (and another thank you to Boris for taking care of the prize and the project administration).

Membership remains a concern. If you know someone who uses TEX and friends then urge them to consider joining TUG. In particular, point out to them how very inexpensive it is to support their tools. The basic membership early bird rate is $85, with a $40 deduction if you opt out of a paper copy of *TUGboat* and the software DVD and get them in electronic form only. If you are a student, a new grad, a senior, or a citizen of a country with a modest economy then after the deduction the annual membership is only $15. It could not be more reasonable. See `tug.org/join.html`.

Reflections

Although I am not a tumultuous person, it is fair to say that my brief time in office has been a tumultuous one for TUG. At times it seemed that the organization could collapse. It is a lesson in how fragile things are, that being good today is not enough and we must work hard to keep it up.

But we came through, with the support of members and with the pitching-in of everyone on the Board. I must especially note the efforts of Pavneet Arora, Barbara Beeton, Michael Doob, and Boris Veytsman. These folks have been an inspiration.

We are all glad to see TUG back accomplishing things in our community. Last summer's conference was delightful. We have, with our friends in GUST, every prospect of another wonderful meeting in a few months. And, we are optimistic about our chances for a prestigious conference venue in 2018.

Positive things are of course also happening outside of TUG. I am particularly cheered by the emergence of online sites for using TEX and LATEX. As a college teacher, I would like to see more undergraduates turned on to TEX and these sites significantly lower the barrier to entry.

Finally, to finish my time writing this column, I ask your indulgence to state a personal opinion.

All is not roses. TUG's slow but steady loss of members is a worry. Others may not agree but I perceive that it is in part a reflection of a larger problem, decreased use. For instance, among colleagues at my school TEX is no longer the go-to, even for semi-formal mathematical documents such as course exams. Part of this decrease comes from users wanting things that they have trouble getting with TEX and so they shift to reaching first for other tools.

Everyone here knows that the TEX family programs set a standard. They have made it possible for users who have not spent years being trained in printing, but instead are simply reasonably competent with the programs, to produce documents of professional quality, especially for mathematical text. And they have done it while maintaining the advantages of being markup-based and of being Free software. TEX continues in many ways to set the standard; the work of Knuth, and of others in our community, remains a milestone accomplishment.

But in acknowledging the milestone we must also acknowledge that the work continues. One of TUG's goals[1] is "To foster innovation in high-quality electronic document preparation." And users want innovation. We have had our idea of documents widened, in many cases by what browsers give us, or even word processors. We want a variety of non-text elements, including video and audio. We also want to simply include many more of them, so that the model of floating occasional figures is stretched beyond its usefulness. We want reflowing text, including high quality line breaking, even if our content is mathematics.

We also want documents that are active. I am often asked about including interactive graphics such as zoom-in illustrations in Calculus, in both 2D and 3D. A colleague asked about a textbook for a topic that requires the most current information, mathematics for social justice, so that the document must update itself from the web with the latest statistics, including changing its graphs to reflect those facts, not just at compile time but at run time. And, I am writing a document on computability in which I want to embed a Scheme interpreter, so that students can run code.

Here is an example that is relatively mainstream. My department teaches from a statistics text where doing the homework requires that students leave the text for a JavaScript-heavy web site, and then return to the book for the next exercise. It is an awkward pairing. Why are we going on the web? Because that's the only way to do the needed activity (for instance, to have the computer simulate a thousand samples, each consisting of flipping a coin a hundred times). Then why are we going back to a book? Because web pages in browsers do not succeed as texts, at least as they are today constituted, at least in part because they are not typographically good enough. Despite the advantages of web pages — hyperlinks, searchable text, etc. — the elements of composition that have developed over centuries, the elements that TEX does so well, are critical to making the text usable, to making the material easily comprehensible, and current pages and browsers do not do that well enough.

Do TEX users have a way to do best-of-both? In the PDF viewer category, the standard is Acrobat Reader. It has high quality output and promises an embedded JavaScript interpreter and reflowing text, but it doesn't do that across platforms. Among active formats, besides a browser the leading freely-usable one is Jupyter notebooks. They are very interesting but have no claims to first-class typography, to being able to produce professional-quality output.

Now, readers of *TUGboat* know that a great deal of very good work is being done on many of the issues I've named. I'll mention the LuaTEX and LATEX3 efforts as admirable examples and there are others. Indeed, some of the things I've mentioned are possible today, but often even those can be done only in ways that are ungainly, that require knowledge of a variety of technologies, that do not work across all major platforms, or that are a hack that we cannot reasonably expect to still work in a decade.

I don't have a solution. But I do have this column, this one last time, and I will use it to make a request: the next time that you see a document on the web that wows you, ask yourself whether this could be done today with the TEX family, by a reasonably capable user. To the extent that the answer is "yes" we as a community are doing great.

But where the answer is "no", I hope that we will see that as a challenge, and will work together to *make* them do-able. Then TEX will retain its place as a leading tool for creating the best documents. The work that we've done will continue to move forward.

And I shall enjoy that, from the relative calm of being an ex-President.

◇ Jim Hefferon
Saint Michael's College
jhefferon (at) smcvt dot edu

[1] tug.org/aims_ben.html

Editorial comments

Barbara Beeton

iTEX lives!

(ding-ding!)[1] A communication from Don Knuth reports that his sister gave him "a curious Christmas present". Gift Republic Ltd. has a gimmick to offer a previously nameless star to customers, and allow the customer to give it an unofficial name. According to the promo lit,

> The chosen star name will not necessarily be recognised by any scientific organisation or by astronomers and Gift Republic can accept no liability for this. Star names are published in the Gift Republic Star Registry which is stored in a secure location. A record of the Registry is periodically submitted to the British Library to ensure public accessibility and preservation.

The star chosen for Don has the AGASC_ID[2] of 29897992; its coordinates are 9h 23m 46.4s, 1° 12′ 49.6″.

> That (appropriately) puts it in the constellation Hydra, near the 'head' end — roughly midway between the serpent's head and the constellation Sextans.
>
> Its magnitude is 12.9. (That's pretty dim, but it surely would be a lot brighter if we could get closer.)
>
> I have no idea of the estimated distance from Earth to iTeX. But it's probably pretty large, and that might explain why I haven't seen any implementations yet.

Are there any astronomers in the audience? If so, we invite confirmation of the name.

Don concluded his message with this postscript: "Maybe there's even a computer scientist in iTeX's solar system who has given the name 'iTeX' to *our* sun. . . "

[The recent announcement of a star, only 40 light years away, with seven approximately earth-sized planets, leads one to think that this could even be a possibility, however remote.]

LaTeX tutorials

Searching with Google for "LaTeX tutorial" produces the result "About 11,700,000 results (0.42 seconds)". This is rather overwhelming. How is it possible to know what is current, demonstrates best practices, and presents material in a logical order? To some extent, this is subjective, and what is helpful for one person will not necessarily be effective for another.

There are just a few tutorials listed on the TUG website.[3] While the basics of LaTeX are stable, there are always new developments worth sharing — tutorials can be instructive for both beginners and more advanced users.

Prodded by a suggestion that this might be a useful area to explore, I decided to review some of what is now available.

I admit to a strong bias in this area, based on decades of experience assisting authors in preparing manuscripts for publication by the AMS, and answering questions on various TeX forums. I am not encouraged by much of what is found on the web. For example, the first two introductory tutorials I sampled were well produced and organized logically, but they both recommended using a double backslash to start a new line of text. Prejudice against this practice is not just personal bias; the learners are being taught some things that will get them in trouble later on. These particular examples had many thousands of views reported, and favorable comments, but in my opinion they should not be recommended unconditionally, which is what I would hope for, for presentations listed by TUG.

Another example of a really good idea, but one that needs work, is the LaTeX Wikibook, `en.wikibooks.org/wiki/LaTeX`. Like the tutorials, video and audio, this includes material does not exemplify "best practices". The Wikibook *is* being worked on, but progress is slow.

It would certainly be welcome to have more good tutorials listed on the TUG website. If someone out there would be interested in reviewing what is available, please let us know. Like book reviews, which are both published here in *TUGboat* and posted on the TUG website, such exposure can (we hope) be a positive incentive for improved instruction.

The Go fonts

A family of fonts commissioned by the Go programming language project has been developed by the Bigelow & Holmes font foundry. The family (called "Go") includes proportional and fixed width faces in normal, bold and italic renderings. These fonts are particularly well adapted for use in displaying program code, with "punctuation characters easily distinguishable and operators lined up and placed consistently." They are licensed under the same open source license as the rest of the Go project's software.

[1] `zeeba.tv/tug-2010/an-earthshaking-announcement`

[2] AGASC stands for the AXAF (Chandra) Guide and Acquisition Star Catalog.

[3] `tug.org/begin.html#doc` and `tug.org/interest.html#latextutorials` with related material in the surrounding sections.

The announcement, made last November, appears at https://blog.golang.org/go-fonts.

LaTeX support was announced shortly thereafter as the gofonts package, which includes both font subfamilies under the names GoSans and GoMono; the package is available from CTAN.

Chuck Bigelow reports that the Go fonts work well with Lucida fonts

> because they have the same x-heights and similar weights as Lucida, although width metrics are like those of Helvetica and Arial. That's because the Go fonts were derived from the Luxi fonts, which were derived from Lucida fonts.

The Go proportional fonts are sans serif, especially appropriate for screen displays at small sizes. They are "humanist" in style, "derived from Humanist handwriting and early fonts of the Italian Renaissance, and still show subtle traces of pen-written calligraphy." This is in contrast to the "grotesque" style characteristic of fonts like Helvetica. (An early 19th century sans serif typeface was named "Grotesque", and the name became generic.) The monospace fonts carry slab serifs.

The italic form of the Go fonts is an oblique version of the roman, with the notable exception that the *a* is single story, redesigned to harmonize with the bowl shapes of *b* and similarly shaped letters, as shown in this sample.

abdgpq *abdgpq* **abdgpq** ***abdgpq***

abdgpq *abdgpq* **abdgpq** ***abdgpq*** **abdgpq** ***abdgpq***

The Go Mono fonts are eminently suitable for rendering computer code, as

> [they] conform to the DIN 1450 standard by differentiating zero from capital O; numeral 1 from capital I (eye) and lowercase l (ell); numeral 5 from capital S; and numeral 8 from capital B. The shapes of bowls of b d p q follow the natural asymmetries of legible Renaissance handwriting, aiding differentiation and reducing confusion.

The cited blog entry provides more complete information.

Late addition from Chuck: An upgrade of the Go fonts is (already) in the making. It consists mostly of a bug fix to the chart/box characters, plus some renaming to conform to current PostScript character naming. The Unicode "replacement" character (U+FFFD, a question mark reversed out of a black lozenge) is also being added.

On a related note, "Go" is not the first font to be named after a programming language. Adrian Frutiger designed a font named "Algol" to be used in a book about the Algol language, published in 1963.[4] This was a one-off creation in phototype, and has been long out of production. Jacques André, who told Chuck about this type, wishes someone would make a digital version. Any takers?

METAFONT at 32

The 2^5 anniversary of METAFONT was celebrated last year, but I failed to mention it at the time. Well, better late than never.

As might be expected, Don Knuth presented a retrospective, with slides, at the San Francisco Public Library. The talk was part of the "Type@Cooper West" series, and the video is posted at vimeo.com/184705112. The video is also posted at www.youtube.com/watch?v=0LR_lBEy7qU.

The slides from the talk can be found on Don's web page: www-cs-faculty.stanford.edu/~knuth/MFtalk.pdf.

Among the photographs are pictures of everyone who worked on METAFONT, font designers and graduate students alike. There's a particularly nice photo of Don with Hermann Zapf and Matthew Carter; another shows the main "team": Don with Richard Southall and Arthur Samuel as well as the students — how young they look!

From TUG'16, more from Joe Clark

Two adjuncts to Joe Clark's presentation at TUG 2016 are posted on his web site:

blog.fawny.org/2016/11/07/tug2016/
joeclark.org/appearances/tug/

The video from the conference is also up:

zeeba.tv/type-and-tiles-on-the-ttc/

The blog post contains Clark's reactions to the TUG contingent that took up his invitation to tour some of the subway stops discussed in his talk. It also contains photos that are not in the published article. And finally, it contains his admission that, after the *TUGboat* editors had "converted [it] to some version of TeX" he simply edited the TeX file, "despite my never having done that before. A reason why TeX is still in use is because it actually works."

◇ Barbara Beeton
tugboat (at) tug dot org

[4] https://tinyurl.com/frutiger-algol

Interview: Scott Pakin

David Walden

Scott Pakin has developed many LaTeX packages and other TeX-related tools.

Dave Walden, interviewer: Please tell me a bit about yourself.

Scott Pakin, interviewee: I'm 46 years old and have lived my whole life in the United States. I grew up in Chicago, Illinois (population: 2,700,000) and, after graduating high school, moved repeatedly to successively smaller cities and towns: Pittsburgh, Pennsylvania (population: 304,000) for my undergraduate degree at Carnegie Mellon University, then Champaign, Illinois (population: 232,000) for my Master's and PhD at the University of Illinois at Urbana-Champaign, and finally to Los Alamos, New Mexico (population: 18,000) to work at Los Alamos National Laboratory (LANL), where I'm still employed.

I knew from an early age I wanted to work with computers. I started programming in BASIC at age 9 on an obscure computer at my parents' company: an SDS 420 from Scientific Data Systems. (It used a 1 MHz 6502 processor and took 8-inch floppy disks, which contained maybe a hundred kilobytes of capacity.) When I was in high school, I wrote, in 8088 assembly language, a screen-dump utility called DumpHP that printed a screen of CGA graphics to an HP LaserJet printer. A small company named Orbit Enterprises licensed the code from me, incorporated it into their commercial LaserJet setup program, SetHP, and paid me royalties. Over the next few years, I made US$3000 in royalties — not bad for a teenager. I had no trouble deciding I wanted to get a bachelor's, master's, and eventually a doctoral degree in Computer Science. Along the way, I realized I especially enjoyed working with novel hardware and high-performance computers, and LANL has some of the world's fastest.

DW: Can you say something about the kinds of computing you do at LANL?

SP: At LANL, I've worked on a variety of research projects including tools for analyzing and improving the performance of supercomputers and the applications that run on them. I'm currently having great fun experimenting with a supercomputer we just bought from D-Wave Systems, Inc., that exploits quantum effects to solve a specific type of optimization problem. Think of TeX's paragraph-building algorithm, for example: It tries to find the best way to break paragraphs into lines to minimize the total penalty for awkward spacing. The research question I'm currently investigating is if it's possible to transform more-or-less ordinary looking computer programs into optimization problems suitable for running on a D-Wave system.

DW: How did you first come in contact with TeX?

SP: I began using the WordPerfect word processor (under DOS) to write documents. (WordStar was already losing popularity, and Microsoft Word hadn't yet caught on.) In my opinion, the best thing about WordPerfect was its "reveal codes" feature, which let one see the formatted document — calling it WYSIWYG would be overly generous — and the underlying markup (begin bold, end bold, begin italic, end italic, etc.) in a split-screen layout. Both were editable, but I really liked the precise control provided by the markup pane so I favored using that.

I hadn't heard about TeX until college, where a math-major friend who had recently learned LaTeX was excitedly talking about it. However, I didn't bother trying it out myself at the time. Once I began writing research papers in graduate school, I took the time to read through Lamport's book (first edition, of course) and learn LaTeX. Having been weaned on WordPerfect's "reveal codes" feature, I found LaTeX very natural to use.

Like most LaTeX newcomers, I managed to sloppily hack my way through whatever typesetting challenges I encountered. It wasn't until I started writing my dissertation that I decided to spend some effort on *really* learning how LaTeX works, how to use it more efficiently, and how to more precisely control its behavior. While doing so, I picked up TeX and LaTeX programming aspects and even wrote my first LaTeX package, `bytefield`, which I used in my dissertation.

DW: What was your dissertation topic?

SP: My PhD thesis considered processors distributed over a high-speed network working together to perform a computation fast. I presented an approach

for robustly synchronizing large numbers of such processors such that a few laggards don't necessarily slow everyone else down.

DW: You have a bunch of useful tools at CTAN[1]. Trying to grasp what is there, I can divide them into several categories: LaTeX packages; LaTeX metathings (e.g., `ctanify`, `dtxtut`, `bundledoc`); PostScript, EPS, and PDF related tools; tools for moving fonts into the TeX world (including some Metafont aspects), and combining TeX with other languages (Perl and Python). On your own website[2] you categorize things as packages, script, and documents. What motivated you to create all these different tools?

SP: Most of my LaTeX packages and programs were written to satisfy some typesetting need I had. Then, figuring that others might have the same need, I polished the code, documented it, and released it to CTAN.

It's always interesting to see which packages and programs have really caught on, and which quickly faded into obscurity. In fact, even *I* get surprised when I look over my list of CTAN contributions and find things I haven't used in years and barely even remember writing. From what I can tell, my `savetrees` package, which tries to squeeze a document into as few pages as possible, is wildly popular with researchers trying to stay within a mandated page limit for publication; and `attachfile`, which facilitates embedding arbitrary files within a document, and `hyperxmp`, which lets one include a large amount of metadata in a document, also seem to get a fair amount of attention. On the other hand, `spverbatim`, which enables verbatim text to wrap at spaces; `listliketab`, which typesets lists that arrange data in columns; and the `newcommand` script, which generates `\newcommand` templates for macros with complex juxtapositions of required and optional arguments, apparently get little or no use. Heck, I don't think *anyone* has ever used `dashrule`, which draws dashed horizontal lines.

DW: You may be too pessimistic about `dashrule`; it's recommended at `http://tex.stackexchange.com/a/125503` and is mentioned in many other places on that website.

DW: Both the Visual LaTeX FAQ and the Comprehensive LaTeX symbol list seem like they must have been enormous efforts. How have you gone about creating each of these?

SP: Indeed, the Visual LaTeX FAQ required quite a bit of effort to create, and the Comprehensive LaTeX Symbol List required substantially more. In both cases, one big challenge was to incorporate mutually conflicting elements in the same document. The symbol list, which tabulates a vast number of symbols that LaTeX documents can typeset, started with relatively few packages — base LaTeX, AMS, St Mary's Road, wasy — so it began being reasonably manageable. However, each new symbol package that gets added brings a new source of woe. Perhaps the biggest headache is that TeX has a hard-wired limit of 16 math alphabets. I typically have to access math fonts as if they were text fonts in order not to overflow that limit. Even worse, I've recently been encountering newer symbol packages that require LuaLaTeX or XeLaTeX, while some older packages break when using those TeX engines. Each new release of some symbol package seems to introduce a new conflict with some other package. As a result, the symbol list has become almost completely unmaintainable. I've begun work on a complete rewrite that should be robust to those issues, but that effort is slow-going and is still many years away from being usable.

DW: Please tell me your thoughts on the overall TeX infrastructure and whether you think it can be made better given practical limitations.

SP: TeX and LaTeX have a thriving infrastructure in terms of the sheer number of readily available LaTeX packages and the great strides being made in recent years enhancing the underlying TeX engine with improved support for system fonts and improved automation using Lua.

A practical limitation is getting new users to adopt the TeX ecosystem. Despite being only four years older than Microsoft Word, TeX has a far more "old-school" feel to it. Yes, TeX installation has improved over the years; and yes, GUIs do exist to simplify usage, obviate the need to learn control sequences, and provide word-processor-like synchronous editing. However, (La)TeX's lack of integration is a huge shortcoming relative to a word processor. If a user wants to typeset a table in a particular form, does he/she use an ordinary tabular environment or load one or more of the `array`, `bigtabular`, `booktabs`, `btable`, `calls`, `colortab`, `colortbl`, `ctable`, `dcolumn`, `easytable`, `hvdashln`, `longtable`, `ltablex`, `makecell`, `mdwtab`, `multirow`, `polytable`, `sltables`, `stabular`, `supertabular`, `tables`, `tabls`, `tabu`, `tabularborder`, `tabularew`, `tabularht`, `tabularkv`, `tabularx`, `tabulary`, `threeparttable`, `threeparttablex`, or `xtab` packages? Even worse, many packages conflict with each other either explicitly (giving an error message) or implicitly (screwing up some unrelated aspect of the document in some hard-to-diagnose manner). Worse

still, the set of conflicts can change from version to version of any given package.

Another example of (LA)TEX's lack of integration relative to a word processor is that a word-processing document is stored in a single file that can easily be transmitted to colleagues. My `bundledoc` script helps with this on the LATEX side by bundling together all the separate document files, style files, class files, graphics files, etc. into a single `.tar` or `.zip` file, but usage is still a bit clunkier than what an integrated tool can provide.

Word processors have been improving their typesetting quality, support for mathematics, support for international scripts, logical structure, and other features that have traditionally lain in TEX's wheelhouse. For most users, word processors are good enough tools for the jobs they have. I think the wrong approach is to try to turn (LA)TEX into a word processor. It lacks the foothold of, say, Microsoft Word and is unlikely ever to become a dominant form of document interchange. Instead, (LA)TEX infrastructure enhancements should focus on the system's core strengths: ease of making global, structural changes to an entire document; ease of automation; and ready and convenient support for a variety of specialized typesetting requirements in areas such as linguistics, mathematics, and natural sciences.

DW: Given you've built various PostScript, EPS, and PDF tools (such as `purifyeps`), do you have thoughts on what might practically be done with (LA)TEX to make them more suitable for integrating with PDF et al.?

SP: I guess I don't have any grand vision for better integration of (LA)TEX with the PDF world. That said, native support for PDF/A-1a generation and fully tagged PDF would be nice. The former guarantees a high degree of portability, and the latter facilitates reflowing text on a tablet and improves mechanical reading of a document to the vision-impaired.

DW: At the 2014 TUG annual conference in Portland, Mertz, Slough and Van Cleave presented a paper[3] that included a significant discussion of your `bytefield` package. Also, Mertz and Slough previously presented a lengthy discussion[4] of your PerlTEX system. I am interested in how you feel about other people describing your work and whether they interacted with you as they wrote their papers.

SP: I'm always eager for people to use my tools. It's wonderful to know that I helped someone get the typesetting they were looking for or automate some tedious task.

I was not contacted by the authors of the papers you cite above, but that's probably a good sign; it says the authors were able to get `bytefield` and PerlTEX to work without extra help. For PerlTEX, which lets users write LATEX macros in Perl, that's especially encouraging. PerlTEX was extremely challenging to develop and is therefore likely to be a lot more fragile than a typical LATEX tool. It requires a lot of TEX trickery to process what could be considered syntactically incorrect TEX but syntactically correct Perl from within TEX, and it takes a Computer Science-y distributed-systems-style protocol to implement correct, two-way communication between TEX and a Perl wrapper script given TEX's limited ability to communicate with the outside world in a safe (i.e., not-`\write18`) and portable manner. It's great that PerlTEX works fine for Mertz and Slough and that they were able to perform some interesting and creative tasks with it.

DW: Do you still see a role for PerlTEX with LuaTEX now available?

SP: Not so much. LuaTEX deeply integrates Lua with the TEX engine while PerlTEX is more loosely coupled. Consequently, there are things LuaTEX can do that PerlTEX can't (e.g., directly manipulating some of TEX's internal representations). On the other hand, I find PerlTEX's `\perlnewcommand` and `\perlnewenvironment` macros very convenient. Perhaps I should write a package that provides the analogous `\luanewcommand` and `\luanewenvironment` macros ....

DW: You've also developed lots of other tools, such as those listed on your personal website (readers: see `http://www.pakin.org/~scott/`). Perhaps you also do not hesitate to build a new tool in the course of accomplishing your work at LANL[5]. Can you speak about tradeoff between (a) just doing what you have to do to accomplish some primary task, and (b) first building a tool to help you with the primary task and then applying it to accomplish the task?

SP: I write lots of tools, and I always learn something new when I do. It's always a good idea, though, to perform a task manually the first few times to determine what aspects are suitable for generalization and automation — and to convince yourself that the task is in fact something that gets performed sufficiently often as to warrant building a tool for it. I suppose the following are a good set of questions a tool-builder might ask himself before embarking on developing a new tool or, in the context of this discussion, a new LATEX package:

- Is the task sufficiently common as to warrant building a tool for it?

- Is the task sufficiently complex for users to be willing to install and learn a new tool rather than perform the task manually?
- Is the task sufficiently general for a tool to perform it without having to be so parameterized that it becomes almost as difficult to learn and use as it is to perform the task manually?

DW: Were any of these tools particularly more fun to work on?

SP: It's hard to pick a single, most fun piece of LaTeX development. PerlTeX is the most sophisticated LaTeX-related tool I've ever created, and it was exciting when I finally got that to work. My three standalone documents — The Comprehensive LaTeX Symbol List, The Visual LaTeX FAQ, and How to Package Your LaTeX Package — all required a fair amount of thought to produce, and I learned quite a bit from each one. I guess the common thread is that tools, packages, and documents that were intellectually challenging to develop are more rewarding than those that required only straightforward coding.

DW: Thank you for taking the time to participate in this interview. You are working on a lot of fascinating things.

[Interview completed 2017-02-05]

Links

[1] https://www.ctan.org/author/pakin
[2] http://www.pakin.org/~scott/latex-stuff.html
[3] https://www.tug.org/TUGboat/tb35-2/tb110mertz.pdf
[4] https://www.tug.org/TUGboat/tb28-3/tb90mertz.pdf
[5] https://ccsweb.lanl.gov/~pakin/

◇ David Walden
http://tug.org/interviews

What's a Professor of Neurology doing using LaTeX?

David B. Teplow, Ph.D.

Abstract

In the general biomedical and academic research communities, most people have never heard of LaTeX. Students and professors in the humanities, social sciences, biological sciences, and clinical medicine who have heard of LaTeX often don't appreciate the value of such a sophisticated and elegant typesetting engine. Instead, as with most of the world, they succumb to the forces of the *Dark Side* (traditionally Microsoft Corporation) and use expensive, inflexible, closed source, and relatively primitive programs to compose documents. They also suffer the continuing frustrations of doing so. I discuss here my own journey out of the darkness and into the light of LaTeX.[1]

1 Introduction

My first exposure to document creation, like most in my generation, was in English class in elementary school. Documents all were hand written and one was graded on "penmanship." As one got older, one was expected to use a typewriter to produce professional looking documents. It was not until the 80s, with the introduction of the IBM PC (personal computer) in 1981 and the Apple Macintosh 128K in 1984, that the average consumer could compose documents electronically and print them using line printers or dot matrix devices. I remember my excitement running WordPerfect or WordStar on my IBM PC and MacWrite on my Macintosh Plus (with 1 MB of memory and a 20 MB disk drive large enough to use as a crane counterweight). One actually could "program" how individual letters, sentences, paragraphs, or document sections should look. This was done using keyboard commands embedded in the text. WYSIWYG GUIs came later and were seen as revolutionary. One no longer had to guess how their typescript would look. It was right there in front of you on the screen.

Development of personal computers and software has continued during the almost four decades since a vision for personal computing occurred at IBM. The most important software design principle was "do more and make it easier." Unfortunately,

[1] The reader is cautioned that what follows is my personal perspective on LaTeX. This perspective is not meant to be, nor is it, a definitive review of LaTeX and its uses. I ask the indulgence of *TUGboat* readers if they find any inaccuracies in this article and would be grateful if these inaccuracies were brought to my attention.

and particularly in the case of Microsoft Word, doing more and making it easier actually meant "make it more complicated, inflexible, and buggy." When the Unix-based Mac OS X operating system came to the Macintosh platform with its protected memory and preemptive multitasking architecture, software programming errors only crashed the application in use, not the entire computer and not that often. The exception, of course, was Microsoft Word, which to this day can be counted on to crash at the least opportune moments, often making one's prior work unrecoverable.

As a professor, I had to continually compose documents, be they notes, letters, grant applications, or manuscripts to be published in academic journals. Text processing capability was mandatory and Word was the *de facto* standard for this purpose. This meant that year after year, decade after decade, I, like others, had to suffer the frustrations inherent in trying to get text processing "bloatware" to do what one wanted it to do. These frustrations included, among many, application crashes, the well known "Word has insufficient memory" error messages when one tries to save a file, difficulties embedding images and maintaining their location during subsequent editing, problems formatting tables, bizarre placement of equations constructed using MathType or Equation Editor, and an inability to stop the program from "helping" you by automatically changing formatting, word spelling, and other aspects of document creation.

These computing and composition experiences made clear a desperate need for a better method of document composition. Enter LaTeX.

2 How I met LaTeX and why I fell in love with it

My first exposure to LaTeX occurred in the context of a collaborative effort to understand the mechanistic bases of Alzheimer's disease. The collaboration integrated biochemical and computational studies of protein aggregation. My laboratory carried out the biochemistry work while the computational studies were done by physicists. When our studies were complete, we discussed how and where to publish our results. I had assumed that our manuscript would be composed using Word, which I suggested to my physicist colleagues. To my surprise, the leader of the physics group, a world authority in the field of statistical physics, told me that he did not know what Microsoft Word was! All word processing in his group was done using LaTeX, which I had never heard of. The composition and publishing of scientific manuscripts is an arduous process that requires tremendous attention to detail and many, many iterations during initial manuscript creation and the peer review process. Authors must use the same text processing platform during this process. It thus appeared that either my learned colleague and his group would have to learn Word or I would have to learn LaTeX.

I chose to be the one to learn a new document preparation system. I did so for a number of reasons, some practical and some personal. The practical reason is that professors tend to become ossified as they age, which means that change can be difficult. It would be easier for me, as a new Assistant Professor, to learn a new system than it would for my senior colleague, a distinguished Professor of Physics. The second reason was my high regard for the academic acumen of most physicists, which suggested that the tools they used in their research, including those for document preparation, likely would be powerful and elegant. As I mentioned above, the introduction of the Unix-based Mac OS X operating system made the Mac platform remarkably powerful because now one could take advantage of the huge reservoir of expertise and software associated with Unix, which of course included LaTeX. I quickly learned that the LaTeX source files were simple ASCII text files. I could work on my Mac, either inside a LaTeX GUI or in Terminal, while my colleagues could use their PCs and we could easily exchange files and be certain they would compile,[2] regardless of platform. This eliminated the continuing problem of format alterations in Word files caused by file movement between Mac and PC platforms. I was tremendously impressed with the professional layouts of our manuscript after source file compilation. For the first time in my academic life, I could create manuscripts that, essentially, *were already typeset.* They were beautiful. Finally, from the perspective of a science nerd, I found the prospect of learning what essentially is a programming language to be very exciting.

As I became more adept at using LaTeX, I realized that it provided many capabilities that were superior to those of standard word processing programs. One of the biggest headaches in the composition of scientific manuscripts is the need to change figure and table numbers if such items are added or deleted from manuscript drafts. One can do this manually if one is particularly attentive to detail, or automatically using search and replace functions. However, this is time consuming and often results in multiple figures or tables having the same number, which is confusing — especially to peer reviewers

[2] Assuming no trivial coding errors existed.

and editors who decide if your manuscript will be accepted for publication. Enter the LaTeX `\label{}` command! What an easy and elegant way to ensure that any editorial changes result in automatic, accurate renumbering of figures and tables.

I was equally, if not more, delighted by BibTeX, especially after struggling with EndNote for so many decades. I could now create a single library and format my bibliography using pre-existing bibliography style `.bst` files — no more hassles with constantly having to edit EndNote output styles. I could edit my library in any text editor or use one of the many reference management programs available. I've used JabRef and BibDesk, among others, and find JabRef particularly useful.

The use of pre-existing class and style files illustrates the power, ease of use, and time efficiency of the LaTeX platform. If I am required to use a particular class for a publication, I simply download it from the web or get it from the publisher (as I did for the `ltugboat` class used for this article). I can compose my source file without any concerns about it being properly rendered. Of course, as *TUGboat* readers well know, and as neophyte LaTeX users rapidly learn, TeX and its derivatives are designed to enable writers to focus on the *content* of their work as opposed to its *formatting*. I no longer have to spend time carefully reading document formatting instructions from a publisher or agency to whom I am submitting a grant application and then converting these instructions into an acceptably formatted Word document. The class and style files do it all for me.

Experienced computer users know that it is most time efficient to operate your computer by leaving your fingers on the keyboard, as opposed to constantly having to manipulate a mouse or other input device. This is no more evident than when one is creating mathematical formulas. Although Equation Editor and MathType are useful point-and-click formula creation applications that interface seamlessly with Word, one must invoke either one and then click, click, click . . . to create the formula, which then is inserted, often with bizarre vertical alignment within text lines, into the document. In addition, it is common to find that these formulas are rendered improperly once the file has been typeset by a publisher. When I create formulas in LaTeX, I can do so without lifting my hands off the keyboard and I do not experience any subsequent formatting or rendering errors.

Anyone who has tried to create lists using standard word processors likely has encountered problems with indentation, nesting of list elements, and most vexing, stopping the program from adding new text to the end of an existing list. The fine control of the list environments in LaTeX eliminates these problems. Similar advantages exist with respect to table creation. I am an experienced Word user (unfortunately), but I still can't figure out how to format and align tables in a reasonable amount of time.

I find that figure and caption placement can be problems both for word processor and LaTeX users. One also encounters figures that mysteriously change their positions within a document. In Word, these problems are exacerbated by the fact that figures and captions are entered independently.

3 How I learned LaTeX

As we all know, LaTeX, in essence, is a programming language. Its code may be less complicated than C++, Fortran,[3] Objective-C *et al.*, but it nevertheless requires the user to create source code that instructs a compiler how to produce a properly rendered document. One of the beauties of LaTeX is that a new user need not know anything to begin using LaTeX other than how to open a `.tex` file in a text processor. This was how I began the learning process, by simply editing the text within the source files created by my collaborators. It was easy to learn how to encode underlined, italicized, or bolded text. After all, how hard is it to "escape" the obvious "bf" abbreviation for bold font and type `\bf`?[4] In the process of assimilating this simple syntactical information, one begins to get a sense of how LaTeX programming works and this sense then provides a framework for adding new skills to one's repertoire. "Environments" then are encountered that require learning how they are parameterized and about what can and cannot be done within them. At this point, the neophyte LaTeX user needs to begin studying the language more deeply.

I found myself doing what any self-respecting academic would — I bought books. The first two are well known in the TUG community, namely *Guide to LaTeX* by Kopka and Daly and *The LaTeX Companion* by Mittelbach and Goossens. These two volumes became my "go to" references for questions. I also found *First Steps in LaTeX* by Grätzer, *LaTeX Line by Line* by Diller, and *Learning LaTeX* by Griffiths and Higham to be useful. Scientific publications usually contain tables and figures. In the beginning, as I began creating my own `.tex` files, it was simple to copy and paste a figure environment from a file of my collaborators and just insert the path to my own

[3] Including Fortran 4, which I used a half century ago!

[4] Interestingly and ironically, I just now learned how to escape the backslash so all the following text was not bold!

figure. This got me started. I also cut and paste table environments. However, to gain more expertise in managing these environments, I added *Typesetting Tables with LaTeX* by Voß, and *The LaTeX Graphics Companion* by Goossens, Rahtz, and Mittelbach, to my "go to" references. These days, however, I rarely consult these references, not because they are uninformative, but because so much detailed information is available on the web. I routinely access `tex.stackexchange` if I need help, download package manuals, or access other sources found through web searches. It's remarkable how many preamble lines, environments, minipage formats, and other bits of code one can simply cut from web pages and paste into their source file to achieve a particular typesetting goal without any *pre facto* syntactical knowledge.

I find "playing" with LaTeX to be a lot of fun. It's often a challenge to render and position text, figures, and tables in a specific way. I like trying different things and seeing the output. I might switch between standard `figure` and `wrapfigure` environments, use minipages, or try other methods to achieve a particular document rendering. The process of self-directed investigation provides rich rewards in terms of better understanding how LaTeX works and how to manipulate output, as opposed to memorizing how to perform a single task. One is able to develop an intuition that facilitates learning and accelerates the process of problem solving.

4 How I use LaTeX

"If you got a terminal, you can use LaTeX."

This certainly is true for those who are ⋆nix (Unix, Linux *et al.*) savvy or love the command line. However, the majority of the world's computer users interact with their computers through GUIs. A major advance for the general LaTeX community, one that made the use of these programs much more attractive to the average computer user, was the introduction of GUI front ends to LaTeX compilers. Users were able to run LaTeX by pointing and clicking with their mice. No knowledge of ⋆nix commands and syntax were required. What was required, both for command line users and GUI users, were multiple steps before a finished document could be viewed. For academicians, for whom extensive referencing is required, the process included triple compilation (LaTeX→BibTeX→LaTeX) so that references were numbered correctly and a bibliography was created. Multiple steps also were required to produce an output file that could be easily shared with others who might be relatively computer illiterate or worked using different platforms and operating systems (e.g., Mac and MacOS, PC and Windows, terminals and ⋆nix). This typically involved a tex→dvi→pdf compilation and conversion process. One also could produce output files in other formats, including postscript, html, or rtf, but pdf was the most useful for collaborations and submission of manuscripts to most journals. These processes were not onerous in nature, but they were burdensome and time-consuming. For those used to WYSIWYG document composition, this need to first compile the source code before seeing the finished work product was a bit off-putting. However, with the advent of three-panel application interfaces (file directory, source file, rendered output) and automatic file compilation, users can immediately see the results of their work. This has been an important development because it has streamlined the document preparation process for the average computer user, eliminating the need to understand the source file→compiler→output file process.

My initial LaTeX front end was TeXShop, which provided a simple, useful method for compiling source files and viewing their output. As one who enjoys determining if newly developed or updated applications might offer an easier or more powerful user experience, I also have used Texmaker, TeXworks, TeXnicle, Texpad, TeXstudio, and Latexian, as well as web-based document creation and compilation engines like Overleaf (formerly writeLaTeX) and ShareLaTeX. LyX is unique among these front ends in that its default document view hides the source code from the user and its interface looks more like the icon-based interfaces of non-TeX-based word processors. It also requires the user to port the output file to a different application (e.g., Adobe Acrobat) to view the rendered output. I found this type of GUI to be an unhappy medium between the extremes of document preparation, i.e., using a terminal or using a standard word processor (e.g., Word).

I am composing this document using Texpad, which is my current favorite. A helpful feature of Texpad and other programs is their handling of compilation errors. Error and output logs are instantly available for user review either within the main application window itself or by a simple click on an icon. In addition, the user is provided the means to rapidly edit offending syntax simply by clicking on a particular error message in the message viewer, which then moves the focus of the keyboard to the offending line of code. This saves a lot of time during the error correction process. Other features of these front ends that are particularly useful are code completion, flash bulb-like highlighting of beginning and ending characters (e.g., curly braces), and syntax highlighting.

To further facilitate document creation, and to deal with the progressive memory loss experienced by Professor Emeriti, I also create a variety of preambles, tables, and figure environments and store them in a special directory. I then can simply cut and paste the code into new documents without having to remember any special syntax that I might have used in the past. For example, as a professor, I am asked to compose many different kinds of recommendation letters, including those for undergraduates, graduate students, postdoctoral fellows, different professorial ranks, etc. To do so, I have created a directory in which boilerplate letters for each type of recommendation exist. When I need a template, I can rapidly access these pre-made files. This has made composition of new letters trivial. The same strategy is used for grant application preambles, be they for the National Science Foundation, the National Institutes of Health, or other agencies.

5 Using LaTeX in a non-LaTeX world

Readers of this article already know, and likely much better than I, how powerful, flexible, and efficient LaTeX is. These are some of the reasons we choose to use this document preparation system. Unfortunately, the rest of the world either is not aware of the existence of LaTeX or is precluded from using it due to restrictions on how document preparation is to be done. The latter restriction usually is imposed on employees to ensure company-wide consistency in document preparation, which is a reasonable concern. Establishing a standard application for document preparation allows diverse groups of people to seamlessly exchange documents.[5]

Microsoft Word has become the *de facto* standard document preparation application. There are many reasons for this, all of which can be debated among computer users, businesses, the general public, educators, sociologists *et al.*, but one of the key reasons, vis-à-vis why LaTeX is not a standard, is that Word uses a GUI as opposed to the command line interface of LaTeX. This makes Word easier to learn for the vast majority of computer users, who are not capable of using the command line for document preparation or simply may not want to do so. As a realist, I do not expect this situation to change. I also think it unlikely that proselytizing for LaTeX converts will be particularly effective, especially in a world in which OUIs ("oral user interfaces") appear destined to supplant keyboard, mouse, and other data entry methods, as well as supplant many aspects of application and system control.[6]

Where then do these facts leave the LaTeX community as a whole? I suggest, in the future, that the community will continue to thrive, as it is now. There are myriad reasons for this, many of which have been discussed above. The most important of these is that for many applications, especially in mathematics, physics, and engineering, LaTeX is a superior document preparation system.[7] Given this fact, those who choose to prepare their own documents using LaTeX, but who also work in the larger world of Word and other document preparation applications, must implement strategies for interfacing these two "worlds."

The specific strategies depend on a number of factors, the most important of which are how the master document is to be prepared and what the final output file format must be. The first factor depends primarily on whether document preparation is done by a single person or in the context of a collaboration. The second generally is dictated by the requirements of the end-user of the document, e.g., a publisher. Publishers specify the file types accepted for publication, which increasingly include PDF. The beauty of LaTeX is the facile compilation of the source code as a PDF file, which renders moot the original document preparation system. This PDF output also circumvents problems with providing documents to those working with computer hardware or OSs different from one's own.

If manuscripts can be submitted for publication using one of many file types, e.g., `.tex`, `.doc`, or `.docx`, and the manuscripts present collaborative work, the decision about source file type can be pre-determined among collaborators, usually using a metric based on ease of group composition. Practical considerations also may factor into this decision. For example, if the contributing author, the one who is responsible for the actual compilation of the manuscript and its submission for publication, is not familiar with LaTeX, then Word often becomes the default document preparation application. However, if I am the contributing author and I want to prepare the manuscript using LaTeX, I can do so by adding files from my collaborators into my `.tex` source file. The easiest way to do this is to ask for `.txt` files. Surprisingly, I often encounter collaborators who don't

[5] Of course, *though seamless in theory*, cross-platform (MacOS vs. Windows) document preparation and management using Microsoft Word (or PowerPoint) remains problematic and vexing in practice.

[6] In fact, there is no reason, theoretically, why such OUIs could not be implemented for source file creation in LaTeX.

[7] It should be noted that LaTeX is not restricted to these fields. It has been used effectively in a broad range of fields, including philosophy, economics, theology, the law, and *neurology*.

know how to create `.txt` files and instead provide `.doc` or `.rtf` files. These then must be converted into plain text.

Conversion can be done automatically using a variety of programs or web-based conversion utilities, although I have found that the fidelity of conversion often is lacking. Post-conversion processing of the resulting text file thus is required to remove hidden or special characters that create serious or fatal errors during LaTeX compilation. I have found that utilities that clean up text, e.g., by removing extra spaces, carriage returns, tabs, or forwarding characters, are especially useful in this regard. Once clean, I then execute a second post-conversion process, usually using global find-and-replace functions, to make sure, among other things, that quotation marks will be rendered properly (i.e., converting " and " into `` and ''), percentage symbols are escaped (% to \%), Greek characters are encoded properly, and one-, two-, and three-em dashes will appear correctly. These conversions can be done quite rapidly, after which the plain text can be pasted into the source file. Additional edits, which usually are minor, then are done in the source file after compilation if error messages are displayed or rendering problems exist.

"Cross-world" bibliography creation and management is a bit more cumbersome, if one defines cumbersome as an author needing more than one library. I maintain two comprehensive reference libraries, one in `.bib` format and one in EndNote format (`.enlp`). In collaborative work in LaTeX, the collaborators agree about whose library will be used and simply upload it to a shared directory containing the manuscript source file. This is trivial. If I must incorporate citation markers from Word documents, regardless of their provenance (Word, Bookends, Mendeley), then I do so manually. This requires a significant amount of program switching (LaTeX↔JabRef), but with the powerful search capabilities of library management software, and patience, the process is straightforward. It should be mentioned that neither world is free of typographical problems within rendered bibliographies. Surprisingly, special symbols, capitalizations, and especially Greek characters, are usually not coded properly in references downloaded from the web, especially in the case of EndNote. I automatically examine each downloaded reference to ensure that the bibliography created by BibTeX or Endnote is an exact rendering of the reference as originally published. After having done this for so many decades, this process has become almost autonomic.

The fundamental principle guiding my cross-world collaborations is "LaTeX takes anything and Word takes nothing." This means that if a manuscript is composed in LaTeX, I can take content from essentially any file type provided by a collaborator and incorporate it into my source file. In contrast, if manuscript composition is to be done in Word, then LaTeX is not used at all.

6 Concluding remarks

Among the community of computer users who are free to choose their hardware, software, and style of use, adherence to their choices may have the flavor of religious fanaticism. This long has been true in the Mac community, not even considering Apple's efforts to portray Mac users as "cool," "with it," or "different." I am a long-time Mac fanatic, not because of such superficial characterizations but rather because of my recognition of a superior operating system and how it makes my computational efforts easier and more efficient. My extensive experience with document processing systems has led me to the same recognition with respect to LaTeX. I look forward to a time when this recognition will be universal.

Acknowledgement

The author gratefully acknowledges Dr. Eric Hayden (UCLA) for helpful comments on this manuscript.

◇ David B. Teplow, Ph.D.
Professor Emeritus
Department of Neurology
David Geffen School of Medicine at UCLA
635 Charles E. Young Drive South
Los Angeles, CA 90095
USA
dteplow (at) mednet dot ucla dot edu
http://teplowlab.neurology.ucla.edu/

Typographers' Inn

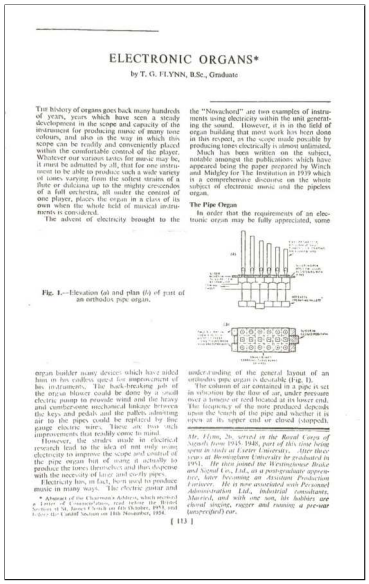
Peter Flynn

Layouts

The three main default document classes in LaTeX (book, report, and article) implement a consistent page layout with conservative features:

- centered title block or title page;
- wide set and wide margins;
- centered, narrower Abstract in smaller type, indented;
- justified text;
- indented, unspaced paragraphs;
- bold sectional divisions, widely-spaced;
- heavily-indented lists with wide spacing;
- non-hanging footnotes;
- floating tables and figures with caption position dependent on caption text quantity;
- same-size type in block quotations with independent indentation;
- a single typeface throughout (Computer Modern).

At the time LaTeX was written (1985), these defaults were already quite conservative, and have been described as 'based on then-common conventions for scientific publishing' [5]. I recently came across an article a relative wrote in the 1950s — it could almost have been formatted with LaTeX's defaults (see Figure 1).



ELECTRONIC ORGANS*

by T. G. FLYNN, B.Sc., Graduate

The history of organs goes back many hundreds of years, years which have seen a steady development in the scope and capacity of the instrument for producing music of many tone colours, and also in the way in which this scope can be readily and conveniently placed within the comfortable control of the player. Whatever our various tastes for music may be, it must be admitted by all, that for one instrument to be able to produce such a wide variety of effects, ranging from the softest strains of a flute or diaphone up to the mighty crescendos of a full organ — under the control of one player, places the organ in a class of its own when the whole field of musical instruments is considered.

The advent of electricity brought to the organ builder many devices which have aided him in his endless quest for improvement of his instruments. The back-breaking job of the organ blower could be done by a small electric pump to provide wind and the heavy and cumbersome mechanical linkage between the keys and pedals and the pallets admitting air to the pipes could be replaced by fine gauge electric wires. These are two such improvements that readily come to mind.

However, the strides made in electrical research lead to the idea of not only using electricity to improve the scope and control of the pipe organ but of using it actually to produce the tones themselves and thus dispense with the necessity of large and costly pipes.

Electricity has, in fact, been used to produce music in many ways. The electric guitar and the "Novachord" are two examples of instruments using electricity within the unit generating the sound. However, it is in the field of organ building that most work has been done in this respect, as the scope made possible by producing tones electrically is almost unlimited.

Much has been written on the subject, notable amongst the publications which have appeared being the paper prepared by Winch and Midgley for The Institution in 1939 which is a comprehensive discussion on the whole subject of electronic music and the pipeless organ.

The Pipe Organ

In order that the requirements of an electronic organ may be fully appreciated, some

Fig. 1.—Elevation (*a*) and plan (*b*) of part of an orthodox pipe organ.

understanding of the general layout of an orthodox pipe organ is desirable (Fig. 1).

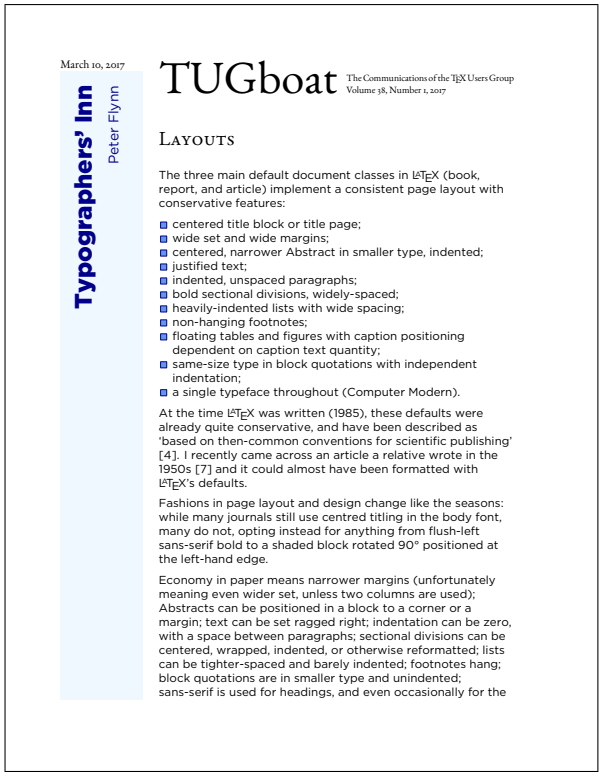
The column of air contained in a pipe is set in vibration by the flow of air under pressure past a tongue or reed located at its lower end. The frequency of the tone produced depends upon the length of the pipe and whether it is open at its upper end or closed stopped.

Mr. Flynn, Sr., served in the Royal Corps of Signals from 1935–1948, part of this time being spent in radio at Exeter University. After those years at Birmingham University he graduated in 1951. He then joined the Westinghouse Brake and Signal Co., Ltd., as a post-graduate apprentice, later becoming an Assistant Production Engineer. He is now associated with Personnel Administration Ltd., Industrial Consultants, Married, and with one son, his hobbies are choral singing, rugger and running a pre-war (unspecified) car.

[113]

Figure 1: 1950s journal style [4]

Fashions in page layout and design change like the seasons: while many journals do still use centered titling in the body font, many do not, opting instead for anything from flush-left sans-serif bold to a shaded block rotated 90° positioned at the left-hand edge (as in Figure 2).



March 10, 2017

Typographers' Inn

Peter Flynn

TUGboat The Communications of the TEX Users Group Volume 38, Number 1, 2017

Layouts

The three main default document classes in LaTeX (book, report, and article) implement a consistent page layout with conservative features:

- centered title block or title page;
- wide set and wide margins;
- centered, narrower Abstract in smaller type, indented;
- justified text;
- indented, unspaced paragraphs;
- bold sectional divisions, widely-spaced;
- heavily-indented lists with wide spacing;
- non-hanging footnotes;
- floating tables and figures with caption positioning dependent on caption text quantity;
- same-size type in block quotations with independent indentation;
- a single typeface throughout (Computer Modern).

At the time LaTeX was written (1985), these defaults were already quite conservative, and have been described as 'based on then-common conventions for scientific publishing' [4]. I recently came across an article a relative wrote in the 1950s [7] and it could almost have been formatted with LaTeX's defaults.

Fashions in page layout and design change like the seasons: while many journals still use centred titling in the body font, many do not, opting instead for anything from flush-left sans-serif bold to a shaded block rotated 90° positioned at the left-hand edge.

Economy in paper means narrower margins (unfortunately meaning even wider set, unless two columns are used); Abstracts can be positioned in a block to a corner or a margin; text can be set ragged right; indentation can be zero, with a space between paragraphs; sectional divisions can be centered, wrapped, indented, or otherwise reformatted; lists can be tighter-spaced and barely indented; footnotes hang; block quotations are in smaller type and unindented; sans-serif is used for headings, and even occasionally for the

Figure 2: Re-envisioning some of LaTeX's defaults

Many publishers and journals have written their own document classes which can be downloaded (some are included with TeX Live), and many helpful people have written classes and packages which provide alternative layouts (the memoir and koma-script bundles are excellent examples).[1] You can of course go the whole hog and employ a designer to draw up a suitable set of (possibly corporate) designs, and then get them implemented as LaTeX classes by one of the TeX consultants listed at the back of every issue of *TUGboat*.

However, for the users who just want to create simple layouts of their own, here are some guidelines to current practice that can be used. After all, if LaTeX were to be written today, what defaults would we want?

[1] I tried to introduce one myself (vulcan) many years ago [1], which was a compendium of packages and modifications, but it required more resources than I had available at the time.

Titling Redefine the `\maketitle` command to put the title, author, date, and other metadata where you want it.

Margins Reset margins with the geometry package.

Abstract Redefine the `abstract` environment to control the font size and positioning.

Justification This is still the standard for books but ragged right is less formal and has been popular in wordprocessors for decades.

Spacing The parskip packages turns off indentation and uses space between paragraphs instead. This is a popular office-document format.

Sections The sectsty and titlesec packages let you change the styling of headings. In justified text, set headings to ragged right to avoid hyphenation and justification (H&J) problems.

Lists Use the enumitem package to restyle all types of lists, not just enumerated ones.

Footnotes The footmisc package (and others) provide alternative layouts for footnotes.

Tables and figures Caption styling can be changed with the ccaption package and others.

Block quotations Redefine the `quotation` environment to get rid of the first line indent, and to change the font size.

Typefaces Use XeLaTeX to get a much wider and more easily-accessible choice of typefaces [3], but even with `pdflatex`, there is a substantial range available (see the LaTeX Font Catalogue `tug.dk/FontCatalogue`). If you have mathematical content your selection is more limited.

I haven't mentioned the letter class. It's something of an anomaly, although it too implements what was a fashionable layout in the days of the typewriter: a vertically-centered document (that is, equal amounts of white-space above and below). I'm not aware of anywhere using this layout today, and restyling it is much harder than with the other classes. However, the koma-script bundle mentioned above provides a very adaptable alternative.

Go forth, etc...

Afterthought

Just after the last *Typographers' Inn* hit the streets, I spotted another example (see Figure 3) of a centered heading broken unusually [2, 3]. In the early centuries of printing, it wasn't considered 'bad'; in fact it occurs frequently on title pages.

In a narrow measure, long headings are going to cause H&J problems because they are typically set in a much larger size than the body copy.[2]

[2] Your Editor-in-Chief, Barbara Beeton, and your Production Manager, Karl Berry, will know this only too well from *TUGboat*.


THE CON-
tentes of this Booke.

i. A preface.

ii. A table and Kalendar for psalmes and Lessons, with necessary rules pertenyng to the same.

iii. The ordre for Matins and Euensong, throughout the yeare.

iiii. The Introites, Collectes, Epistles and Gospelles, to be vsed at the celebracion of the lordes Supper, and holy Communion through the yere, with proper psalmes and Lessons, for diuerse feastes & dayes.

Figure 3: Unusual line-breaking in a heading (Book of Common Prayer, 1549, fragment, courtesy The Society of Archbishop Justus)

This example provides us with two insights into the minds of the 16th-century compositor and page designer (if they were two separate people):

1. that he felt it necessary to expand 'The Contentes' by adding 'of this Booke' (perhaps 'Contentes' alone was not yet established);
2. that he felt it was perfectly normal for a word which needed to be hyphenated to be continued in a different size and even a different typeface on the second line.

We do live in a different worlde.

References

[1] Peter Flynn. The vulcan Package — A Repair Patch for LaTeX. *TUGboat*, 20(3):208–213, Sep 1999. `http://tug.org/TUGboat/tb20-3/tb64flynn.pdf`.

[2] Peter Flynn. Typographers' Inn — Formatting and centering. *TUGboat*, 33(1):8–9, Jan 2012. `http://tug.org/TUGboat/tb33-1/tb103inn.pdf`.

[3] Peter Flynn. Typographers' Inn. *TUGboat*, 37(3):266, Dec 2016. `http://tug.org/TUGboat/tb37-3/tb117inn.pdf`.

[4] Thomas George Flynn. Electronic Organs. *Institution of Electrical Engineers Students' Quarterly Journal*, 25(98):113–118, Dec 1954.

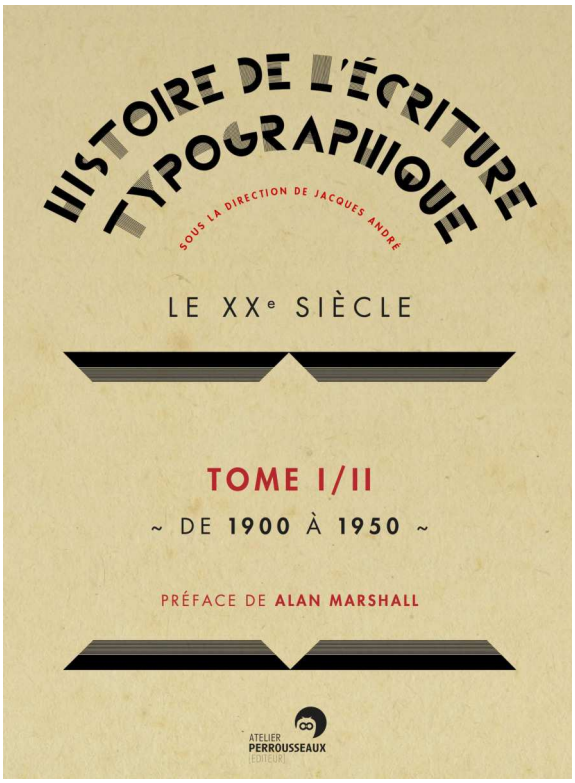
[5] musarithmia. What is the name of LaTeX's default style and why was it chosen for LaTeX? — (answer). *tex.stackexchange.com*, 272607(1), Oct 2015. `http://tex.stackexchange.com/questions/272607`.

◇ Peter Flynn
Textual Therapy Division,
Silmaril Consultants
Cork, Ireland
Phone: +353 86 824 5333
`peter (at) silmaril dot ie`
`http://blogs.silmaril.ie/peter`

Review and summaries: *The History of Typographic Writing — The 20th century* Volume 1, from 1900 to 1950

Charles Bigelow

Histoire de l'Écriture Typographique — le XXième siècle; tome I/II, de 1900 à 1950. Jacques André, editorial direction. Atelier Perrousseaux, Gap, France, 2016, `http://tinyurl.com/ja-xxieme`. The book is in French. Volume 2 covers the years 1950 to 2000, to be reviewed later.



The 20th century saw the three most transformative innovations in typographic technology after Gutenberg's invention of alphabetic movable type five centuries earlier. Each innovation by turns increased speed, reduced cost, and increased efficiency of text composition. First came keyboard-driven hot-metal composing machines like Linotype and Monotype; invented in the late 19th century, these achieved commercial dominance in the early decades of the 20th century, supplanting most hand-set type. Next came phototype — electro-optical photographic composition; invented in the 1940s, phototype replaced most hot-metal typesetting by the 1970s. Lastly came digital typography; invented in the 1960s, it replaced most metal and photo typesetting by the year 2000. Typesetting occurs prior to print, so these technological changes went mostly unnoticed by readers. By the end of the millennium, however, digital typography had begun to supplant print itself, because text display and reading increasingly shifted from paper to computer screen, a phenomenon now noticed by nearly all readers and publishers.

In the 20th century, typography was also transformed by cultural innovations that were strikingly visible to readers. In a profusion of new styles, movements, and polemics, a plethora of avowedly revolutionary "-isms" challenged traditional tenets of typography in zealous efforts to reformulate, abandon, or replace long-held principles of typographic organization and expression.

Some of these cultural movements hearkened back to an idealized typographic past, while others pointed to an idealized future. Our typography today is a mix of such memories versus desires: old and new, traditional and modern, potential and practical, obsolete and avant-garde.

There have been relatively few books on typography that provide deep analysis of its cultural transformations, knowledgeable explanations of its technological progression, and copious illustrations to accompany both aspects.

That is why this book is a milestone in the scholarship and appreciation of modern typography. Totaling 522 well-illustrated pages in two volumes of essays by a group of typographic experts under the editorial direction of Jacques André [1], the books provide an impressive perspective on the typographic art, culture and technology of the past century. [2]

Instead of an overview of the whole book, this review of the first volume gives partial summaries and comments on each of the chapters, This is done for two reasons. First, the book is in French, so the short, mini-summaries may help English-language readers get some idea of the contents and significance of those various chapters that may be of particular interest. Second, because each author writes with different expertise, perspective, and literary style, these mini-summaries may give some hint of the variety of styles and sensibilities in the essays. The book deserves an English edition, but it is hoped that these notes may at least point to what is contained therein.

Here are the chapters of Volume 1.

Alan Marshall: Preface (*Préface*)
The 20th century was one of the most eventful periods in the history of typography, influenced by two major changes, the mechanization of typesetting and the diversification of the use of print. The former advanced typography for *reading*, while the latter transformed typography for *seeing*. The selection and production of typefaces for mechanized composition concentrated mainly on traditional type forms that

are nearly subliminal in extended texts, but for promotion of goods and dissemination of publicity in the increasingly mass-market 20th century, super-liminal type styles were created to arrest, shock, intrigue, seduce, and persuade readers of short commercial messages. As technology and usage changed, traditional methods did not entirely disappear; hand-set display types, for example, were often used in conjunction with mechanized composition of longer texts.

1. Matthieu Cortat: The flowering of the Modern (*La floraison de la modernité*)
From beginnings in the late 19th century English Arts & Crafts movement and French lithographic poster lettering, new typographic styles emerged, especially the flowing, floral, and youthful styles gathered under the banner of Art Nouveau, which exercised international influence on lettering and typography, often in contrast to traditional typographies of France and Germany. The botanical style of Art Nouveau is a recurrent theme in this essay, which ends with an olfactory metaphor: that when the avant-garde moved on to other styles, Art Nouveau typography faded into obsolescence, leaving only occasional whiffs of a flowery perfume in its wake. (This reviewer recommends a whiff of Jacques Guerlain's "Après l'Ondée" (After the Rain Shower), the quintessential Art Nouveau perfume, composed in 1906 but still in production and regarded as one of the greatest fragrances of all time.)

2. Roxanne Jubert: Signs of the avant-garde: the alphabet between construction, system, art and utopia (*Signes des avant-garde: l'alphabet entre construction, système, art et utopie*)
This lucid yet congenial exposition, reminiscent of the essays of Roland Barthes, analyses the explosion of diverse avant-garde movements in Europe, including Futurism, De Stijl, Dadaism, Constructivism, Bauhaus, New Typography, and their effects on typographic forms and organizations. The structuralist approach of the essay effectively elucidates implicit (and sometimes explicit) aesthetic and semiological philosophies of the avant-gardists with their arrestingly visual modularization and segmentation of typographic images, their construction of experimental alphabets, and their integration of typography, geometry, and photography. Even the best English discussions of avant-garde typography are rarely this interesting.

3. Roxanne Jubert: The Art Deco Letter: variety, stylization, play, and contrast (*La lettre Art Déco: variété, stylisation, jeu, contraste*)
Art Deco was (and still is) an aesthetic family that encompassed several different but somehow related visual styles. Understanding Art Deco visual relationships is a bit like grappling with Wittgenstein's remarks on family resemblance — is there a single core element or a set of overlapping similarities? Whatever its visual core, Art Deco influenced poster art, architecture, signage, advertising, and typography and characterized the then-modern era of the 1920s and 1930s, between the two great wars. Despite diversity within the style, multiple connections can be traced among its apparently disparate forms. Of particular typographic note are Art Deco typefaces by designers who are better known today for more sober creations, including Morris Fuller Benton, Rudolf Koch, Imre Reiner, Robert Middleton, and Dick Dooijes. There are also notable designs by artists firmly within the Art Deco genre such as A. M. Cassandre. Unlike the Art Nouveau types, many forms of Art Deco continue to be used today.

Jacques André: First Interlude: The sociology and revival of a type style: stencil (*Pause: Sociologie et renouveau d'un caractère: les pochoirs*)
This absorbing, often amusing, and copiously illustrated exposition proceeds from a hand silhouette in palaeolithic cave painting to the analytic logic of form and counter-form in letter shapes; from spray-painted graffiti to labels on gunny sacks and letters on wine barrels; from slogans on walls to road markers; from signs cut in metal to stencil-like typefaces by Auriol, Jacno, and other 20th century type designers.

4. Nelly Gable and Christian Paput: Perennity of punches and matrices (*Pérennité des poinçons et matrices*)
A clear and beautifully illustrated treatise on type punch-cutting, emphasizing the tools and techniques still used today at the French Imprimerie Nationale. For five and a half centuries [3], punch-cutting has been at the core of every era of typography, practiced by a tiny group of skilled artists whose exquisitely precise work has rarely received public recognition, first because it was necessarily executed in miniature (like the 10 point type you are reading now) and was usually anonymous (apart from in-group knowledge of a few typographic cognoscenti) because type was made in service of the arts of literature and knowledge. Stanley Morison wrote that typography is only accidentally aesthetic, "for the enjoyment of patterns is rarely the reader's chief aim." This applies as well to the jewel-like intricacy of finished punches. In this chapter, the methods, tools, and techniques of cutting and proofing punches, and of striking and

justifying matrices, are precisely described and explained, coupled with clear photographs by Nelly Gable and Daniel Pype. Among its side-revelations are the names of the principal French punch-cutters who worked at the Imprimerie Nationale, Deberny & Peignot, and other organizations in the 20th century. Thus, artists who worked mostly in obscurity are rescued to some extent from anonymity. A photo of punches bearing the punch-cutters' own identifying stamps furthers the cause. Little has been published on punch-cutting, and even less is still in print, so this excellent chapter especially merits an English translation and republication, perhaps as an offprint. [4]

5. Christian Laucou: Technical innovations from 1900 to 1945 (*Les innovations techniques, de 1900 à 1945*)
In the Internet era, we may believe that electronic innovation travels at a faster speed than ever before, but this essay demonstrates the dizzying pace of mechanical innovation in typesetting at the start of the 20th century. Typography was, after all, the dominant information technology of that era. Even the lexicon of names of typesetting inventions is enough to write rhyming poetry to accompany the clatter of machines like Barotype, Diotype, Francotype, Intertype, Linotype, Monotype, Nebitype, Rototype, Stringertype, Teletype, Typar, Typograph, Typomeca, Typostereotype and more. Engineering drawings, particularly of matrices and mechanisms, illustrate the ingenuity devoted to turning keystrokes into print, a process that continued to be reinvented throughout the century in other technologies, even as metal-based composition approached near-extinction.

Jacques André: Second Interlude: Louis Jou, an idiosyncratic brilliance (*Louis Jou, un marginal génial*)
An appreciation of the typographic work of Louis Jou, an engraver, typographer, type designer and fine book printer-publisher whose work combined the richness, variety, and elegance of Renaissance typography with the exuberance of Art Nouveau, enhanced by his own inventiveness in ornamental lettering and layout. A friend of Apollinaire, Dufy, Cocteau, and other literary and artistic luminaries of his time, Jou, who was born in 1881 and died in 1968, "came too late to achieve the glory of his English equivalent, William Morris but too early to use digital type, which would have enabled him to play more with fonts and compose his books with the perfectionism of an aesthete."

6. Manuel Sesma: Return to historical and neo-historical typefaces (*Retour aux caractères historiques et néo-historiques*)
Revival and practice of past letterforms is traditional in Asian calligraphy. In the Italian Renaissance, Humanist handwriting revived Carolingian court handwriting of six centuries earlier and became the model for the first roman and italic types. Arts and Crafts printers revived older types using photographic enlargements from books and hand punch-cutting. Augmented by pantographic punch-engraving, revivals achieved commercial success in the 20th century, especially with revivals of types cut by Claude Garamond in the 16th century (or derivatives cut by Jean Jannon in the 17th century but misidentified as those of Garamond). These revivals were marketed under the names "Garamond," "Garamont," "Granjon," "Estienne," "Sabon," and others, causing this essay to call the phenomenon "garamonomanie" (perhaps "Garamonomania" in English). Questions about whose types were actually revived as Garamond's resulted in intriguing typographic scholarship by Jean Paillard in 1914 and Beatrice Warde (writing as Paul Beaujon) in 1926. [5]

The Peignot foundry types "Cochin" and "Nicolas Cochin" were based on elegant lettering by 18th century engravers Charles-Nicolas Cochin and son. The Peignot Cochin types became fashionably popular in France and were imitated by foundries elsewhere.

In the U.S., American Type Founders (ATF) produced successful revivals by designer Morris Fuller Benton and printer-scholar Henry Lewis Bullen, including Bodoni, Jenson (called Cloister), (Fry's) Baskerville, Caslon, and the Garamond that ignited Garamonomania. Frederic W. Goudy often took Renaissance models as inspiration but imbued them with his own artistic sensibility. Goudy Old Style from ATF, with additional versions by M. F. Benton, has remained popular through every major change in typesetting technology. Goudy also drew "Garamont" for American Monotype.

Revivals were also produced by the Bauer and Stempel foundries in Germany and Linotype in England, Germany, and the U.S. In mechanical composition, the best known series of revivals came from the English Monotype corporation. Under the direction of Frank Hinman Pierpont and with the advisement of Stanley Morison, Monotype revived types by Jenson, Aldus, Arrighi, Garamond, Van Dijck, Fournier, Baskerville, Bell, Bodoni, and others, and also produced original faces by designers with classical affinities: Perpetua by Eric Gill, Romulus by Jan van Krimpen, and Dante by Giovanni Mardersteig.

Though waxing and waning at times, type revivals continued through the rest of the 20th century.

7. Manuel Sesma: Lead again
(*Encore le plomb*)
As the European typographic industry strove to recover after World War II, an exuberant flowering of imaginative typefaces emerged from French designers and typefoundries with an inventive sense of graphic style termed "La Graphie Latine" ("Latin Typography"). These spirited French typefaces brimmed with inspiration: Paris, Flash, Île de France and Champs Elysées by Enric Crous-Vidal; Choc, Banco, Mistral, Calypso, and Antique Olive by Roger Excoffon; Jacno by Marcel Jacno; Ondine and Phoebus by Adrian Frutiger. Though revolutionary in style, these faces were produced as lead foundry types. [6] Mistral and Calypso were "tours de force," challenging the constraints of metal type. In the 1950s and 1960s, expressive French designs differed markedly from the sober, grotesque-style sans-serifs at the core of Swiss typography and its allied international modernism that favored grid-based bureaucratic regulation over charismatic expression. But, as a philosophical complement to effervescence in letter design, French writing and thinking on type also featured acutely rational reflections on the logic of typography, as seen in the typeface classification system devised by Maximilien Vox and adopted as a standard by the Association Typographique Internationale as the Vox-ATypI system, and in the numeric naming system for typeface weight, width, and posture devised by Adrian Frutiger for his pioneering Univers neogrotesque produced by the Paris foundry Deberny & Peignot, for both phototype and foundry type.

8. Charles Bigelow: Legibility and typography: research in the first half of the 20th century (*Lisibilité et typographie: les recherches durant la première moitié du xxe siècle*)
By the first decade of the 20th century, literacy rates in France, England, Germany, and America had soared to more than 90 percent due to national expansions of free, public education. The vast increases in literacy fueled the printing and marketing industries but raised concerns about typeface legibility in reading education, ocular health of children, and the physiology and psychology of reading. Émile Javal in France and Edmund Burke Huey in America pioneered reading research. Shortly afterwards in New York, Barbara Roethlein (with font assistance from Morris Fuller Benton) conducted one of the earliest psychological studies comparing type legibility. Elsewhere, Richard L. Pyke in England, Gerrit Willem Ovink in Holland, and Miles Tinker with Donald G. Paterson in Minnesota conducted legibility studies, the last of these continuing through the first half of the century. Although most typeface development followed traditional faith in the trained eye of the designer, legibility research did influence the design of a few popular typefaces for specific purposes. The enduringly popular Century Schoolbook, originally designed by Morris Fuller Benton for a textbook publisher, drew upon Roethlein's earlier research. Linotype's "Legibility Group" was at one time used in more than half of all newspapers in the United States. It included several closely related designs (Ionic No. 3, Excelsior, Opticon, and Corona) that were influenced by Century Schoolbook and unpublished legibility research by Linotype. Not all legibility studies were reliable. R. L. Pyke skeptically remarked, "Four times as many writers have measured legibility as have defined it. Three out of every four writers have been attempting to measure something the exact nature of which they have not paused to examine." [7]

Paul-Marie Grinevald: Third Interlude: Survey of historians of typography
(*Aperçu des historiens de la typographie*)
This is a rare essay in typographic historiography: a history of histories of typography. It includes social histories of printing such as Elizabeth Eisenstein's *The Printing Press as an Agent of Change*, Lucien Febvre & Henri-Jean Martin's *The Coming of the Book*, Marshall McLuhan's *The Gutenberg Galaxy* and *The Medium is the Message*. Recognizing that typography is only the most recent form of writing, the chapter cites Jack Goody's anthropological treatises on writing and society, ancient and modern, in *The Domestication of the Savage Mind* (the title an in-joke on a structuralist treatise by French ethnologist Claude Levi-Strauss) and *The Logic of Writing and the Organization of Society*.

On the forms of letters and typefaces, this survey touches on Euclidean letter constructions by Luca Pacioli, Albrecht Dürer, Geoffrey Tory, and others in the 16th century, on the refined Cartesian geometry of type forms developed circa 1693–1702 by savants Jean-Paul Bignon, Jacques Jaugeon, and Sebastian Truchet, which led to the creation of the Roman du Roi, the French royal typefaces. Valuable essays on type and history, to mention a few of the many cited, include: in the 18th century, Pierre-Simon Fournier's *Manuel Typographique* and the chapter "Caractère" in the Diderot *Encyclopédie*; in the 19th century, Talbot Baines Reed's *A History of the Old English Letter Foundries*; in the 20th century, Daniel Berkeley Updike's *Printing Types: Their History,*

Forms and Use — A study in survivals, Marius Audin's *Histoire de l'Imprimerie par l'Image* and *La Somme Typographique*, Harry Carter's *A View of Early Typography*, Fernand Baudin's *L'Effet Gutenberg*, John Dreyfus' *Into Print*, Alan Marshall's *Du plomb à la lumière*, and other recent works. Preceding volumes from Perrousseaux in the series on *Histoire de l'Écriture: Typographique* must also be mentioned: *De Gutenberg au XVIIe siècle*, by Yves Perrousseaux; *Le XVIIIe siècle* (two volumes), also by Yves Perrousseaux; and *Le XIXe siècle français* by Jacques André and Christian Laucou.

The book ends with an extensive general bibliography as well as bibliographies specific to each chapter, totaling 412 references in all, to works in French, English and German. These are followed by indexes of typefaces, typographers, and typographic topics.

Notes

[1] The general editor, Jacques André is a French computer scientist with an intense interest in typography. He organized the first academic conference on the integration of computer science with typography, "La Manipulation des Documents", in Rennes, France in May, 1983, and organized the later international "Raster Imaging and Digital Typography" (RIDT) conferences. He has published papers on TEX, and readers of *TUGboat* may know his reviews and articles including "Father Truchet, the typographic point, the Romain de Roi, and tilings" (*TUGboat* issue 20:1, 1999) and "The Cassetin project — Towards an inventory of ancient types and the related standardised encoding" (24:3, 2003). Notable are his translations and re-interpretations of selected print works, such as "Petits jeux avec des ornements", a translation into French with digital re-composition of ornamental patterns by Swiss typographer Max Calisch in *Kleines Spiel mit Ornamenten* (Berne, 1965), and an electronic revival of P.-S. Fournier's *Manuel Typographique* (Barbou, 1764). `http://jacques-andre.fr/japublis/`

[2] These two volumes are the latest in a series on "Typographic Writing" from Atelier Perrousseaux, a French publisher of typography books. The series, in several successive volumes, begins with Gutenberg and thence covers the next six centuries. The term "typographic writing" in the title affirms that typography is the latest form of writing in a long history of literacy. All of the books in the series are well worthwhile.

[3] Gutenberg's method of making type in the 1450s remains shrouded in mystery, but the technique of punch and matrix was probably developed and practiced before 1470, when Nicolas Jenson, a master of the French mint who had studied the infant art of typography in Mainz, gave up minting for printing and opened his shop in Venice.

[4] The chapter's bibliography includes *Counterpunch* by Fred Smeijers, now out of print but an entertaining and informative book that combines the author's efforts to learn punch-cutting, including a history and explanation of the tools and techniques, comparisons of hand work to computer work, the creating of new type designs inspired by models from the golden age of typography, and comments on other topics arising during the author's story. A good short essay, not in the bibliography, is Paul Koch's "The Making of Printing Types", translated from German by Otto W. Fuhrmann, in *The Dolphin: A Journal of the Making of Books*, No. 1, pp. 24–57. Illustrations by Fritz Kredel. The Limited Editions Club, New York, 1933.

[5] See: 'Paul Beaujon' (Beatrice Warde), 'The Garamond types: 16th and 17th century sources considered', *The Fleuron*, 5, 1926, pp. 131–79. A recent review of Garamond scholarship is James Mosley's "Garamond or Garamont", Typefoundry Blog, Apr. 1, 2011. `http://typefoundry.blogspot.com/2011/04/garamond-or-garamont.html`

[6] The title of this chapter and the bravado of the designers may remind fans of American western movies of a memorable line in the classic *The Magnificent Seven*, delivered by Steve McQueen: "We deal in lead, friend."

[7] A short talk on this chapter was delivered at the TEX Users Group meeting in Toronto, Ontario, on July 27, 2016. Legibility research lost academic popularity in the middle of the century, but scientific reading research was revived in the 1970s and 1980s, principally in three areas: eye movements in reading, researched by G.W. McConkie, Keith Rayner, and Andrew Pollatsek, among several others; psychophysics of reading, researched by Gordon E. Legge and others; various practical studies comparing typefaces, by Cyril Burt, Bror Zachrisson, E.C. Poulton, Dirk Wendt, Herbert Spencer, Linda Reynolds, and others.

Disclosure: As seen above, the reviewer contributed a chapter to this book (but receives no monetary compensation). He has known the general editor over more than three decades of friendly mutual interest in digital typography.

◇ Charles Bigelow
`http://lucidafonts.com`

SILE: A new typesetting system

Simon Cozens

Abstract

SILE is a new typesetting system, influenced by TEX but written from scratch in Lua. While still in the early stages of development, it holds potential as a typesetting system designed for unsupervised automated typesetting, especially in non-Latin scripts. SILE can be obtained from `http://www.sile-typesetter.org/`.

1 Introduction

In 2012, I wrote a typesetting system by mistake.

As part of my work for a small publishing company, I wrote a simple Perl script to automate the production of book covers. However, I soon discovered that the typesetting of the back cover blurb was unacceptable without proper justification. I ported Bram Stein's JavaScript version [8] of the original TEX justification algorithm [5] to Perl. Since there was already a Perl implementation [4] of TEX's hyphenation algorithm [6], I added support for hyphenation at the same time.

Now I had something which could reliably typeset paragraphs to PDF ... well, you can probably guess the rest. Adding a page builder was the obvious next step, and soon penalties, skips, glues and the rest followed. The project was rewritten in JavaScript, and then finally in Lua.

Why does the world need another typesetting system? Of course, it doesn't. But sometimes it's a good idea to reinvent the wheel; that's how we get better wheels. If we never reinvented wheels in the software industry, this journal would be called *troffboat*. And a friend who works in Bible typesetting let me know about a number of things that current automated typesetters can't do well — column balancing with multi-page lookbehind *and* grid typesetting; layout of parallel polyglots across page spreads; and so on — which gave me a number of goals.

Because of these goals and my own interest in non-Latin scripts, SILE has developed a focus on multilingual typesetting, particularly with complex and minority scripts, and the unsupervised layout of large, complex documents. SILE will see a 1.0.0 release when it is capable of taking a Unified Scripture XML [7] Bible translation and an appropriate class file, and producing a print-ready Bible of quality equivalent to that of a human typesetter. Even if I never achieve it, I'm having fun trying.

2 SILE's Component Parts

One of the advantages of writing a typesetting system in 2012 rather than in 1982 is that most of the hard work is already done for you. As we have mentioned, core typesetting algorithms are readily available; Unicode, together with its standard annexes and technical reports, describes good solutions to many of the problems of multilingual data representation; OpenType fonts and shaping engines help with the layout of complex scripts; embedded, interpreted languages won out over macro processors; and the world has effectively standardised on PDF as a document format.

A bird's eye view of SILE is shown in fig. 1. Text is consumed, and is reordered according to the Unicode Bidirectional Algorithm [9]. Then each run of text, together with its font, language, direction and other settings, is passed to the HarfBuzz [1] shaping engine. HarfBuzz returns a stream of glyph IDs and metrics, which are then assembled into a list of nodes, either by language-specific processors or by the default Unicode processor. The nodes are fed to the familiar H&J algorithms and collected into vboxes, vboxes into frames, frames into pages, and pages are finally output as PDF.

The choice of Lua as an implementation language hinged on a number of factors; obviously there are some benefits to using a language which is familiar to a pre-existing community of typesetting software engineers, although I have no strong desire to 'convert' anyone! But there are also benefits to using an interpreted language for implementation: first,

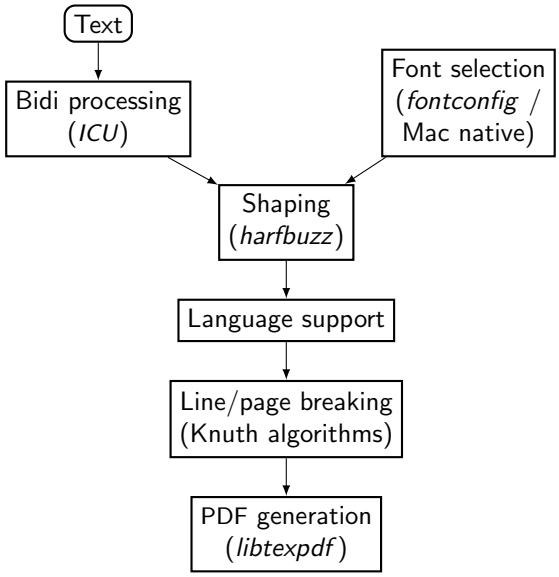


Figure 1: SILE's component parts

```
\SILE{}.registerCommand("tableofcontents:item", function (options, content)
  \SILE{}.settings.temporarily(function ()
    \SILE{}.settings.set("typesetter.parfillskip", \SILE{}.nodefactory.zeroGlue)
    \SILE{}.call("tableofcontents:level" .. options.level .. "item", {}, function ()
      \SILE{}.process(content)
      \SILE{}.call("dotfill")
      \SILE{}.typesetter:typeset(options.pageno)
    end)
  end)
end)
```

Figure 2: Lua code to typeset a TOC entry

Lua is designed as an embedded language, which means that SILE can provide a complex text layout system for embedding within other applications. (For instance, there is a SILE preview plugin for the Glyphs font editor.) It also means that any area of SILE's operation can be overridden or extended, not just those with pre-defined hooks. For instance, the grid typesetting package works by overriding the leading calculation; similarly, when setting Japanese text on a *hanmen* grid, there is no need to apply full best-fit paragraph composition — it's fine to replace the Knuth-Plass algorithm with a simple first-fit line breaker for speed.

SILE's modular design also means that everything is pluggable. I tried a number of different PDF libraries while developing SILE; the first versions used Cairo [2], but Cairo's PDF surface is fairly limited, and does not allow for the generation of PDF annotations, links and outlines — not to mention any of the tagged and structured markup required for accessible PDFs — so I started looking for alternatives: Haru, PoDoFo and others. Since the output system has a well-defined interface, I could easily test a new PDF generation library by slotting a new output implementation in place. Similarly, SILE's regression test system works by plugging in a custom outputter which produces a textual representation instead of a PDF; this can then be compared against the expected results using *diff*.

Incidentally, the PDF library I settled on was both an old one and a new one: I extracted the PDF generation backend from *dvipdfmx* and made it available as a library–*libtexpdf*. This was the only PDF generation system I could find which allowed me to address glyphs by ID, and also to add arbitrary PDF operators to the output.

Apart from C interfaces to HarfBuzz, ICU (the Unicode support library), *fontconfig* and *libtexpdf*, the core of SILE comprises a little under 5,000 lines of Lua code. (10% of which is made up of a somewhat literal port of TEX's line breaking algorithm.) This makes sense — with so much done by third-party libraries, there is relatively little left for SILE to do by itself.

3 Input formats, packages and classes

Just like the output system, SILE's input system is modular. The first input format implemented for SILE was XML — the idea being that SILE is to be used to typeset data produced by other software, such as translation databases, rather than documents constructed by hand; XML is both an easy format to parse and an easy format for other software to output. But while SILE needs to ingest XML, for whatever reason people wanted to hand-generate SILE documents, and so SILE added a parser for a simple, TEX-like input format.

The TEX-like format is only superficially TEX-like. It is, essentially, simply another way of representing an XML tree structure. These two SILE documents are equivalent:

```
\begin{foo}
Text
\bar[this=that]
\end{foo}
```

```
<foo>
  Text
  <bar this="that"/>
</foo>
```

The implementation of `<foo>` and `<bar>` is, of course, up to the user. In this sense, SILE is similar to an XML stylesheet processor: alongside a document must come a set of processing expectations which define how the tags will be typeset. SILE's `\define` command provides an extremely restricted macro system for implementing simple tags, but you are deliberately forced to write anything more complex in Lua. (Maxim: Programming tasks should be done in programming languages!) For example, the command to typeset a table-of-contents item is implemented by the code in fig. 2. This expects a command of the form:

```
\tableofcontents:item[level=2,pageno=3]
  {Something}
```

and passes the text and page number separated by leaders to the command which styles a level 2 TOC entry; this command, which is more easily implemented with a `\define` at the SILE level, will in turn set the appropriate font size, style and so on.

Lua code is loaded into SILE as packages or classes, similar to LaTeX — classes define the layout and key formatting expectations for tags, while packages provide additional functionality. Classes can be inherited (in the object-oriented programming sense) from other classes; SILE comes with a number of basic document classes but the expectation would be that each substantial document project would define its own class.

Since classes can be loaded even before the document is opened, they can do things such as providing a new input format. The `markdown` class does just this, implementing a parser and providing processing expectations for Markdown documents.

Naturally there are not currently anything like as many packages for SILE as for TeX derivatives. But fig. 3 is (an abridged version of) my favourite. This implements boustrophedon text by overriding the typesetter's function for turning horizontal lists into vertical lists. After the default implementation, the vertical list is inspected, and a custom whatsit (`swap`) is inserted after every vbox. When the whatsit is output, the typesetter's direction is reversed: if the previous line was left-to-right, the next line will be right-to-left, and *vice versa*.

SILE's programmability leads itself to experimentation and implementation of new technologies; support for OpenType color fonts was added as an external package in 85 lines of code, and rudimentary support for OT fonts with SVG outlines has recently been added.

4 The language support system

While Harfbuzz and Unicode provides a lot of what SILE needs to support complex scripts, different languages have different typographic conventions. For instance, correctly typesetting Japanese is not a matter of inserting line break opportunities between every pair of characters; Japanese *kinsoku-shori* rules stipulate that some punctuation characters cannot start lines and others cannot end lines. Additionally, characters are generally set on a fixed grid, but spacing is reduced around brackets and commas. These language-specific typesetting conventions are encoded in SILE's language support system, which assembles the stream of glyphs from the shaper into nodes, giving SILE a chance to implement hyphenation points, line breaking opportunities and so on.

```
local swap = \SILE{}.nodefactory.newVbox({})
swap.outputYourself = function(self,typesetter)
  typesetter.frame.direction =
    typesetter.frame.direction == "LTR-TTB"
      and "RTL-TTB" or "LTR-TTB"
  typesetter.frame:newLine()
end

\SILE{}.typesetter.boxUpNodes = function(self)
  local vboxlist =
    \SILE{}.defaultTypesetter.boxUpNodes(self)
  local nl = {}
  for i=1,#vboxlist do
    nl[#nl+1] = vboxlist[i]
    if nl[#nl]:isVbox() then
      nl[#nl+1] = swap
    end
  end
  return nl
end
```

Figure 3: The `boustrophedon` package, abridged.

Another pertinent example is that of many southeast Asian languages which are written without interword spaces but which line break between graphical syllable clusters, the clusters being determined by morphological analysis. SILE's support for Javanese uses a Parsing Expression Grammar [3] to detect syllable boundaries and insert penalties into the node stream to specify potential break points. Access to ICU means that language-specific casing rules (such as the Turkish i/İ and ı/I combinations) are correctly applied.

SILE does not assume any default directionality, meaning that left-to-right typesetting is not privileged over right-to-left processing. Indeed, supporting Mongolian, which is traditionally written top-to-bottom and left-to-right, is simply a matter of telling the typesetter about the new direction: `\thisframedirection{TTB-LTR}`.

Figure 4 demonstrates SILE's multi-script capabilities; notice how SILE has respected the typographic conventions of each script, and how the RTL texts (Arabic and Hebrew) have been reordered according to the conventions of mixed directionality typesetting. In the source file, each text is marked up with its language so that SILE can select the appropriate set of rules, but the bidi reordering is performed by default and requires no additional markup.

5 Frames

In our overview of SILE's component parts, we mentioned in passing that vboxes are assembled into frames and frames are assembled into pages. Frames

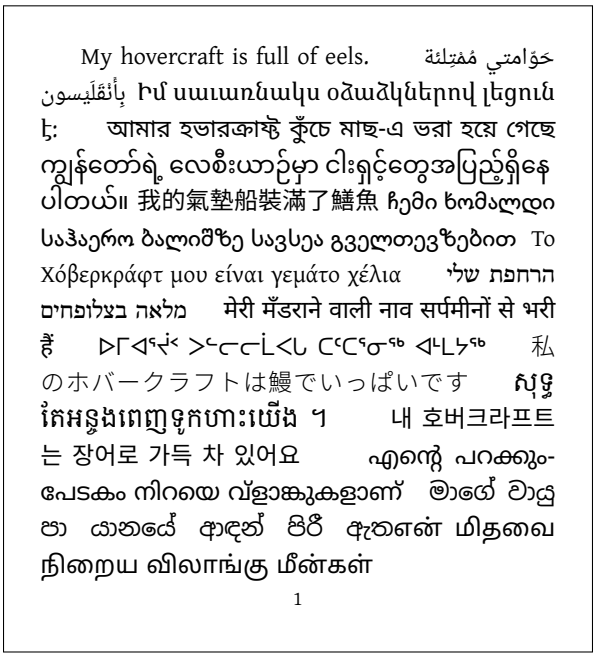


Figure 4: SILE's multi-script capabilities

are areas on the page where text is to be set. The frames for a page are generally defined by the document's class, but can be modified dynamically. For instance, footnotes are generally implemented by placing a zero-height frame at the bottom of the main content area. As footnotes are placed on a page, the footnote frame expands and the content frame shrinks. This allows for interesting layout possibilities: in a two column layout, the footnotes can be placed at the end of the second frame, or across the bottom of two frames, or in a separate area of the page altogether.

Frame layouts are generally specified relatively rather than absolutely; for instance, fig. 5 shows the frame declarations of the standard book class. The dimensions `%pw` and `%ph` refer to percentage of page width and page height respectively.

6 Various neat hacks

SILE packages implement a number of interesting ideas; in no particular order: best-fit page breaking; access to OpenType font features; parallel polyglot layout (see fig. 6); justification alternates (rewriting the text to improve justification); font fallback on missing glyphs; grid typesetting; Japanese vertical typesetting and ruby (*furigana*) support; automatic generation of font specimens; a BibTeX-like bibliography manager implemented at runtime within the typesetter; support for producing Structured and Tagged PDFs.

7 Challenges remaining

While SILE has been used successfully to produce a number of print books and articles, it is still not a finished product. The development team is very small; I'm the primary developer, with Caleb Maclennan and Khaled Hosny as notable contributors. Development happens on github and we are very open to issues and pull requests. However, there are a number of challenges remaining.

The lack of a Windows distribution is currently hampering adoption; the pushback routine (which disgorges the vertical list back onto the horizontal list after every frame break, in case the next frame has a different width) is a perennial source of mind-boggling bugs; and currently we lack a good solution for moving between multi-column and single-column layout and back again, which is a blocker for serious Bible typesetting.

We are also still working out how best to use SILE. For instance, I initially anticipated that the frames feature could be used to solve all kinds of layout challenges — drop caps, wrapping text around floated figures and so on. After some investigation we have found that frames are better suited to page-level layout, and other solutions such as nested vboxes and custom packages work better for altering layout within a frame. As more packages are developed, idiomatic use of the system will become more clearly defined.

I am often asked about typesetting of mathematics. My usual glib answer is that there is not very much mathematics in the Bible! However, I would like to see an implementation of math typesetting for SILE. The current plan is to find a way to call out to MathJax to perform the layout computations, and have SILE lay out the resulting nodes. However, SILE's development is essentially driven by user requests; I don't need math for the kind of books I'm typesetting, but if this is something you need, I would be glad to help you implement it!

8 Conclusion

I have deliberately avoided making comparisons between SILE and TeX derivatives in this article, attempting to introduce SILE on its own merits. In a sense there is no comparison. TeX is an incredibly mature and stable code base with a large and vibrant community; SILE is new, fast-moving and buggy, with few active developers. But I believe that, with time and development, SILE has the potential to provide better output than TeX for complex automated layout of non-Latin documents. It's also really fun to mess about with!

```
content = {
  left = "8.3%pw", right = "86%pw",
  top = "11.6%ph", bottom = "top(footnotes)"
},
runningHead = {
  left = "left(content)", right = "right(content)",
  top = "top(content)-8%ph", bottom = "top(content)-3%ph"
},
footnotes = {
  left = "left(content)", right = "right(content)",
  height = "0", bottom = "83.3%ph"
},
folio = {
  left = "left(content)", right = "right(content)",
  top = "bottom(footnotes)+3%ph",
  bottom = "bottom(footnotes)+5%ph"
}
```

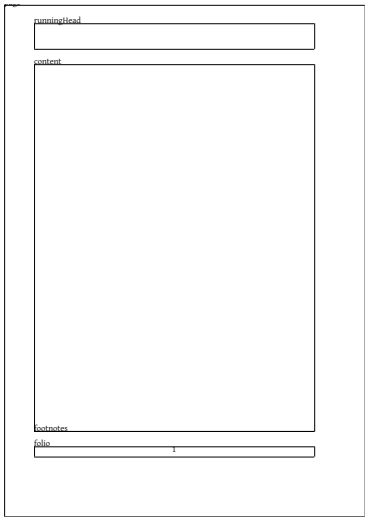
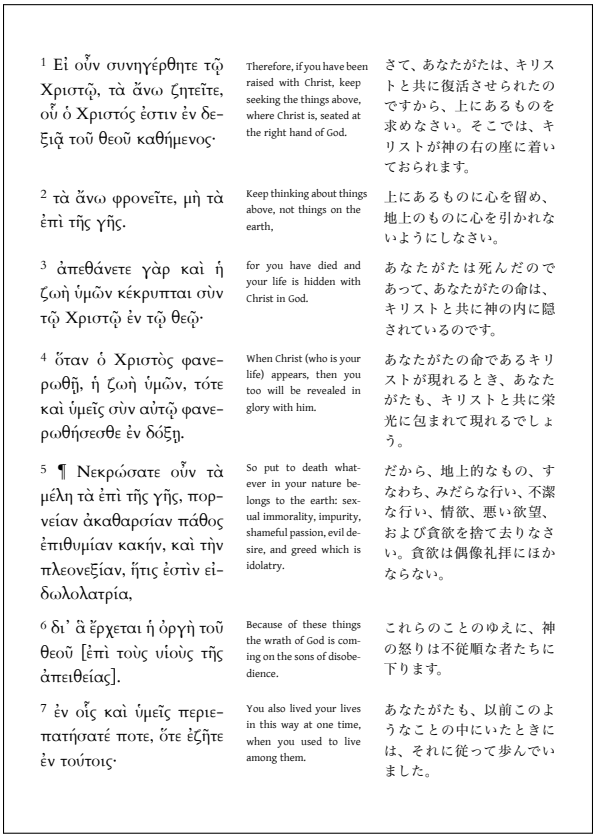


Figure 5: Frame layout in the standard book class



[1] Εἰ οὖν συνηγέρθητε τῷ Χριστῷ, τὰ ἄνω ζητεῖτε, οὗ ὁ Χριστός ἐστιν ἐν δεξιᾷ τοῦ θεοῦ καθήμενος·	Therefore, if you have been raised with Christ, keep seeking the things above, where Christ is, seated at the right hand of God.	さて、あなたがたは、キリストと共に復活させられたのですから、上にあるものを求めなさい。そこでは、キリストが神の右の座に着いておられます。
[2] τὰ ἄνω φρονεῖτε, μὴ τὰ ἐπὶ τῆς γῆς.	Keep thinking about things above, not things on the earth,	上にあるものに心を留め、地上のものに心を引かれないようにしなさい。
[3] ἀπεθάνετε γὰρ καὶ ἡ ζωὴ ὑμῶν κέκρυπται σὺν τῷ Χριστῷ ἐν τῷ θεῷ·	for you have died and your life is hidden with Christ in God.	あなたがたは死んだのであって、あなたがたの命は、キリストと共に神の内に隠されているのです。
[4] ὅταν ὁ Χριστὸς φανερωθῇ, ἡ ζωὴ ὑμῶν, τότε καὶ ὑμεῖς σὺν αὐτῷ φανερωθήσεσθε ἐν δόξῃ.	When Christ (who is your life) appears, then you too will be revealed in glory with him.	あなたがたの命であるキリストが現れるとき、あなたがたも、キリストと共に栄光に包まれて現れるでしょう。
[5] ¶ Νεκρώσατε οὖν τὰ μέλη τὰ ἐπὶ τῆς γῆς, πορνείαν ἀκαθαρσίαν πάθος ἐπιθυμίαν κακήν, καὶ τὴν πλεονεξίαν, ἥτις ἐστὶν εἰδωλολατρία,	So put to death whatever in your nature belongs to the earth: sexual immorality, impurity, shameful passion, evil desire, and greed which is idolatry.	だから、地上的なもの、すなわち、みだらな行い、不潔な行い、情欲、悪い欲望、および貪欲を捨て去りなさい。貪欲は偶像礼拝にほかならない。
[6] δι’ ἃ ἔρχεται ἡ ὀργὴ τοῦ θεοῦ [ἐπὶ τοὺς υἱοὺς τῆς ἀπειθείας].	Because of these things the wrath of God is coming on the sons of disobedience.	これらのことのゆえに、神の怒りは不従順な者たちに下ります。
[7] ἐν οἷς καὶ ὑμεῖς περιεπατήσατέ ποτε, ὅτε ἐζῆτε ἐν τούτοις·	You also lived your lives in this way at one time, when you used to live among them.	あなたがたも、以前このようなことの中にいたときには、それに従って歩んでいました。

Figure 6: A parallel triglot: different scripts, different column widths, different font sizes

References

[1] Behdad Esfahbod. HarfBuzz. http://harfbuzz.org/, June 2016.

[2] Bryce Harrington. PDF surfaces. https://www.cairographics.org/manual/cairo-PDF-Surfaces.html, April 2016.

[3] Roberto Ierusalimschy. Lua parsing expression grammars. http://www.inf.puc-rio.br/~roberto/lpeg/, September 2015.

[4] Alex Kapranoff. Text::Hyphen. http://search.cpan.org/perldoc?Text::Hyphen, October 2015.

[5] Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. *Software — Practice and Experience*, 11(11):1119–1184, 1981.

[6] Franklin Mark Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Department of Computer Science, Stanford University, 1983. http://tug.org/docs/liang/.

[7] United Bible Societies. Unified Scripture XML. https://ubsicap.github.io/usx/, 2016.

[8] Bram Stein. TEX line breaking algorithm in JavaScript. https://github.com/bramstein/typeset, April 2016.

[9] The Unicode Consortium. UAX #9: Unicode bidirectional algorithm. http://unicode.org/reports/tr9/, May 2016.

◇ Simon Cozens
Worldview Center for Intercultural Studies
St Leonards, Tasmania
Australia
simon (at) simon-cozens dot org
http://www.simon-cozens.org

BaskervilleF

Michael Sharpe

Abstract

BaskervilleF, where *F* may stand ambiguously for either Free or Fry's, is derived from *Libre Baskerville*, and attempts to move away from its web font design toward a font of more traditional Baskervillian appearance. Its model is the *Baskerville* from the American Type Foundry (ATF) in the early twentieth century, which in turn was loosely based on *Fry's Baskerville*, and had a new italic designed afresh by Morris Fuller Benton.

1 Baskerville's life and work

John Baskerville (1706–1775) was born to a poor family in Birmingham but his talents became apparent early. He received basic schooling and worked for a period as a writing master before turning to what made him quite wealthy, creating and operating a business that japanned furniture. (Japanning involved multiple layers of heat treated, highly polished lacquer. The process was developed as a European response to a similar Japanese metal lacquering process which had become very popular. Birmingham was the center of the industry in England. Success in the business at the upper end required meticulous attention to be paid to the fine detail of every aspect of the manufacturing process. We see remnants of the japanning method in older electrical transformers, whose windings bear layers of insulating shellac built up in a similar manner.)

In his mid-forties, Baskerville moved away from the japanning business back to his roots and started a firm devoted to high-quality printing. He was wealthy enough to not have to be concerned about making money from bread-and-butter jobs and could focus on high-end projects. At that time William Caslon was the dominant figure in English typography. His eponymous font was wildly popular both in England and America (U.S. Constitution!) but derivative, being based on other old-style fonts that had been available for many years, carried to some calligraphic extremes.

Baskerville's idea for a typeface that would reflect his ideals of stately, sober perfection was released in 1757. In contrast to the playfulness and asymmetries of old-style fonts, his designs relied on simplicity of form and a strict attention to detail. The characteristics, new to that era, were:

- very regular glyphs, with few embellishments;
- curved strokes that are close to circular, at least in the roman and bold;
- sharply cut serifs;
- the axis of rounded forms became almost vertical;
- high contrast (ratio of thickest to thinnest strokes).

To bring his ideas to fruition required making his own printing machinery, his own ultra-smooth, pressed paper and his own high quality ink. It is understandable that his business was not greatly successful as a commercial enterprise, and he stands as an exemplar of the aphorism about making a small fortune as a printer and type founder.[1] Following his first production of an edition of a work of Virgil, he was appointed (1758) as printer to the Cambridge University Press and in the course of the next fifteen years published about fifty classic volumes. His greatest publication is thought to be his folio Bible (1763), made according to his highest standards.

Baskerville seems to have been unusually progressive for his time, a free thinker though much of his work involved religious texts, and did not marry his housekeeper, and, finally, mistress and business partner, Sarah Eaves, until very late in life. (Zuzana Licko's font *Mrs. Eaves* is based on *Baskerville* but with softer features.)

At the time, the English printing community did not give Baskerville much credit for his innovations. His rivals spread stories about the sharpness of his serifs and the excessive contrast being a danger to the eyes and even with the then-primitive state of fake news and social networking, these views took a firm hold. Foreign printers, however, such as Benjamin Franklin, Giambattista Bodoni and Pierre Simon Fournier were highly impressed by his designs, which led to the development of the "modern" style, for which *Baskerville* is labelled a "transitional" font. Following Baskerville's death in 1775, his wife sold all his machinery, punches and matrices to Beaumarchais in Paris for £3700, having found no one in Britain willing to make an offer, such was the popularity of *Caslon*. These were used by Beaumarchais in his massive effort, requiring the purchase of three paper mills, to print in Germany the complete works of Voltaire, then banned from publication in France. The surviving punches and matrices were eventually donated back to Cambridge University in the early twentieth century. *Baskerville* has continued to be a French favorite, being used as the text font in many of their mainstream mathematical publications.

By 1765, Baskerville appears to have tired of the printing business and later abandoned it entirely. In 1768, a punch-cutter named Isaac Moore, who worked for a Bristol print foundry owned by Joseph Fry, cut an imitation of *Baskerville* that did not require the highest level of equipment, paper and ink. It is now considered that *Fry's Baskerville*, as it is now called, has more in common with later fonts such as *Bell*, *Bulmer* and *Scotch Roman*. In 1923, Morris Fuller Benton designed for

[1] It being essential to first possess a large fortune.

American Type Foundry (ATF) a revival of *Fry's Baskerville* with a new italic to replace the original poorly thought-of version in *Fry's Baskerville*. This version appears in several of their catalogs, which have been scanned at high resolution by Raph Levien and are available from TUG to interested parties.

In 2012, Pablo Impallari and Rodrigo Fuenzalida created a new web font, *Libre Baskerville*, that seems to be based on tracing the scans of the images above, as described in [KB01]. The x-height was increased, the ascenders were shortened, and the contrast was lowered substantially, while the serifs and tails were thickened considerably. Their font was well done technically (very good spacing and kerning) but did not look especially close to Baskerville, in my opinion.

2 BaskervilleF

In discussions with Karl Berry at the TUG meeting in Toronto, 2016, I expressed my interest in making a version of *Baskerville* more attuned to the interests of traditional TEX users, and Karl later supplied me with a wealth of information about doing this starting from Raph Levien's scans.[2] Remaking the fonts from scratch lacked appeal, and once I examined *Libre Baskerville* closely, I saw that much could be achieved by undoing their transformations to web fonts—reducing the x-heights, increasing the ascender heights, sharpening the serifs and hollowing out the glyphs to raise the contrast and reduce the heaviness. Wherever possible, I kept the side-bearings the same so that the new spacings and kernings were as similar as possible to those of *Libre Baskerville*.

2.1 Additions

In reworking *Libre Baskerville*, abbreviated below to *LB*, to *BaskervilleF*, abbreviated below to *B*, there were some major additions, listed here.

- *LB* provides no bold italic style—*B* adds one. Although most users most likely make little use of bold italic in text, apart perhaps from titles in some packages, it is essential for producing a proper bold math italic alphabet.
- *LB* had only one normal sized figure style, proportional lining.[3] *B* contains four such styles, adding tabular lining, tabular oldstyle and proportional lining, the default being tabular lining, which is needed for proper mathematical figure spacing.
- *LB* has superior and inferior figures at just 40% of the size of its lining figures. *B* contains a reworked set of inferior and superior figures at approximately 60% of the size of its lining figures. (The pre-built fractions, like ¾, were remade to use the larger figures, and the `\textfrac` macro attempts to provide other fractions with attention paid to appropriate spacing.)
- *LB* has only figures in its superiors. *B* adds superior upper and lower case letters, with `eacute` and `egrave` the only accented superior letters.
- *LB* has no small caps. All four styles of *B* contain a Small Cap alphabet containing all characters in the T1 encoding.
- An LY1 encoding is provided. This may be of interest principally to Scandinavian users who wish to have access to the `fj` and `ffj` ligatures, for example, in typesetting fjord. (There is no room left in the T1 encoding for these ligatures.) The other unusual ligatures available in LY1 are `fb` (fb), `fh` (fh) and `fk` (fk).

As `otf` versions of the fonts are provided, they may be used directly in Unicode TEX by means of the `fontspec` package. The `baskervillef` package provides `baskervillef.fontspec`, a file specifying the names of the relevant `otf` files. It is loaded automatically by `fontspec`, so it suffices to include in your preamble:

```
\usepackage{fontspec}
\setmainfont[Ligatures=TeX]%
 {baskervillef}
```

Usage under LATEX has several options that are spelled out in detail in the package documentation.

For math typesetting to accompany BaskervilleF, one may use the package `newtxmath` with the option `baskerville`. For instance:

```
\usepackage{baskervillef}
\usepackage[baskerville,vvarbb]{newtxmath}
\usepackage[cal=boondoxo,frak=boondox]{mathalfa}
```

Some sample output:

A Simple *Central Limit Theorem*:
Let $X_1, X_2, \cdots$ *be a sequence of* i.i.d. *random variables with mean* 0 *and variance* 1 *on a probability space* $(\Omega, \mathcal{F}, \Pr)$. *Let*

$$\mathfrak{N}(y) \coloneqq \int_{-\infty}^{y} \frac{\mathrm{e}^{-t^2/2}}{\sqrt{2\pi}}\,\mathrm{d}t,$$

$$S_n \coloneqq \sum_{1}^{n} X_k.$$

Then

$$\Pr\left(\frac{S_n}{\sqrt{n}} \le y\right) \xrightarrow[n\to\infty]{} \mathfrak{N}(y)$$

or, equivalently, for $f \in \mathcal{C}_\mathrm{b}(\mathbb{R})$,

$$\mathbb{E}f\left(S_n/\sqrt{n}\right) \xrightarrow[n\to\infty]{} \int_{-\infty}^{\infty} f(t)\frac{\mathrm{e}^{-t^2/2}}{\sqrt{2\pi}}\,\mathrm{d}t.$$

[2] It is a pleasure to acknowledge Karl's encouragement and very useful technical feedback throughout this project.

[3] In the original *Baskerville*, all figures were oldstyle, a.k.a. lowercase figures. Lining figures, the much more common form now, were not introduced until several years after Baskerville's death.

BaskervilleF is provided with a theorem font, a version of italic having upright punctuation and lining figures, which is, in my opinion, more suitable than ordinary italic for theorem statements and the like. I have abused the NFSS standards by setting `\textsl` to point to the theorem font rather than a real slanted variant. This is rather handy because we are then not limited to using the theorem font only in those theorems where the styles using appropriate declarations are in force.

3 Comparison to other Baskerville fonts

3.1 Other free Baskerville fonts

The following illustrates some of the differences between *BaskervilleF* on the left and its parent, *Libre Baskerville* on the right. (*BaskervilleF* has been rendered at four times natural size and *Libre Baskerville* at 3.6 times natural size, so as to match cap-heights.)

BaskervilleF Libre Baskerville

The sharper serifs, higher contrast and shorter x-height of *BaskervilleF* is quite apparent.

Had we instead matched x-heights, the comparison would be:

BaskervilleF Libre Baskerville

This time, the cap-height and ascender height of *Libre Baskerville* are clearly much less those of *BaskervilleF*.

If we compare *BaskervilleF* against *Baskervaldx* with matched x-heights we get:

BaskervilleF Baskervaldx

As is apparent, *Baskervaldx* is heavier, lower contrast, and more spread out horizontally.

3.2 Commercial Baskerville fonts

The next two comparisons involve the commercial font *Monotype Baskerville*, available on the Mac as a system font.

BaskervilleF Monotype Baskerville

The Monotype roman has different glyph widths (note in particular the lower bowl of its B, which is less prominent than in other *Baskervilles*), but is otherwise a fairly good match to *BaskervilleF* in terms of weight, contrast and spacing when magnified by 106.5%.

BaskervilleF Monotype Baskerville

The Monotype italic is substantially more delicate than *BaskervilleF*'s italic.

BaskervilleF ITC NewBaskerville

The *ITC NewBaskerville* italic, on the other hand, shown with matching x-height, is heavier, has lower cap-height and ascender height, and its lower-case letters are a bit wider, but the family resemblance to *BaskervilleF* is apparent.

Finally, here is *Storm Baskerville*, the most authentic revival of the original *Baskerville*, at the same x-height. Though it's not obvious at even the moderate magnifications displayed below, the design details in these fonts are truly remarkable.

Baskerville120 Baskerville120-Italic

In metal type, the *Baskerville* of Deberny & Peignot was from matrices made using Baskerville's original punches, while that of Stephenson Blake (1913) is said [SB88] to be a reproduction of Fry's Baskerville. The samples below were scanned from 30pt sources and so have thinner strokes than normal body text.

Deberny & Peignot Baskerville

Stephenson Blake Fry's Baskerville

4 Bibliography

[KB01] Karl Berry, Making outline fonts from bitmap images. *TUGboat*, 22:4 (2001), 281–285. `http://tug.org/TUGboat/tb22-4/tb72berry.pdf`

[SB88] James Sutton and Alan Bartram, *An Atlas of Typeforms*. Wordsworth Editions, Ware (1988).

◇ Michael Sharpe
`http://ctan.org/author/id/sharpe`

Programming LaTeX — A survey of documentation and packages

Brian Dunn

Abstract

A survey of documentation sources and packages useful for LaTeX programmers.

1 Introduction

Reinventing the wheel may be useful if you think that you can do it better. Worse, though, is not even being aware that the wheel has already been invented in the first place, which can be an embarrassing waste of time. Such can be the case both for a new LaTeX programmer who isn't aware of the many ways things may be done, but also for someone, the author included, who learned LaTeX many years ago but may have missed some of the recent advancements in package code and documentation.

A wealth of information is available, not only in print and online, but also directly embedded in the typical LaTeX distribution. The following is meant to be a broad overview of some of today's resources for LaTeX programmers.

(The latest version of this document is available in the docsurvey package.)

2 Printed books

Even in an electronic/online era, printed books still have the advantage of being able to be opened for reference without taking up space on the screen. Printed books also provide extended discussion of useful topics, have extensive human-edited indexes which are more useful than a simple document-wide search function, and some are also available in electronic format.

LaTeX: A Document Preparation System:
The classic introduction to LaTeX, in continuous reprint for decades. [1]

Guide to LaTeX:
An introduction and more advanced material, including an extensive reference guide. Fourth edition: 2004. [2]

More Math into LaTeX:
Updated to a fifth edition in 2016. [3]

LaTeX Beginner's Guide:
An overview with numerous examples. [4]

LaTeX Cookbook:
More examples. [5]

The LaTeX Companion:
Provides extended discussion and examples of the inner workings of LaTeX and numerous useful packages. Second edition: 2004. [6]

Other books are listed at the UK TUG FAQ:
`http://www.tex.ac.uk/FAQ-latex-books.html`

3 Electronic books

Provided with the TeX distribution:

The Not So Short Introduction to LaTeX 2ε:
Covers introductory material, customizations, and a simple package. [7] (`texdoc lshort`)

LaTeX 2ε: An unofficial reference manual:
A thorough but concise reference manual for LaTeX 2ε, available in several languages. [8] (`texdoc -l latex2e-help`)

LaTeX WikiBook:
An online book, includes information about creating LaTeX packages and classes.
`https://en.wikibooks.org/wiki/LaTeX`

TeX by Topic, A TeXnician's Reference:
A reference for TeX. This may be useful for understanding the source code of LaTeX packages, many of which are quite old and written in low-level TeX. [9] (`texdoc texbytopic`)

4 Symbol references

These are lists of the LaTeX commands which produce symbols.

Comprehensive LaTeX Symbol List:
More than 14,000 symbols and LaTeX commands. [10] (`texdoc symbols-letter`) (`texdoc symbols-a4`)

Every symbol (most symbols) defined by unicode-math:
Unicode math symbols. [11] (`texdoc unimath-symbols`)

5 FAQs

UK TUG FAQ:
A wide-ranging list of frequently-asked questions. [12] (`texdoc letterfaq`) (`texdoc newfaq`)

Visual LaTeX FAQ:
Click on a visual element to learn how it is programmed. [13] (`texdoc visualFAQ`)

6 Accessing embedded documentation

A large amount of documentation is included in a TeX distribution. Most can be accessed with the `texdoc` program. Use `texdoc -l name` to select from many choices of matching package, file, or program names. In some cases the same document is available in both letter or A4 paper sizes, or in several languages.

The program `kpsewhich` may be used to find out where a file is located. `kpsewhich filename`

searches for and returns the path to the given filename.

`kpsewhich` can also return directories, such as:

```
kpsewhich -var-value TEXMFROOT
kpsewhich -var-value TEXMFDIST
kpsewhich -var-value TEXMFLOCAL
```

Some package authors choose not to include the source code in the package documentation. To view the source code:

1. To locate and read a package's `.sty` file:
   `kpsewhich package.sty`
   Usually these files have their comments removed, so it is better to use the `.dtx` file instead.
2. The `.dtx` file is usually available, and will have the package's source code.
   `kpsewhich package.dtx`
   The comments are not yet typeset and so will not be as easily read.
3. To typeset the documentation with the source code, copy the `.dtx` file and any associated image files somewhere local and then look for `\OnlyDescription` in the source. This command tells the ltxdoc package not to print the source code.
4. Remove `\OnlyDescription`, then process the `.dtx` file with `pdflatex package.dtx`. Barring unusual circumstances, this will create a new documentation `.pdf` file with the package source code included.

7 Source code

The source code for LaTeX 2ε itself is also included in the distribution.

The LaTeX 2ε sources:
Occasionally useful for figuring out how something really works. [14] (`texdoc source2e`)

List of internal LaTeX 2ε macros useful to package authors:
A list of the core LaTeX macros, each of which is linked to the source code. [15] (`texdoc macros2e`)

8 Comprehensive TeX Archive Network

The Comprehensive TeX Archive Network (CTAN) provides a master collection of packages. A search function is available, which is useful when you know the name of a package or its author, and a list of topics is also provided. There are so many topics, however, that finding the right topic can be a problem in itself. One useful method to find what you are looking for is to search for a related package you may already know about, then look at its description on CTAN to see what topics are shown for it. Selecting these topics then shows you related packages. [16]

9 Packages

A number of packages are especially useful for LaTeX programmers:

xifthen:
Conditionals.

etoolbox:
A wide range of programming tools, often avoiding the need to resort to low-level TeX.

etextools:
Adds to etoolbox. Strings, lists, and more.

xparse:
Define macros and environments with flexible argument types.

environ:
Process environment contents.

arrayjobx, fifo-stack, forarray, forloop, xfor:
Programming arrays, stacks, and loops.

iftex:
Detect TeX engine.

ifplatform:
Detect operating system.

xstring:
String manipulation.

keyval, xkeyval, kvsetkeys:
Key/value arguments.

pgfkeys, pgfkeyx:
Another form of key/value arguments.

kvoptions:
Key/value package options.

expl3:
LaTeX3 programming.

l3keys, l3keys2e:
Key/value for LaTeX3.

CTAN topic macro-supp:
An entire topic of useful programming macros.

10 Creating and documenting packages

10.1 Packages and programs

Documentation for those interested in creating their own package or class:

How to package your LaTeX package:
A tutorial. [17] (`texdoc dtxtut`)

LaTeX 2ε for class and package writers:
Programming a package or class. [18] (`texdoc clsguide`)

The doc and shortvrb packages:
Packages for documenting packages. [19] (`texdoc doc`)

The DocStrip program:
The program which processes `.dtx` and `.ins` files to generate documentation and `.sty` files. [20] (`texdoc docstrip`)

10.2 Articles

Related articles from *TUGboat*:

Rolling your own Document Class: Using LaTeX to keep away from the Dark Side:
An overview of the article class. [21]

Good things come in little packages: An introduction to writing .ins and .dtx files:
How and why to create your own `.dtx` and `.ins` files. [22]

How to develop your own document class — our experience:
A comparison of developing class vs. package files. [23]

11 Online communities

English forums:

TeX — LaTeX Stack Exchange:
Almost any question has already been asked, and a quick web search will find answers, ranked by vote.
`http://tex.stackexchange.com/`

LaTeX Community:
A traditional forum with quick replies to your questions.
`http://www.latex-community.org/`

German forums:

TeXwelt:
`http://texwelt.de/wissen/`

goLaTeX:
`http://golatex.de/`

Newsgroup: `comp.text.tex`

◇ Brian Dunn
bd (at) bdtechconcepts dot com
http://bdtechconcepts.com

References

[1] LaTeX: A Document Preparation System, Leslie Lamport, second edition, Addison Wesley, 1994, ISBN 0201529831.

[2] Guide to LaTeX, Helmut Kopka and Patrick W. Daly, fourth edition, Addison-Wesley, 2004, ISBN 0321173856.

[3] More Math Into LaTeX, George Grätzer, 5th ed., Springer, 2016, ISBN 3319237950.

[4] LaTeX Beginner's Guide, Stefan Kottwitz, Packt Publishing, 2011, ISBN 1847199860.

[5] LaTeX Cookbook, Stefan Kottwitz, Packt Publishing, 2015, ISBN-13 9781784395148, `http://latex-cookbook.net/`

[6] The LaTeX Companion, Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle and Chris Rowley, second edition, Addison-Wesley, 2004, ISBN 0201362996.

[7] The Not So Short Introduction to LaTeX 2ε, Tobias Oetiker, `https://ctan.org/pkg/lshort`

[8] LaTeX 2ε: An unofficial reference manual, George D. Greenwade, Stephen Gilmore, Torsten Martinsen, and Karl Berry, `http://home.gna.org/latexrefman`

[9] TeX by Topic, A TeXnician's Reference, Victor Eijkhout, Addison-Wesley UK, 1991, ISBN 0201568829, `http://eijkhout.net/texbytopic/texbytopic.html`

[10] The Comprehensive LaTeX Symbol List, Scott Pakin, `https://ctan.org/pkg/comprehensive`

[11] Every symbol (most symbols) defined by unicode-math, Will Robertson, `https://ctan.org/pkg/unicode-math`

[12] UK TUG FAQ, UK TeX Users Group, `http://www.tex.ac.uk/`

[13] The Visual LaTeX FAQ, Scott Pakin, `https://ctan.org/pkg/visualfaq`

[14] The LaTeX 2ε Sources, Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley, and Rainer Schöpf, `https://ctan.org/pkg/source2e`

[15] List of internal LaTeX 2ε Macros useful to Package Authors, Martin Scharrer, `https://ctan.org/pkg/macros2e`

[16] Comprehensive TeX Archive Network (CTAN), `https://ctan.org`

[17] How to Package Your LaTeX Package, Scott Pakin, `https://ctan.org/pkg/dtxtut`

[18] LaTeX 2ε for class and package writers, LaTeX3 Project, `https://ctan.org/pkg/clsguide`

[19] The doc and shortvrb packages, Frank Mittelbach, `https://ctan.org/pkg/doc`

[20] The DocStrip program, Frank Mittelbach, Denys Duchier, Johannes Braams, Marcin Woliński, and Mark Wooding, `https://ctan.org/pkg/docstrip`

[21] Rolling your own Document Class: Using LaTeX to keep away from the Dark Side, Peter Flynn, *TUGboat* 28:1 (2007), pp. 110–123, `http://tug.org/TUGboat/tb28-1/tb88flynn.pdf`

[22] Good things come in little packages: An introduction to writing .ins and .dtx files, Scott Pakin, *TUGboat* 29:2 (2008), pp. 305–314, `http://tug.org/TUGboat/tb29-2/tb92pakin.pdf`

[23] How to develop your own document class — our experience, Niall Mansfield, *TUGboat* 29:3 (2008), pp. 356–361. `http://tug.org/TUGboat/tb29-3/tb93mansfield.pdf`

CTAN goes 2.0 — New community features and more

Gerd Neugebauer

Abstract

The portal of the Comprehensive TEX Archive Network (CTAN) is continuously improving. Additional features for the personalized interaction with the portal and the contribution to its contents are now publicly available. This allows — among other features — registered and authenticated users to rate packages and leave comments on them.

In addition, CTAN announcements are now available on the portal and via RSS feeds.

1 Introduction

In the Web context the magical number is 2.0. This is originated in the term Web 2.0 as coined by Eric Knorr in 2003 [Kno03]. It is associated with the enrichment of the user experience by community functions. The community member is not only a pure consumer but can contribute to the Web content.

CTAN is based on the principle of contribution of material and participation of community members on the level of packages. Anyone can upload a package to CTAN. The CTAN team helps to ensure consistency and a high quality of the information provided in the catalogue of packages. This aims not only at direct online use but also at the production of TEX distributions like TEX Live and MiKTEX.

With the new features of the CTAN portal we enter the era of Web 2.0. Community members are now enabled to contribute information to CTAN as well. First of all, this information augments the package description with rating and commenting of packages. Thus it allows other members to benefit from the experiences with packages shared on the CTAN portal.

2 CTAN taken personally

From the start, CTAN has been built with user management and login capabilities. Until now, I have used this feature as the only user for managing the portal. The registration has been deactivated and the login page hidden.

2.1 Experiences with the guest book

The CTAN portal has had a guest book since its relaunch in 2012 [Neu13]. Soon spammers found this page and started to pollute the entries. I have manually deleted offending entries — sometimes quite quickly. Nevertheless it was annoying.

Thus I have added a list of stop words to the portal. Whenever I delete a guest book entry I also add one or more stop words. The stop words are checked when a new guest book entry is submitted. The entry is automatically deleted when a stop word is found.

As a result, I observed that trash was reduced but did not vanish completely. It's an arms race and I am certain that it will continue. Thus it became clear that something has to be done to avoid a similar effect for the community features.

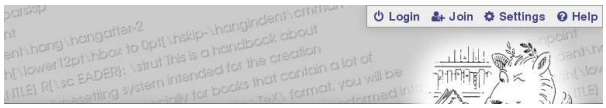


Figure 1: The login and registration ("Join") links

2.2 Registration and login

CTAN has had a strict security and privacy policy since its beginnings. As one consequence we have activated an encrypted communication via HTTPS some time ago. The new community features follow this spirit. We take care of your data and you have control over the information presented about you on the Web.

One change is visible on each page: the links to the page for login and registration ("Join"). They appear in the upper right corner of each page, as shown in figure 1.

To get started you have to register a new account to gain access to the community features. We collect a few pieces of information during the registration process. You can choose a nearly arbitrary account name. If all privacy guards are active then only this alias will be seen on the Web portal. Nevertheless we prefer attributions to you with your real name. Thus it needs to be given. At least the CTAN team wants to know who the members are. Finally you need a working email address. This is used during the registration process to verify that you can be reached if required — for instance if you forgot your password.

Two things are used to identify a user: the first is the account name and the second is the email address. Thus the portal tries to avoid different accounts being created with identical account names or email addresses.

A few more settings can be made on the registration page to get things right immediately. After you have submitted the registration an email will be sent to the email address you have entered. It contains a confirmation link to a page where you can set up your initial password and activate the account. Now you can log in and participate.

A few situations deserve special treatment. It appears obvious that a user name already in use cannot be chosen. If you enter an email address which is already in use, you cannot register a second account for it. This is most probably the case when you have forgotten that you had already registered. Just login — or request a password renewal if you've forgotten that as well.

The login is fairly simple. You can use the link on each page to get to the login page. Alternatively if you request a private page without being logged in you are automatically redirected to the login page first.

The login uses your account name and your password. The password is not known to the CTAN team and cannot easily be uncovered. Remember our privacy policy — we try to protect your privacy.

If you happen to forget your account name or password you can request an email with a reminder or a link to a page which allows you to choose a new password.

You can even decide to delete your account on CTAN. In this case you cannot login to this account any more and the information gathered by the system is deleted.

2.3 Your personal dashboard

You have a personalized starting page — called a dashboard. It is accessible exclusively to you. Nobody else sees this page. This page is usually shown immediately after you log in. A sample is shown in figure 2.

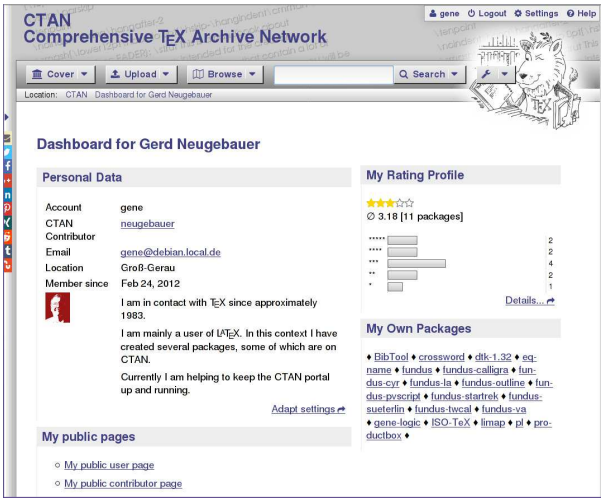


Figure 2: A sample dashboard

From your dashboard you can see which data is stored for your account. Nearly everything can be modified at any time. The only fixed value is the account name. It is your unique identifier for the system. Once chosen during registration it cannot be altered. Thus choose wisely.

You can see there statistics about your ratings. There you can find a link to the complete list of your ratings and comments. With two clicks you can reach a particular package page. There you can revise your rating and your comments at any time and as often as you like.

2.4 More preferences for you

Any community member has had a small set of settings used even without being logged in. This consists mainly of the selection of the skin — the appearance of the Web pages. Some time ago, this was augmented by the option to disable hyphenation and the selection of the language. This has been requested particularly because the cut-and-paste operation has revealed insufficient support of the UTF-8 character encoding by several programs. This leads to mysterious unprintable characters for the hyphenation marks in pasted text.

The selection of the skin and the hyphenation flag are stored locally in the browser. Thus they can be used without login. On the other hand, they are bound to a single browser and cannot be shared between different computers or devices.

These choices can now be found on the settings page. This page provides access to additional settings if the member is logged in. The complete group of settings about your account and privacy can be found there too. Notably, most of the fields from the registration can be modified here.

2.5 Your public account page

Each registered community member has a `/home` page as the URL `/home/account`. This `/home` page shows all information about the account which has been approved by the user.

Currently the public account page is similar to the dashboard (see figure 3). But this might change in the future.

2.6 Identifying package authors

The strict privacy policy of CTAN prevents an author's email address from being shown prominently on the Web. The CTAN team does not have the capacity to extend the information about authors with a flag signaling the agreement of the author to pass on their email address. Nor do we have the capacity to get such an agreement from each of the more than 2000 authors.

Authors are usually interested in getting feedback on their packages. Thus in most cases it is

Figure 3: A public account page

not hard to find an email address in a "readme" file, the package documentation, or even the source code. Now an author is also enabled — and encouraged — to provide his email by registering as a CTAN community member and establishing a link to the CTAN author which should already be in place. Thus package users can more easily find the email address and get in contact with the author.

The association works via the email address of a contributor known to the CTAN team. If a user registers an account with the same email address then he is asked to establish an association. For instance, I have chosen the account name "gene". In the CTAN catalogue someone named me "neugebauer". Since the email addresses were identical, the system has offered me the possibility to establish an association and I did it. Now a link is on my `/author` page pointing to my `/home` page and vice versa.

This additional association between package authors and CTAN accounts allows visitors to find the email address of an author more quickly. Of course, the publication of the email address remains under the control of the contributor.

2.7 Package upload simplified

Package upload is one place where a contributor can profit from having a CTAN login. As you may already know, the upload form is pre-filled with the data from the last visited package page. This means that if you navigate to a page which is related to a package then this information is kept during the current browser session. The page can be the package page under `/pkg/...` or a package-related page in the `tex-archive/` tree.

When you are logged in, your name and email address is filled additionally. Thus the procedure to upload an update of one of your packages becomes rather simple:

- Login
- Select the package from the list of your packages
- Press "Upload"
- Fill in the version number, announcement text, and select the file to be uploaded
- Press "Submit contribution" and you are done

With these simplifications, authors can be further encouraged to upload their package developments to CTAN.

3 Sharing your experience — Rating and commenting packages

The user rating of products can be a good source of information. We see this kind of feedback on many online shops. A visitor can get an impression of the experiences of other users. Now we carry this over to the packages on CTAN.

3.1 New building blocks on package pages

Each package has a page on CTAN which presents the data gathered in the catalogue about this package. For instance, the package `biblatex` is associated with the page `https://www.ctan.org/pkg/biblatex`.

When looking at the package pages in recent years, they may have seemed a bit underpopulated. This is fixed now. More information and functionality is in place. An example can be seen in figure 4.

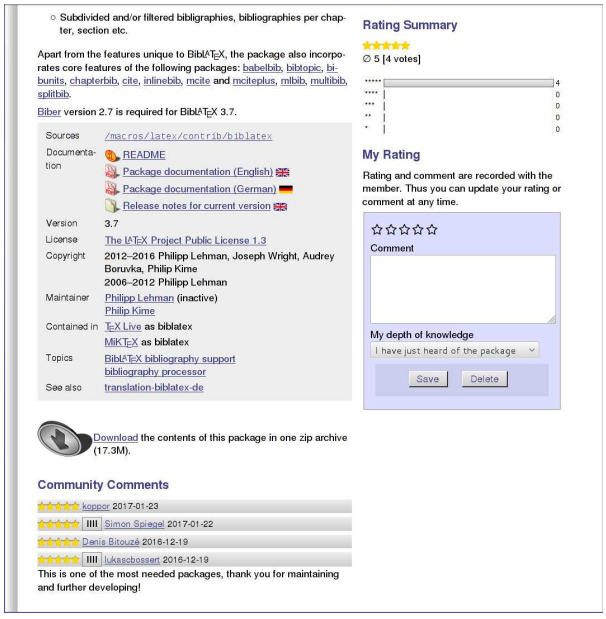


Figure 4: New features on package pages

On this page you can find three new sections. The section "Community Comments" lists the comments and ratings of the CTAN community members. The section "Rating Profile" shows some statistics about the average rating, the number of votes and the distribution of the votes. The section "My Rating" allows you to rate the current package or modify your previous rating. This functionality is available to authenticated members only.

The aim of the CTAN team is to avoid abuse and nonsense content. Thus we do not allow anonymous ratings and comments. As a side effect this policy allows you to review your rating and adapt it to your current experiences with the most recent version of the package.

When you enter the rating for a package you are asked for several fields. First is the number of stars — more is better. You can enter a textual comment. It contains whatever you have to say but is optional. Finally you are asked to rate yourself. This provides an indication for the readers on how deeply you are involved with the package. By the way, it is legitimate for an author to rate their own package. Even this can provide insights for readers.

In figure 4 you see the sections with a few ratings and comments. The packages await your contribution with ratings and comments.

4 CTAN and JavaScript — Supporting old-fashioned ways

Nowadays the Web is nearly unusable without JavaScript. This creeps into CTAN as well. CTAN uses JavaScript, but sparsely. Nevertheless the 2.0 features make use of JavaScript. You will not be able to see or use much without JavaScript enabled — don't even try.

Recently a complaint reached me about the use of JavaScript on the archive browser pages. Since some directories are huge it can take some time to render the directory listing. In order to show some kind of progress the implementation uses an asynchronous AJAX call — i.e. JavaScript — to get the data while a CTAN banner is rotating. If JavaScript was not available the banner rotated forever since no attempt is made to load the data.

This has been fixed. A page without JavaScript is used automatically — it may take slightly longer before the browser shows a reaction. At least the basic functionality of browsing the archive directory and searching can be used without JavaScript now.

5 Feeds from the Lion — News on CTAN

The TEX world is constantly evolving. New and updated packages can be found on CTAN. The announcements have been published via the mailing list `ctan-ann@dante.de` for a long time. This mailing list is run by DANTE e.V. Anyone interested can subscribe to receive the email via the administration page, `https://lists.dante.de/mailman/listinfo/ctan-ann`.

Newsgroups supplement this service. For instance the newsgroup `comp.text.tex` receives the announcements from the mailing list.

5.1 Gmane is dead — Long live CTAN

Gmane has been a good and reliable way to access many mailing lists. When it went down at the end of June 2016 we felt that there was a gap to be filled — at least for CTAN and the TEX world. With the end of Gmane, there was no good way to learn of changes via a news feed. This functionality is now provided by the CTAN portal.

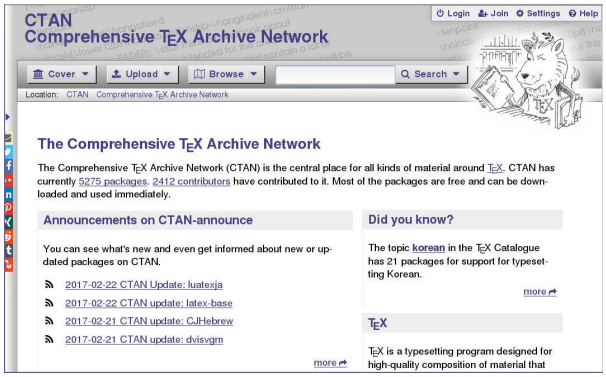


Figure 5: Announcements on the CTAN home page

The CTAN home page (see figure 5) now shows the four most recent announcements on `ctan-ann`. In addition, a link to a page with a larger list is provided (see figure 6). On this page you can find the orange buttons for "Atom" and "RSS". They are links to the respective feeds. Currently the formats Atom 1.0 and RSS 2.0 are supported. You can enter these URLs in the feed reader of your choice to read them.

Maybe you are interested in single packages only. In this case you can navigate to the respective package page. If there have been announcements for this package, you can again find the four most recent ones listed and a link to a page with more, and again the orange buttons for "Atom" and "RSS" with the links for the newsfeeds.

5.2 Activity diagrams — sort of

On the package page you can sometimes find a diagram with (sort of) spectral lines. The idea was to visualize the activity of the package development in

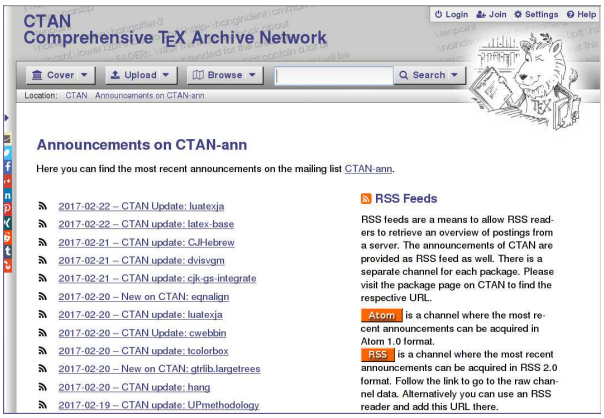


Figure 6: The announcements page

some way. The only information available at CTAN are the announcements. Thus the announcements are shown in this diagram (see figure 7).

This diagram can be misleading in several situations. The association of an announcement to a package relies on a heuristic which tries to identify the package name in the announcement text. This heuristic works fairly well for recent announcements but may fail for older ones or announcements for several bundled packages.

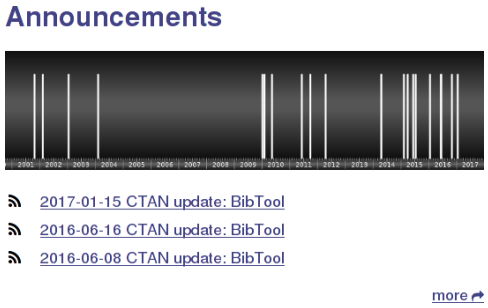


Figure 7: Announcements spectral lines

Some of the packages are mirrored to CTAN automatically. In this case no announcement may appear on `ctan-ann`. The authors can ask the CTAN team — via email — to publish an announcement anyway. But this option is rarely used.

The upload form offers the choice to upload a package without announcement. If the uploader selects this option then no announcement is sent and consequently no spectral line is shown.

6 The road ahead

The CTAN portal is evolving. And features can be added, modified, or even removed entirely.

6.1 The light on the horizon

With the new possibility of identifying users, we are able to approach more functionality. Things which should not be allowed anonymously can be opened for community members or package authors.

We can envision allowing package authors to see and edit the catalogue entry for their package. For instance the description and the topics might be made accessible in this way. Nowadays this information is maintained by the CTAN team. Anyone can submit suggestions for improvements by sending an email. But such a process might be offered on the portal as well.

For any user, one can envision lists on the home page containing links to packages of interest or documentation. For the user this list acts as a personal library. These can be taken into account when rating a package. One could consider making such lists public. Thus lists of packages for special interests could be maintained by the community.

Another possibility would be to make the community features themselves configurable for an authenticated user. If someone does not like the rating and commenting, then this could be disabled. But I suspect this configuration would not be often used.

6.2 It's your turn to contribute to www.ctan.org

In this article, you have seen some of the new community features of the CTAN portal. Now it is up to you to make best use of it. Visit the CTAN portal, register, and start rating the packages you know best from your daily use. Please share the wealth of your experiences with some of the packages on CTAN and make the portal more fruitful for others.

Enjoy `www.ctan.org` and keep on TEXing.

References

[Kno03] Eric Knorr. 2004 — The Year of Web Services. *CIO*, December 15 2003. `http://www.cio.com/article/2439869/web-services/2004--the-year-of-web-services.html`.

[Neu13] Gerd Neugebauer. CTAN: Relaunch of the Web Portal. *TUGboat*, 34(1):6–9, 2013. `http://tug.org/TUGboat/tb34-1/tb106neugebauer.pdf`.

◇ Gerd Neugebauer
Im Lerchelsböhl 5
64521 Groß-Gerau, Germany
`gene (at) gerd-neugebauer dot de`
`www.gerd-neugebauer.de`

An introduction to the LaTeX cross-referencing system

Thomas Thurnherr

Abstract

One of the most powerful features of LaTeX is the cross-referencing system. It allows referencing numbered structures, such as headings, figures, tables, or equations anywhere in the same document. Importantly, references are automatically updated by the system whenever necessary. This article provides an overview of the referencing system and introduces several packages that extend it.

1 Introduction

LaTeX provides a referencing system that allows referencing almost any numbered object in the same document. To do that, the system implements the `\label` and `\ref` macros. The `\label` macro sets a marker that uniquely identifies a numbered object, which can then be referenced through the `\ref` macro. The `\pageref` command is less frequently used. It prints the page number of the labeled object. An example is provided below, where a marker `sec:intro` is added to a section. This marker is then used in the next section to refer to the previous section. Assuming that "Introduction" is the first section in the document, `\ref{sec:intro}` just prints the number: '1'.

```
\section{Introduction}\label{sec:intro}
% Content

\section{Methods}
In section \ref{sec:intro} on page
\pageref{sec:intro} we introduced ...
```

2 Specifying a label or marker

The system imposes no restrictions on the format of a marker. However, it is common practice to add a prefix + colon (`:`) to every marker. A prefix helps the author to identify the kind of object that is referenced. For example, to reference a figure a possible marker might be `\label{fig:schema}`. Table 1 shows prefix suggestions for the most frequently referenced numbered objects. Although it is good practice to use a prefix, these are merely suggestions and an author is free to choose a different set of prefixes or omit them altogether.

3 Placement of labels

Ideally, the `\label` is placed right after the numbered object to be labeled. For floating environments such as `figure` or `table`, `\label` has to be placed after or within the `\caption` macro, because the caption produces the figure or table number. Placing a `\label` before the numbered object almost certainly will produce the wrong reference. The same might happen if the `\label` is placed too far after the numbered object.

Table 1: Suggestions for marker prefixes.

Object	Prefix	Object	Prefix
Chapter	ch	Figure	fig
Section	sec	Table	tab
Subsection	ssec	List item	itm
Appendix	app	Equation	eqn

4 Generating/updating references

A document generally requires two typesetting runs to generate reference numbers. Behind the scenes, the system collects all markers in an `aux` meta-file while typesetting the document once. It then reads this `aux` file to update references during the second run of typesetting. Whenever references cannot be generated correctly by the system, double question marks (**??**) are produced in the output document.

If references are not updated correctly after retypesetting, a quick look through the log file might help identify the problem. The next section (5) describes warnings related to the referencing system.

5 LaTeX warnings related to the referencing system

It is not uncommon to see warnings related to the referencing system in the `log` file. Two kinds of warnings may arise from ill-defined markers:
1) `There were undefined references.`
2) `There were multiply-defined labels.`
The first is due to a missing marker that is referenced, whereas the latter is due to a marker that is defined more than just once. Multiply-defined labels can happen when the author copies a piece of code, such as a figure environment, and forgets to replace the marker with a unique name.

6 Packages extending the referencing system

There are numerous packages that extend the LaTeX cross-referencing system. Here, I provide a brief introduction to `varioref`, `cleveref`, `hyperref` and `xr`/`xr-hyper`. I chose these particular packages because I think each of them extends the referencing system in a unique and comprehensive way. In general, these packages do not redefine the functions of the standard referencing system, but define new

macros with a slightly different and beneficial behavior. Therefore, the standard behavior of `\label`, `\ref` and `\pageref` is not affected by using these packages.

6.1 The varioref package

The `varioref` package [4] extends the basic referencing system with slightly more sophisticated commands. `\vref` combines `\ref` with `\pageref` to print the number of the referenced object together with the page number. If the referenced object is on the same page, the page number is omitted. The `\vpageref` macro extends this idea to `\pageref`. If the referenced object is on the same page, the output says: "on this page". Otherwise, the output shows the page number. Finally, the package provides the `\vrefrange` and the `\vpagerefrange` commands to print a range of numbers and page numbers when multiple references are provided.

6.2 The cleveref package

Often, the author mentions the type of objects that is referenced in the text. For example, a figure could be referenced as follows: "This is shown in figure 1". The `\ref` macro only prints the number of the referenced object. This is where the `cleveref` package [1] comes in. It automatically recognizes the type of object that is referenced and prints the reference accordingly. For this purpose, the package provides the `\cref` command. Similar to the `varioref` package, it also implements macros to print the page number of a referenced object (`\cpageref`) and a range of objects or page numbers (`\crefrange` and `\cpagerefrange`).

6.3 The hyperref package

The main function of `hyperref` [2] that is related to the cross-referencing system is to make references clickable. Clicking a reference navigates the reader to the page where the referenced object resides. For that, it is sufficient to load the package in the preamble. Moreover, the package defines the `\autoref` macro, which is equivalent to `\cref` in the `cleveref` package. It prints the type of reference together with the reference number. The package implements a plethora of other functions not necessarily related to the cross-referencing system. A discussion of these is out of the scope of this article. Nevertheless, I highly recommend browsing the package documentation to learn more about what you can do with the package.

6.4 The xr/xr-hyper packages

The eXternal References `xr` package [5] allows referencing objects in external documents. This is particularly useful for scientific articles, where supplementary materials are often provided together with the main text. The package allows referencing supplementary figures and tables in the main text. To do that, the external document has to be defined by `\externaldocument{filename}` in the preamble of the main text. If the external `tex` file is in a different folder, the path has to be added to the file name. With that, markers defined in the external document can be referenced in the main text. To produce the correct number, external documents need to be typeset whenever they are changed.

```
\documentclass{article}
\usepackage{xr}
\externaldocument{supplementary-materials}
\begin{document}
    See supplementary figure \ref{fig:abc}.
\end{document}
```

To create hyperlinks to referenced objects in external documents with `hyperref`, load the `xr-hyper` package instead of `xr`.

6.5 Package loading order

As all of these packages affect the behavior of the referencing system, loading multiple packages may give rise to conflicts. To omit these, packages need to be loaded in the right order, namely: 1) `xr`/`xr-hyper`, 2) `varioref`, 3) `hyperref`, and lastly 4) `cleveref`.

6.6 The showlabels package

Finally, I would like to mention the `showlabels` package [3], which shows all markers in the margins of the output document (PDF). This package is extremely useful to keep track of markers in long documents, with a large number of labels.

References

[1] cleveref — Intelligent cross-referencing. `https://www.ctan.org/pkg/cleveref`.

[2] hyperref — Extensive support for hypertext in LaTeX. `https://www.ctan.org/pkg/hyperref`.

[3] showlabels — Show label commands in the margin. `https://www.ctan.org/pkg/showlabels`.

[4] varioref — Intelligent page references. `https://www.ctan.org/pkg/varioref`.

[5] xr — References to other LaTeX documents. `https://www.ctan.org/pkg/xr`.

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

How to use basic color models in LaTeX

Behzad Salimi

Abstract

This article provides a quick reference guide for new or experienced users on how to use the five basic LaTeX color models. We describe the color packages, the primary syntax, and examples of how to include colored text (or objects) in a document. In this article, we do not attempt to include a comprehensive coverage of all the options and features in the basic color models; instead, the intent is to help the interested authors start learning (and using) the color syntax in the shortest time possible.

1 Basic color models

The default basic color models supplied with standard LaTeX installation provide an extensive range of colors that are easy to use for colored text or reverse color highlighted text.

The five standard color models in LaTeX are:

§	*model*	*color names*	*must load*
3	basic	predefined	a or b
4	gray	gray	a or b
5	usenames	predefined	b
6	RGB	arbitrary	a or b
7	CMYK	arbitrary	a or b

a: `\usepackage{color}`
b: `\usepackage[usenames]{color}`

In the **basic** and `usenames` color models, colors and their corresponding names are predefined, so only those predefined color names are available. In the **rgb** and **cmyk** models, a custom color of user-specified name with infinite variation of color hue can be specified (mixed). In the **gray** model, shades of gray are specified.

2 Syntax for colored text

The syntax for using colored text in all color models above is one of two equivalent forms:

1. `{\color{`*name*`}text in color}`
2. `\textcolor{`*name*`}{text in color}`

where *name* is either a predefined or user-specified color name in one of the color models. The `\color` command may make extra vertical space in some environments such as `tabular`; in such cases, use `\textcolor` command.

To specify a user-defined color in **gray**, **rgb** or **cmyk** color models, the syntax may be a local or global definition:

1. Local definition:
   `\textcolor[`*model*`]{`*specification*`}{text}`
   where *model* is **gray**, **rgb** or **cmyk** and specification is described in §§ 4, 6, 7.
2. Global definition: use the `\definecolor` command:
   `\definecolor{`*color1*`}{rgb}{n1,n2,n3}`
   `\definecolor{`*color2*`}{cmyk}{n1,n2,n3,n4}`
   then use these custom colors with the syntax above:
   `{\color{`*color1*`}text in color1}`
   `\textcolor{`*color2*`}{text in color2}`

See examples in the next section.

3 Basic color model

The predefined colors for the basic color model are used without any model specification. The eight predefined color names are **black**, **blue**, **cyan**, **green**, **magenta**, **red**, **white**, **yellow**. Syntax:

```
\textcolor{red}{\textbf{Sample text in red}}
```

Sample text in red

```
{\color{cyan}\textbf{Sample text in cyan}}
```

Sample text in cyan

Another example:

```
{\color{magenta} color changes
\textcolor{blue}{can be nested
\textcolor{red}{in all} }
{\color{green}color models.}}
```

color changes can be nested in all color models.

4 Gray scale model

The gray scale model is a variable scale gray and it takes only one number from 0=black to 1=white (as they appear on white paper).

The base color name in this package is fixed as **gray**; however, various shades can be defined using the base name **gray**. For example:

```
\newcommand{\gry}[1]{\textcolor[gray]{0.70}{#1}}
\definecolor{gray1}{gray}{0.50}
```

Syntax:

```
\gry{Sample text in 0.75 gray}
```

Sample text in 0.75 gray

```
\textcolor{gray1}{Next text in 0.50 gray}
```

Next text in 0.50 gray

5 'usenames' color model

In the LaTeX `usenames` color model, there are 68 predefined color names; see Table 1. This color package must be loaded with the '`usenames`' option:

```
\usepackage[usenames]{color}
```

Syntax:

```
\textcolor{BrickRed}{text color in BrickRed}
```

text color in BrickRed

```
{\color{RoyalPurple} text in RoyalPurple}
```

text in RoyalPurple

```
\colorbox{Fuchsia}{\textcolor{white}{%
          colorbox Fuchsia}}
```

colorbox Fuchsia

You can create an alias for a predefined color name, for example:

```
\definecolor{ogrn}{named}{OliveGreen}
\textcolor{ogrn}{ogrn is OliveGreen}
```

ogrn is OliveGreen

Table 1: The 68 standard colors known to dvips (not pdfLaTeX).

Apricot	Aquamarine
Bittersweet	Black
Blue	BlueGreen
BlueViolet	BrickRed
Brown	BurntOrange
CadetBlue	CarnationPink
Cerulean	CornflowerBlue
Cyan	Dandelion
DarkOrchid	Emerald
ForestGreen	Fuchsia
Goldenrod	Gray
Green	GreenYellow
JungleGreen	Lavender
LimeGreen	Magenta
Mahogany	Maroon
Melon	MidnightBlue
Mulberry	NavyBlue
OliveGreen	Orange
OrangeRed	Orchid
Peach	Periwinkle
PineGreen	Plum
ProcessBlue	Purple
RawSienna	Red
RedOrange	RedViolet
Rhodamine	RoyalBlue
RoyalPurple	RubineRed
Salmon	SeaGreen
Sepia	SkyBlue
SpringGreen	Tan
TealBlue	Thistle
Turquoise	Violet
VioletRed	
WildStrawberry	Yellow
YellowGreen	YellowOrange

6 RGB color model

In the **rgb** (red-green-blue) color model, the primaries are *additive*; that is, the contribution of each base color is added to obtain the range from black (no light) {0,0,0} to white (full light) {1,1,1}. The **rgb** color model takes three parameters — one value for each of the **red**, **green**, **blue** colors. The three colors can be "mixed" in any proportion; each parameter must be between 0.0 and 1.0 but they need not add to 1.0. The three base colors themselves can be selected with **red**={1,0,0}, **green**={0,1,0}, **blue**={0,0,1}.

The syntax is

1. Define a color locally, usually for one-time use:
   ```
   {\color[rgb]{n1,n2,n3}text}
   \textcolor[rgb]{n1,n2,n3}{text}
   ```
   Example:
   ```
   {\color[rgb]{0.8,0.1,0.8} Sample rgb color}
   ```
   Sample rgb color
2. Define a new color name:
   ```
   \definecolor{color1}{rgb}{n1,n2,n3}
   {\color{color1}text}
   ```
   Example:
   ```
   \definecolor{teal}{rgb}{0.0,0.5,0.5}
   \textcolor{teal}{Sample teal in rgb model}
   ```
   Sample teal in rgb model

7 CMYK color model

The *subtractive* primaries cyan, magenta, and yellow are the complements of red, green, and blue, respectively; that is, they are subtracted (like filters) from **rgb** for the range of full light (white) {0,0,0,0} to no light (black) {1,1,1,1}. The **cmyk** (cyan-magenta-yellow-black) model is a more complex color model that takes four parameters — one value for each of **cyan**, **magenta**, **yellow**, **black**. The four colors can be "mixed" in any proportion; each parameter must be between 0.0 and 1.0 but they need not add up to 1.0. The four base colors themselves are selected with **cyan**={1,0,0,0}, **magenta**={0,1,0,0}, **yellow**={0,0,1,0}, **black**={0,0,0,1}.

The syntax is analogous to that in §6:

1. Define a color locally, usually for one-time use:
   ```
   {\color[cmyk]{n1,n2,n3,n3} text }
   \textcolor[cmyk]{n1,n2,n3,n3}{text}
   ```
   Example:
   ```
   {\color[cmyk]{0.2,0.7,0.1,0.2}
   Sample cmyk text}
   ```
   Sample cmyk text
2. Define a new color name:
   ```
   \definecolor{clr1}{cmyk}{n1,n2,n3,n4}
   {\color{clr1}text}
   ```
   Example:
   ```
   \definecolor{brk}{cmyk}{0.2,0.7,0.3,0.3}
   \textcolor{brk}{Sample cmyk color model}
   ```
   Sample cmyk color model

8 Advanced, specialized color models

The basic color models mentioned above generally suffice for most basic (and some complicated) color

applications in text or text objects. If you need (or want) to use a more specialized color feature not easily done in basic color models, you can browse the description of the color packages on the CTAN page `http://ctan.org/topic/colour` which lists over 40 different packages for special applications; some of these applications extend the basic models, some introduce new capabilities.

Perhaps the most versatile and advanced package based on the LaTeX basic `color` model is the `xcolor` package, which extends the capabilities of `color` with a variety of shades, tones, tints, and arbitrary mixes of colors, plus additional features. Among the many capabilities of `xcolor` are:

- arbitrary tints of predefined (custom) colors
- complete tools for transformation between eight color models
- complement color specification (−red=not red)
- color by wavelength
- define color series
- alternating row colors in tables
- global/local color definitions (to save memory)
- invoke color specification within `pstricks` options, e.g.,
  `\psset{linecolor=[rgb]{0.3,0.5,0.7}}`

To load the `xcolor` package, you must include in your document:
`\usepackage[`*options*`]{xcolor}`

The base (predefined) set of named colors in `xcolor` is:

black blue brown cyan darkgray gray green lightgray lime magenta olive orange pink purple red teal violet white yellow

Additional color names can be loaded with package options.

The color definition syntax in `xcolor` is similar to that in `color`, but with extended syntax allowing for expressions. One of the commands to assign a name to a custom color specification is:
`\definecolor[type]{`*name*`}{`*model-list*`}{`*spec-list*`}`

Example:
`\definecolor{red}{rgb/cmyk}{1,0,0/0,1,1,0}`
defines color name `red` in **rgb** or **cmyk** model depending on which model is currently set.

Two of the packages related to `xcolor`:

`xcolor-material` Defines the 256 colors from the Google Material Color Palette.

`xcolor-solarized` Defines the 16 colors from Ethan Schoonover's Solarized palette.

Even a brief discussion of `xcolor`'s capabilities is beyond the scope of this article, so we refer the interested reader to the full `xcolor` documentation.

My favorite graphics and plotting package to use in LaTeX, `pstricks`, loads the `xcolor` package automatically. Need we say more?

9 Concluding remarks and suggestions

The command `\pagecolor` changes the background color of the entire page until another `\pagecolor` command is seen. Use `\nopagecolor` to change the background color back to normal. There are a number of additional color packages available to color specific environments or create special effects.

Color, when used sparingly and effectively, can make a significant impression on the reader (or consumer) in technical publications (or advertisements). However, it is counterproductive if used improperly or excessively. The only exception is perhaps color used for "art".

While it is tempting (and fun) to use a multitude of colors in a paper or document, it may be wise to bear in mind the following facts and ideas:

- Ask yourself, is it necessary to use color?
- The appearance of color is different on a monitor vs. paper, on different monitors and printers, and to different or color-impaired viewers.
- Pastel colors just never look nice on a white background.
- What looks nice on the author's printer may not look as nice on the reader's printer.
- Most color graphs or charts reproduce badly on a monochrome printer or copier; therefore, they lose their effectiveness.
- Color effects, if included, should be used minimally to enhance clarity only, and to aid in communication of detailed, complex information.
- In most professional publications, all special effects including color should be used sparingly.
- If you are compelled to use colored text to aid the reader, first consider using other "normal" text formats such as enumerate, itemize, italics, bold face; even these formats should be used sparingly and consistently.
- Standard LaTeX "documentclass" styles provide automatic (and appropriate) font selection for all title, sectioning, and text environments.

◇ Behzad Salimi
Ridgecrest, CA
USA
`quadratures (at) gmail dot com`
`https://sites.google.com/site/quadratures/`

SageMathCloud for collaborative document editing and scientific computing

Hal Snyder

Abstract

The open-source platform SageMathCloud (SMC) lets users collaborate in real time on scientific computing and authoring technical documents with standard LaTeX tools. This article offers a survey of key features, presenting SMC as a unique, unified platform for teachers and researchers to be productive in today's mixed software environments and interdisciplinary problem spaces.

1 Introduction

All features of SageMathCloud are available as soon as a free login is created. The following sections offer a tour of major features of SMC.

2 Collaboration

SMC collaboration features apply to both static documents and executable files. Supported file types include LaTeX, Markdown, and HTML documents, as well as Sage worksheets, Jupyter notebooks, and Linux terminal sessions.

Multiple users SMC has simultaneous real-time collaboration with no explicit limit on the number of simultaneous users. Each user can customize the color of their cursor. Figure 1 shows an example of a second user collaborating during an edit session.

Work through network interruptions If your network connection temporarily fails, you can continue editing as long as you want, and your changes will be merged into the live document when you reconnect.

Text and video chat SMC has text chat on the side of each document. Unlike most other chat systems, you can include inline and display MathJax-compatible LaTeX formulas and markdown in the side chat. Users may edit any past chat message. Other collaborators are notified of messages via a bell in the upper right. Figure 2 shows chat with LaTeX beside file views.

Video chat is available as well by clicking a button in the side chat panel.

Course management SMC has an integrated course management system, which makes it easy to fully use LaTeX with students in the context of teaching courses and workshops. For example, you can create a document template with questions, push it to all students, let them work on it, then collect it later, grade it, and return it.

Multiple cursors for each user If a user creates multiple cursors using command- or alt-click, all cursors are visible to other users.

3 Editing documents

Nothing to install Because SMC is a cloud service, there is no software to install locally and projects are reachable from any device with an internet connection. Files can be compiled online to PDF without the need for local installation of LaTeX software.

Document history and backups Every edit (at 2-second resolution) is recorded and stored indefinitely in SMC's backend database. You can browse the history using the TimeTravel view, which also includes a mode showing exactly what changed between two points in time (and who made those changes). Figure 3 shows a TimeTravel view comparing two versions of a file.

SMC stores several hundred read-only snapshots of the complete filesystem state, which users can easily browse. This lets them recover older versions of files that might not have been edited via the web-editor (e.g., edited with vim or emacs via a terminal).

Preview SMC supports online preview, even for documents that are 150 or more pages. It progressively refines the resolution of the preview images and nearby pages. One of the SMC developers wrote the 2016 Cambridge University Press book *Prime Numbers and the Riemann Hypothesis* using SMC [2].

Forward and inverse search SMC supports *inverse search*, which means that by double-clicking on an area on a preview page, the cursor in the text editor jumps to the corresponding location. Similarly, you can jump from a point in the text editor to the corresponding point in the preview via *forward search*.

Editor user interface SMC has a full-featured text editor for LaTeX documents. It has horizontal and vertical split view, which lets you look at two points in the document simultaneously. SMC developers wrote a code folding mode (`github.com/codemirror/CodeMirror/pull/4498`), so one can easily toggle display of sections, subsections, etc. Emacs, Vim, and Sublime keybindings are supported along with many color schemes.

Customizable LaTeX build system SMC fully supports processing very complicated LaTeX documents using custom build systems, including several LaTeX engines — `pdflatex`, `latexmk`,

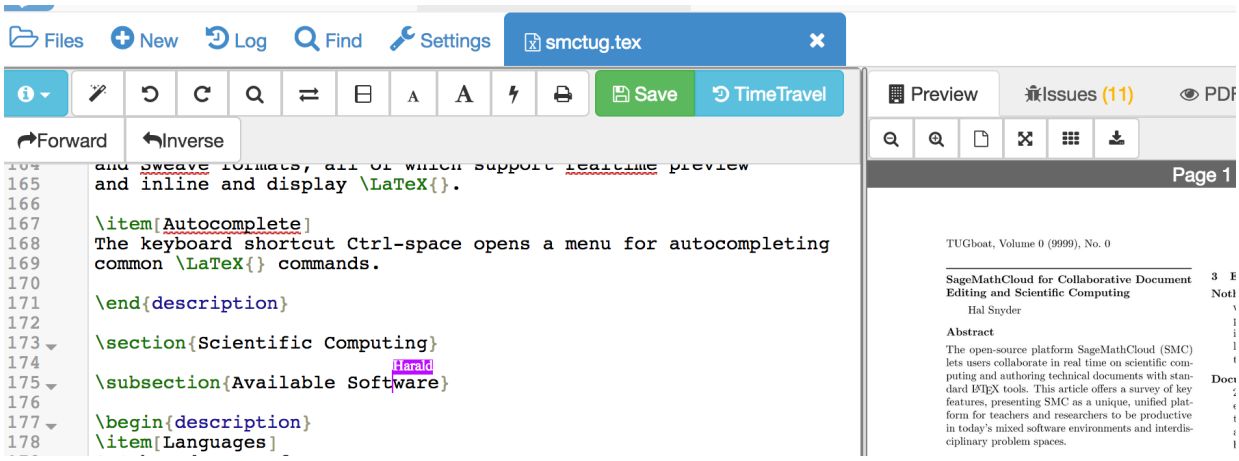


Figure 1: Editing `.tex` file, showing cursor of second user Harald.

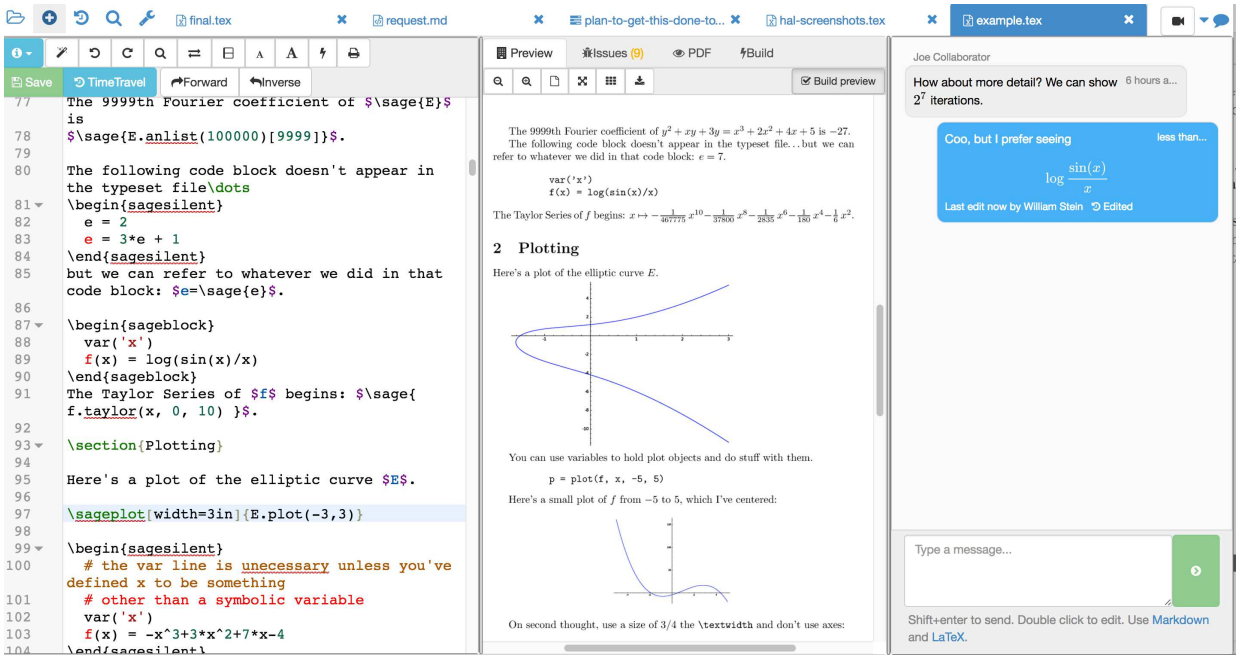


Figure 2: Editable chat beside file views.

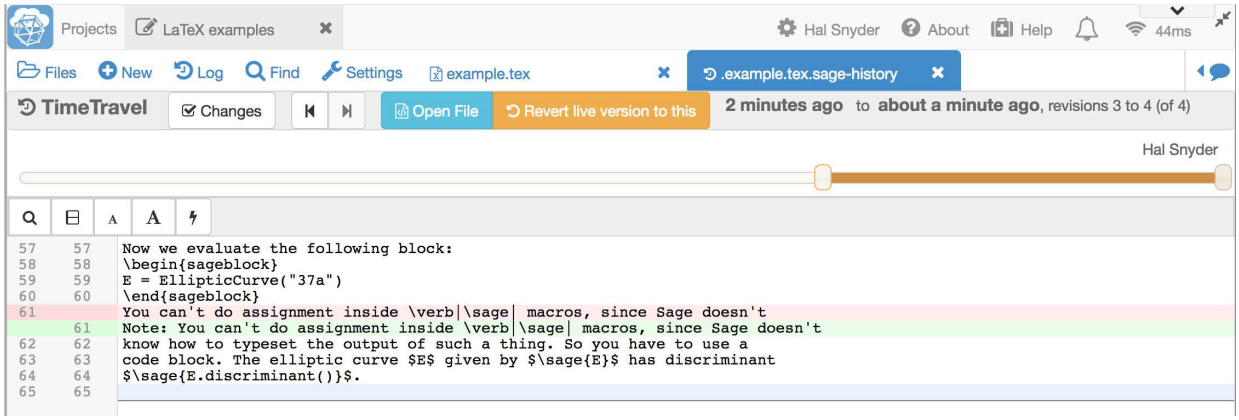


Figure 3: Edit history showing difference between two versions.

and `xelatex` — with most packages preinstalled. Users can easily request additional packages (by clicking the help button), or install them themselves in their local projects. The LaTeX build command is fully customizable, and can involve running arbitrary command sequences, since we offer a full Linux environment. It is, for example, even possible to use GNU `make` to orchestrate the full compilation via a `Makefile`.

Dynamic content Our LaTeX editing environment comes with SageTeX, which makes it easy to add the output of Python (and SageMath!) computations to any LaTeX document. Note that the plots shown in Figure 2 are dynamically generated using SageTeX commands. The figure displays an excerpt from the example file for the `sagetex` package by Dan Drake [1].

Besides SageTeX, SMC supports embedding R code via `knitr` into LaTeX documents. This technique is popular for generating documents with statistical and data science content.

Spell checking In addition to normal spell checking, SMC has TeX-aware spell checking.

Other document formats Side-by-side editing also supports HTML, Markdown, R Markdown, and Sweave formats, all of which support real-time preview and inline and display LaTeX. Figure 4 shows SMC compiling R Markdown.

Autocomplete The keyboard shortcut ctrl-space autocompletes common LaTeX commands.

4 Scientific computing

4.1 Available software

Languages SMC includes dozens of programming languages and thousands of libraries and packages, including computer algebra systems for theoretical mathematics, scientific packages for physical sciences and bioinformatics, and statistical and machine learning software for data science. Here are some of the supported languages: Python, Sage, R, Julia, C, C++, Haskell, Scala.

Packages and environments Popular packages and libraries are provided for each programming language. For Python, the entire Anaconda suite is available. Sage alone includes symbolic and numeric packages, including NumPy, SciPy, matplotlib, Sympy, Maxima, Pari/GP, GAP, R, and Singular. See Figure 5.

4.2 Programming frameworks

Interactive notebooks In the category of interactive notebook computing, SMC offers Sage worksheets and Jupyter notebooks, both of which have chat and TimeTravel and support LaTeX mixed with code. Sage worksheets allow users to easily mix different languages and document formats in different cells of a single worksheet.

Terminal sessions Any number of text-based terminals may be opened to the underlying Ubuntu operating system, providing access to standard Linux tools and command-line interfaces to programming languages.

5 Free and paid accounts

Free accounts provide unlimited projects and collaborators, with 3GB of disk space per project. SMC charges for upgrades, including more disk space, CPU power, and outside network access, which makes it possible to connect to the Internet from within a project in order to push and pull data to remote sites (e.g., GitHub). For more about pricing, see `https://cloud.sagemath.com/policies/pricing.html`

6 Further study

The best way to learn more about SMC is to create a free account at `https://cloud.sagemath.com` and try it out for editing and programming.

There is a wealth of tutorial and reference information at the SMC wiki at `https://github.com/sagemathinc/smc/wiki`.

Articles about new features and system internals can be found at the SMC blog, `http://blog.sagemath.com`. Some of the information in this article appeared in the SMC blog posting [3].

Source code is on GitHub at `https://github.com/sagemathinc/smc`.

References

[1] Dan Drake. `dandrake/sagetex`: embed code, results of computations, and plots from the Sage mathematics software suite (`http://sagemath.org`) into LaTeX documents. `https://github.com/dandrake/sagetex`, 2017. [Online; accessed 2017-02-15].

[2] Barry Mazur and William Stein. *Prime Numbers and the Riemann Hypothesis*. Cambridge University Press, 2016.

[3] Hal Snyder. SMC for Collaborative LaTeX Editing. `http://blog.sagemath.com/latex/2017/02/06/smc-for-latex.html`, 2017. [Online; accessed 2017-02-14].

◇ Hal Snyder
Sagemath, Inc.
`hsnyder (at) sagemath.com`
`https://cloud.sagemath.com/`

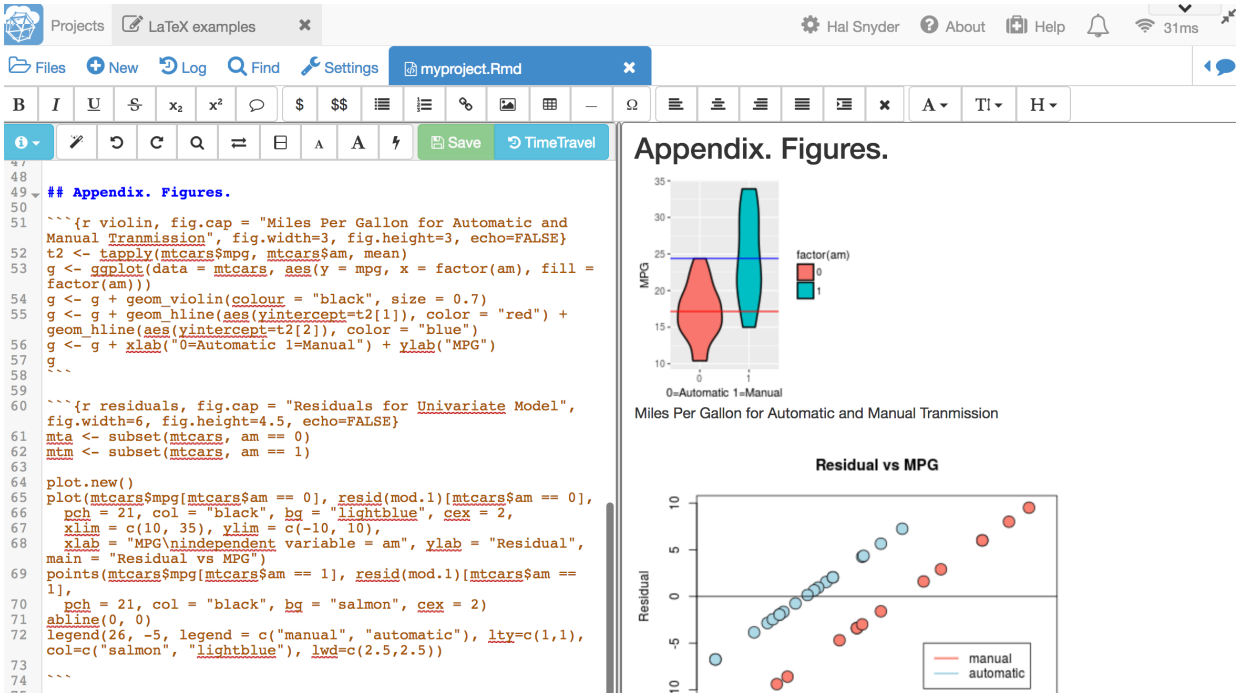


Figure 4: Compiling an R Markdown file in SMC.

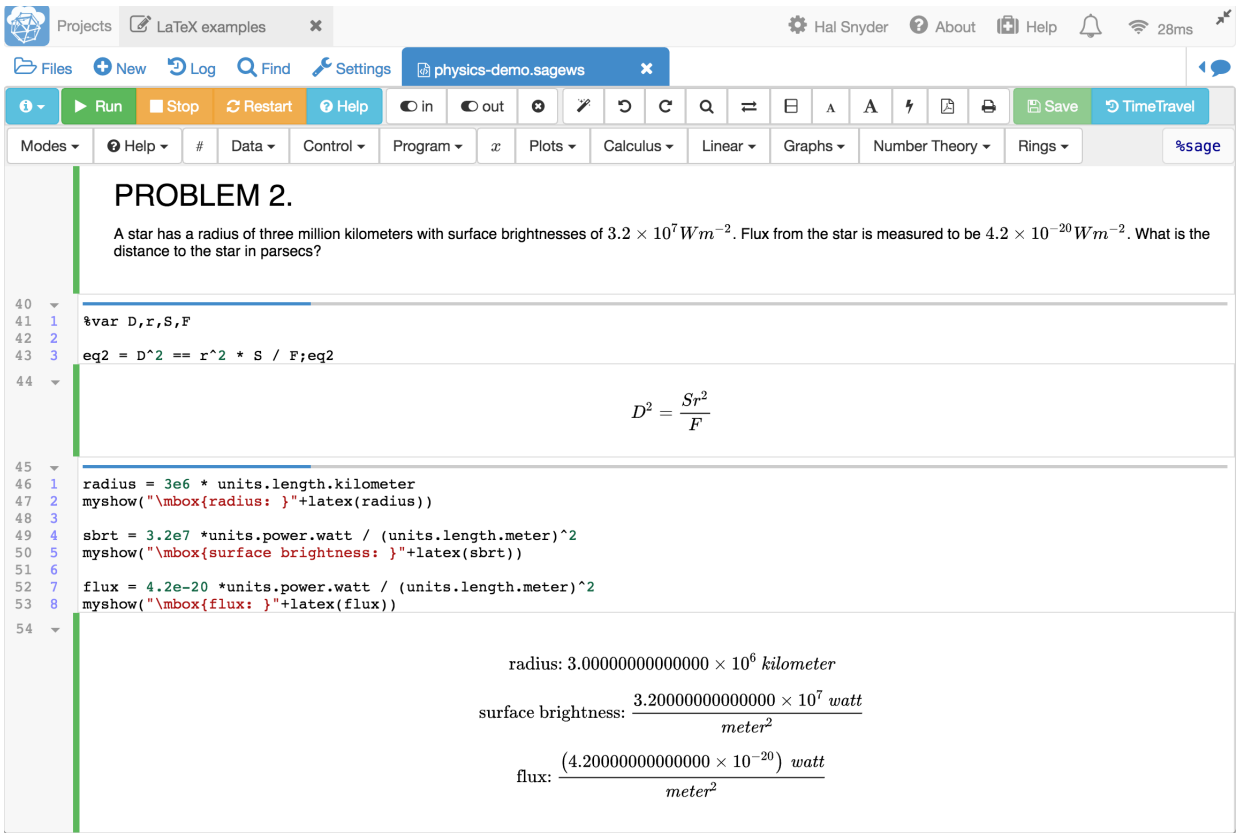


Figure 5: Sage worksheet with LaTeX in markdown, symbolic expressions, and units of measure.

Producing HTML directly from LaTeX — the lwarp package

Brian Dunn

Abstract

The lwarp package allows LaTeX to directly produce HTML5 output, using external utility programs only for the final conversion of text and images. Math may be represented by SVG files or MathJax.

Documents may be produced by LaTeX, LuaLaTeX, or XeLaTeX. A `texlua` script removes the need for system utilities such as `make` and `gawk`, and also supports `xindy` and `latexmk`. Configuration is automatic at the first manual compile.

Print and HTML versions of each document may coexist, each with its own set of auxiliary files. Support files are self-generated on request.

A modular package-loading system uses the lwarp version of a package for HTML when available. Several dozen LaTeX packages are supported with these high-level source compatibility replacements.

A tutorial is provided to quickly introduce the user to the major components of the package.

1 Why LaTeX

Before attempting to justify yet another LaTeX-to-HTML conversion package, it may be worth stepping back for a moment to consider LaTeX itself. A quick web search for "LaTeX vs Word", or some other program, will return many web pages and discussion threads comparing the various programs and their advantages. Things change, however, and many of these discussions are now obsolete due to modern advances in each program's capabilities. As examples, LaTeX no longer has many problems dealing with fonts, and LyX plus a number of integrated development environments are now available, along with online collaborative-development websites [1, 2, 3, 4]. Meanwhile, Word can typeset nice mathematics with a LaTeX-ish input and has improved in its typesetting and stability, and commercial page-layout programs have improved in their handling of large documents.

Nevertheless, many of the traditional advantages of LaTeX still apply: the visibility, stability, and portability of plain-text markup, regular-expression search and replace of both text and formatting commands, easy revision control, the ability to handle large and complex documents, extensive programming capabilities, and the large number of user-supplied packages solving real-world problems. In many cases, it's still faster to type a few arguments than it is to open a dialog box and select and fill in entries, and a powerful programming text editor is usually more responsive than a word processor.

Another development is the large number of markup languages now available, usually with a number of options for output format. These systems are based on plain-text markup using inline tags or sequences of special characters, and thus share some of the advantages of LaTeX. While these systems are useful for smaller documents, cross-referencing is limited (although the AsciiDoc syntax does offer full cross-referencing to figures and tables), much of the customization is done at the back end, and the syntax of special symbols tends to become rather dense once things become complicated. LaTeX has the advantage of giving macros relatively readable names.

Great progress has been made in making LaTeX more widely accessible. Online collaborative LaTeX editing websites now claim a million users and thousands of institutions, and LaTeX is also now available as a browser application [5]. If anything, LaTeX seems to be building momentum, even after all these decades.

2 Why convert LaTeX to HTML

Unfortunately, modern publishing often involves submission and rounds of editing in Word format, conversion to an XML intermediate, then conversion yet again to a professional typesetting system, along with HTML or EPUB versions. Each of these stages may be performed by different groups of people in different parts of the world [6], most of whom are not familiar with the technical content, and also by imperfect algorithms whose programmers haven't thought of every possibility. (Example: An incorrect line break in a superscript, where a hyphen had been used as a minus sign.) The resulting errors are often beyond the author's control — the final product having problems which were not present in the signed-off proof.

While it is unrealistic to expect any of this to change, there is a movement towards self-publishing [7, 8, 9, 10] which removes many of these problems while also providing the benefits of quick turnaround, print on demand, and the ability to make changes or updates as needed. This requires the ability to create a professional-quality printed document in several sizes (e-tablet included), which LaTeX certainly can provide, but also the ability to create HTML or EPUB as well. Providing a high-quality PDF version is better than asking the user to print from HTML, whereas providing an HTML version provides easy accessibility and some search-engine benefits. Providing both is the best option.

Another application of LaTeX-to-HTML conversion is the creation of an informational (non-interactive) website. Many scientists, professors, and engineers would benefit from having their own website on which their own technical papers could be published, and they could apply their pre-existing LaTeX skills not just to the documents but also to the website itself.

3 Why LaTeX is hard to convert

Modern HTML5 and CSS3 are quite capable, to the point where they can be used to produce technical books [11]. Nevertheless, there are some practical problems to overcome in order to create a good conversion from LaTeX to HTML.

One of the first issues is the difference between individual printed pages versus the HTML concept of an endless scroll of variable width. Footnotes can become endnotes, but `\pageref` refers to *what*, exactly? Is `\linewidth` for the current screen size, or is it for a conceptual page size? The relationship between font size, image size, and screen size is broken, there is no margin for marginpars, and text may be reflowed at any time.

LaTeX knows about stretchable space, which is not true of HTML. A `\vfill` is almost meaningless in HTML, and an `\hfill` is not much better. Nor do floating objects translate well, since there are no page breaks at which to place them.

Math in HTML has been a problem for years, and the MathML standard has not been adopted by many browsers [12, 13, 14]. MathJax is nice and getting better all the time, but requires JavaScript and web access or a local copy, possibly making it unacceptable for use in EPUB documents [15], and it can be relatively slow. Drawing math as images has its own limitations.

Aside from display-related issues, another general problem with converting LaTeX to HTML is the fact that LaTeX does not use end delimiters for many of its syntactic units. A `\section` does not have an `\endsection` before the next `\section`, for example, and beginning the next `\section` may first require closing several nested levels worth of currently open subsections and paragraphs. Nor does `\bfseries` have a syntactically defined endpoint, and HTML/CSS do not support state switching.

Finally, LaTeX engines do not allow for the direct plain-text output of HTML tags and text content, thus requiring some kind of PDF-to-text conversion, followed by some system to optionally split the results into separate web pages of HTML, and also copy out any inline images which must be cropped and converted for web display.

4 Existing methods

Several methods already exist for converting some subset of LaTeX into HTML. These are discussed in slightly more detail in the lwarp manual.

The closest to lwarp in design principle is the internet class by Andrew Stacey [16], an interesting project which directly produces several versions of markdown, and also HTML and EPUB.

There is also the TeX4ht project [17], which uses LaTeX itself to do most of the work, along with an external program to convert special codes into HTML or several other formats.

A number of other projects use an intermediate translation program to parse LaTeX source and then convert it externally. See HeVeA [18], TTH [19], GELLMU [20], LaTeXML [21], plasTeX [22], LaTeX2HTML [23], and TeX2page [24], most of which are found on CTAN.

GladTeX [25] may be used to insert LaTeX math expressions into pre-existing HTML code.

For sake of completeness, it should be mentioned that there are plugins allowing the entry of LaTeX math expressions for Word [26, 27] and LibreOffice [28], as well as commercial page-layout programs.

5 Why another approach

Nothing except LaTeX truly understands LaTeX.

More to the point, it's easier for LaTeX to program HTML than for a third-party converter program to understand LaTeX. A larger portion of LaTeX and its associated packages can be parsed and converted when LaTeX itself does the work. Another advantage of staying with LaTeX alone is that development of the core and additional packages can be done without requiring skills in an additional language.

6 Development

6.1 AsciiDoc markup

The initial inspiration for the lwarp package was the internet class. Seeing that someone else had trained LaTeX to produce markup, it was decided to program LaTeX to generate the AsciiDoc markup syntax. AsciiDoc has several advantages over other markup languages, including improved cross-references, and its Asciidoctor variant generating modern HTML5 output. Using AsciiDoc as an intermediate syntax lifted much of the conversion load from LaTeX, while providing almost all of the functionality which would be required for a typical technical paper. Nevertheless, AsciiDoc just couldn't represent many of the concepts commonly-used in LaTeX. Tabular material and minipages were limited, and the toolchain was a

bit of a chore to handle. Thus, the need to program LaTeX to directly produce HTML.

6.2 Low-level and high-level patches

In most cases, code is patched at the lowest level possible, allowing for increased code compatibility and reuse. The process of finding the best place to patch code resulted in several waves of revisions, especially in the areas of floats, auxiliary files, and package handling.

Entire packages must be supported. User-level macros, counters, and so on are intercepted and redirected or ignored as necessary.

6.3 Fonts and encodings

A vector-based font must be used for `pdftotext` to convert the PDF to plain text. A roman face is used in most cases, which preserves em-dashes with `pdflatex`. The HTML tags are printed to the PDF file in a monospaced font, and the quote marks must be upright quotes, but this breaks the em-dash in `pdflatex`.

LaTeX can display many specialized glyphs which are not encoded and thus won't be picked up by `pdftotext`. It may be possible to assign these using `glyphtounicode.tex` or newunicodechar. For many uses `lualatex` or `xelatex` will be preferred, as `pdftotext` can use UTF-8 encoding.

The chosen font will be visible in HTML when rendering math as SVG images.

6.4 Page layout

While generating HTML, a very small font is used and the page layout is changed to allow generous margins. Both are to avoid overflow, which can become a problem with long HTML expressions. Ragged right is used to avoid hyphenation. The `\linewidth` is set for a virtual six-inch wide document, which solves problems where the user specifies a fraction of `\linewidth` for graphics images or tabular columns.

6.5 Paragraph handling

Each paragraph in HTML must be enclosed in an opening and closing tag. To track paragraphs, the `\everypar` hook triggers an action when a paragraph starts, and `\par` is re-assigned to close paragraphs. Flags are used to control whether to turn tag creation on or off in certain circumstances. For example, inside an HTML `<span>` paragraph tags are not allowed, but a `<br>` tag may still be used for something like a multiline caption.

6.6 Sectioning

HTML sectioning requires nesting and unnesting LaTeX sectional units. Since there are no section-ending LaTeX commands, each `\chapter`, `\section`, etc. must first un-nest any previously nested sectional units up to its own level. A simple LIFO stack is used to track section depths and closing tags.

The sectioning code is one area which was rewritten for HTML output, rather than try to reuse something which is patched by so many packages. Section breaks may trigger a new HTML file, and automatic cross-referencing occurs as well. Formatting and paragraph handling depend on which kind of section it is.

6.7 Cross-referencing

While the LaTeX and cleveref cross-referencing code is used, additional referencing is required to track HTML pages and `id` tags. Automatically-generated tags are used for each section and float, allowing cross-references to link to specific objects on each page. Indexing uses the `xindy` program to generate HTML tags.

6.8 Floats

The combination of caption, subcaption, and newfloat packages is supported. These were chosen from among the many alternatives due to being commonly used, flexible, and kept up-to-date. Floats are generated in place, as if they were declared [H]ere. Support is provided for other packages, such as float, floatrow, capt-of, wrapfig, placeins, and the author's own keyfloat which can also support margin floats.

6.9 Image generation

Math, `picture` environments, Ti*k*Z, and anything else with graphic content may be placed inside a special `lateximage` environment. When this environment is started, an HTML open comment tag is generated, followed by a new page. The contents of the graphic environment are then drawn on the empty page, followed by yet another new page whose first line is an HTML closing comment tag. The comment tags encapsulate any text contents of the graphics page such that they are not displayed in the HTML page. Meanwhile, the page and image numbers are written to a text file to be processed by `lwarpmk`, which later separates the PDF file into individual graphics files, each of which is then cropped, converted to SVG, and named, ready for inclusion in the final web page. Finally, HTML instructions are generated to load the resulting graphics file at that position in the web page. Paragraph and formatting elements must be restored to their LaTeX meaning during the creation of the graphic.

6.10 Math

Math may be represented by SVG images using the `lateximage` environment, with the LaTeX source embedded as HTML `alt` tags, or by using the MathJax script.

6.11 Graphics images

Graphics images may be included at a specified width and/or height, or as a fraction of `\linewidth`. When `\linewidth` is used, the assumed six-inch line is used as well, and the final image size is fixed in HTML, along with a `max-width` CSS property to hopefully avoid requiring the user of a hand-held device to pan across the image.

graphicx is emulated quite well, although the HTML standard does not agree with LaTeX about white space while rotating or scaling, so expect ugly results when doing so.

6.12 Minipages

Minipages are created using `inline-flex`, a fairly new CSS3 property which allows side-by-side minipages with a vertical alignment. Unfortunately, a minipage inline with a paragraph of text cannot work since HTML does not allow a block inside a paragraph, so the minipage then goes onto its own line. Furthermore, a `<div>` cannot be used inside a `<span>`, so lwarp disables minipages inside spans, although `\newline` or `\par` can be used to create a `<br>` tag.

For those cases where the user may wish to have an HTML minipage without a fixed width, the new command `\minipagefullwidth` declares that the following minipage may be the natural width of its contents, up to the full width of the display. During print output, the minipage will still use its assigned width.

6.13 Tabular

Tabular material is a challenge, no matter the syntax. This is one area where lwarp had to totally replace the original code rather than try to patch the existing. Data arrays in the computer-science sense had to be used to track column types, as well as actions for `\>`, `\<`, and `\@`. Border-generation logic had to be created. As of this writing vertical rules are not supported, but booktabs are, except for trim options which would be very hard to do in CSS.

6.14 Navigation

In an attempt to avoid resorting to JavaScript, a "sideTOC" has been developed. This is a subset of the table of contents which appears at the side or top of each web page. At present this sideTOC is not in its own pane, which has both its advantages and disadvantages, and this may be changed in the future. To provide for "responsive web design", the sideTOC is moved to the top of the page when the display is narrow, and an additional Home button is placed at the bottom as well.

6.15 Package handling

A major design decision was made regarding handling the loading of additional packages. Some packages may be used as-is, some must be ignored, and some must be patched in some way to be usable for HTML. Furthermore, it would be best if these actions were separated from the lwarp core, interacted well with each other, and expandable by the user.

To provide all of this, lwarp intercepts both the `\usepackage` and `\RequirePackage` macros to first see if there is an lwarp-provided alternative package. If so, that version is used instead of the original. It is up to the lwarp version of the package to either totally ignore the original, or load the original with its options and then perform additional patches or other actions afterward.

Several dozen packages are already supported by lwarp, including some of the most commonly used in all major categories. For packages which lwarp does not yet handle, the user may apply the print-only environment or macro to encapsulate things which do not apply to HTML. The user may also wish to create a custom package for lwarp to use, containing nullified macros and environments, along with any booleans, counters, and lengths which may be used in the source code. Such a package should be named

`lwarp-packagename.sty`

and then lwarp will use it whenever the document calls for `packagename.sty` while creating HTML.

7 Using lwarp

The following is an overview of the configuration and use of lwarp. Major advances have been made in simplifying this process, including the above-mentioned package handling code. As a result, the user may simply add the lwarp-newproject and lwarp packages to the code at the correct place, compile the document once in the traditional way, and then immediately use the `lwarpmk` utility for further print or HTML versions.

7.1 Project setup — lwarp-newproject

Previous versions of lwarp required the user to copy or link a number of configuration files and scripts, and also modify a `makefile`.

Recent improvements include the use of automatic detection of the TeX engine, operating system,

and jobname. These are written to a general configuration file for the new `lwarpmk` program. `lwarpmk` is a utility used to compile print and HTML versions of the document.

Furthermore, the lwarp-newproject package is provided, to be loaded just before lwarp. This package writes various additional configuration and utility files. Included are a project-specific configuration file for the `lwarpmk` utility (thus allowing multiple documents to reside in the same folder), a configuration file for `xindy`, a number of `.css` files, and a fragment of JavaScript used to invoke MathJax.

Also written is a new ⟨*project*⟩`_html.tex` file, whose name is the project's `\jobname` with `_html` appended. This is a small file which simply sets a few options to select HTML conversion, then `\input`s the user's document. In this way, a compile of the user's document generates a print version, while a compile of the `_html` version generates an HTML version. Both versions and their auxiliary files coexist. The lwarp-newproject package is only active when compiling the print version, and the configuration files are regenerated each time the print version is recompiled. Should the user wish to switch TEX engines, the approach is to remove the auxiliary files, then manually recompile the main document using `pdflatex`, `lualatex`, or `xelatex`. This engine will then be used by the `lwarpmk` utility for future compiles of either the print or HTML version.

The `lwarpmk` utility program is to be provided as a LuaTEX executable by the TEX distribution, but it is possible that someone may wish to archive it along with the project. For this purpose, an option for the lwarp-newproject package is available to cause a write of a local copy of `lwarpmk`.

The CSS files include a master `lwarp.css` file which provides the essential functions and a basic LATEX-ish style, along with optional CSS files for a more formal or a more contemporary style. Also created is `sample-project.css`, which shows how to load one of the provided CSS files and also provides a place to make modifications. This file is to be renamed, as it will be overwritten by lwarp-newproject each time a print version is created.

7.2 Compiling the document — `lwarpmk`

Previous versions of lwarp relied on the `make`, `gawk`, and `grep` utilities. Fortunately, modern TEX distributions provide the LuaTEX program — an extension of the Lua programming language. The use of LuaTEX to provide the required utility functions eases issues of availability, installation, and portability.

`lwarpmk`'s configuration file tells it the operating system, the TEX engine, the source `\jobname`, the filename of the homepage, and whether the `latexmk` utility should be used to compile, or whether `lwarpmk` should detect changes and recompile by itself.

`lwarpmk` is able to compile the printed or HTML version of the document, process the index for the printed or HTML version, request a recompile, process the `lateximage` files, clean the auxiliary files, or process the PDF into HTML files (a subset of its functionality, intended to be used by a `makefile` if desired).

If a document name is provided, `lwarpmk` processes that document according to its project-specific configuration file, otherwise it uses its general configuration file to reprocess the last document.

Several utility programs are still required for the HTML conversion. `pdftotext` is used to convert the PDF document into UTF-8 text. `pdfseparate` extracts individual graphic images from the PDF file, `pdfcrop` crops these images, and `pdftocairo` is used to convert PDF images into SVG images. `pdfcrop` is provided as part of the TEX distribution, and the rest are commonly-available utilities from the Poppler project, and should be made available by the operating system's package manager.

7.3 Customizing the HTML

Aside from the CSS files, additional customization is provided by a number of user-adjustable settings and macros.

HTML files may be numbered or named, and a prefix may be applied to each file. The homepage may have its own name. Counters control the depth of the sideTOC and the file division.

Files may be split by the strict sectioning depth level, or higher levels may be combined into one file. For example, a part, its first chapter, and its first section may be combined into one file while further files are split at the section level until the next part or chapter.

The HTML `lang` attribute may be set for the document. The CSS file and HTML `description` may be changed at each file split.

Programming hooks are provided for the top of the home page, the top of other pages, and the bottom of all pages. These are useful for logos, copyright notices, and contact information.

Special environments and macros are provided for functions which should be applied to only the printed or only the HTML versions of the document.

7.4 Tutorial

A tutorial is provided which quickly guides the user through the setup of a document, compiling printed

and HTML versions, processing graphics images, generating math in SVG or MathJax format, customizing the HTML, using `latexmk`, switching the TEX engine, processing multiple documents in the same directory, and cleaning the auxiliary files.

◇ Brian Dunn
bd (at) bdtechconcepts dot com
http://bdtechconcepts.com

References

[1] ShareLATEX. https://www.sharelatex.com

[2] Overleaf. https://www.overleaf.com

[3] Authorea. https://www.authorea.com

[4] Papeeria. https://papeeria.com/

[5] LATEX Base. https://latexbase.com

[6] Forum topic. ResearchGate, "Why is LaTeX not used as an end-to-end solution in the publishing industry?", 2013, https://www.researchgate.net/post/Why_is_LaTeX_not_used_as_an_end-to-end_solution_in_the_publishing_industry

[7] Open Education Database, "The Academic's Guide to Self-Publishing", 2014, http://oedb.org/ilibrarian/the-academics-guide-to-self-publishing/

[8] Dennis Meredith, Research Explainer, "Self-Publishing Series", 2013, https://researchexplainer.com/2013/06/26/self-publishing-series-i-making-the-decision/

[9] Robert Ghrist, "Why I published my mathematics text print-on-demand via Amazon's Createspace", 2014, https://www.math.upenn.edu/~ghrist/whyselfpublish.html

[10] Nicola L. C. Talbot, Dickimaw Books, "Self-Publishing", 2014, http://www.dickimaw-books.com/nonfiction/self-publishing/

[11] Sanders Kleinfeld, "Next-Generation Book Publishing: Of the HTML, by the HTML, for the HTML", Digital Book World, 2014, http://www.digitalbookworld.com/2014/next-generation-book-publishing-of-the-html-by-the-html-for-the-html/

[12] Christian Lawson-Perfect, "Dark days for MathML support in browsers", The Aperiodical, 2013, http://aperiodical.com/2013/11/dark-days-for-mathml-support-in-browsers/

[13] Stephen Shankland, "Google subtracts MathML from Chrome, and anger multiplies", CNET, 2013, https://www.cnet.com/news/google-subtracts-mathml-from-chrome-and-anger-multiplies/

[14] Neil Soiffer, "Microsoft cripples the display of math in IE10 & 11", Design Science News, 2013, http://news.dessci.com/microsoft-cripples-display-math-ie10-11

[15] "EPUB3 Reading systems overview", The MathJax Consortium, 2015, http://docs.mathjax.org/en/latest/misc/epub.html

[16] Andrew Stacey, "latex-to-internet", https://github.com/loopspace/latex-to-internet

[17] TEX4ht: LaTeX and TeX for Hypertext, http://tug.org/tex4ht

[18] HEVEA, http://hevea.inria.fr/

[19] TTH, http://hutchinson.belmont.ma.us/tth/

[20] William F. Hammond, "GELLMU — Introductory Survey", http://www.albany.edu/~hammond/gellmu/

[21] LaTeXML — A LaTeX to XML/HTML/MathML Converter, http://dlmf.nist.gov/LaTeXML/

[22] plasTeX, http://tiarno.github.io/plastex/

[23] LaTeX2HTML, http://www.latex2html.org/

[24] Dorai Sitaram, TeX2page, https://ds26gte.github.io/tex2page/

[25] Martin Gulbrandsen and Sebastian Humenda, GladTeX, https://humenda.github.io/GladTeX/

[26] "LaTeX in Word", https://github.com/EngineeroLabs/latex_in_word

[27] TeXsword, https://sourceforge.net/projects/texsword/

[28] Roland Baudin, "TexMaths, a LaTeX Equation Editor for LibreOffice", http://roland65.free.fr/texmaths/

LaTeX News

Issue 26, January 2017

Contents

ε-TeX

In LaTeX News 16 (December 2003) the team announced

> We expect that within the next two years, releases of LaTeX will change modestly in order to run best under an extended TeX engine that contains the ε-TeX primitives, e.g., ε-TeX or pdfTeX.

and also said

> Although the current release does not *require* ε-TeX features, we certainly recommend using an extended TeX, especially if you need to debug macros.

For many years the team have worked on the basis that users will have ε-TeX available but had not revisited the above statements formally. As of the January 2017 release of LaTeX 2ε, ε-TeX is *required* to build the format, and attempting to build a format without the extensions will fail.

Practically, modern TeX distributions provide the extensions in all engines other than the "pure" Knuth `tex`, and indeed parts of the format-building process already require ε-TeX, most notably some of the UTF-8 hyphenation patterns. As such, there should be no noticeable effect on users of this change.

The team expect to make wider use of ε-TeX within the kernel in future; details will be announced where they impact on end users in a visible way.

Default encodings in XeLaTeX and LuaLaTeX

The default encoding in LaTeX has always been the original 128-character encoding `OT1`. For Unicode based TeX engines, this is not really suitable, and is especially problematic with XeLaTeX as in the major distributions this is built with Unicode based hyphenation patterns in the format. In practice this has not been a major problem as documents use the contributed fontspec package in order to switch to a Unicode encoded font.

In this release we are adding `TU` as a new supported encoding in addition to the previously supported encodings such as `OT1` and `T1`. This denotes a Unicode based font encoding. It is essentially the same as the `TU` encoding that has been on trial with the experimental `tuenc` option to fontspec for the past year.

The XeLaTeX and LuaLaTeX formats will now default to `TU` encoding and `lmr` (Latin Modern) family. In the case of LuaLaTeX the contributed luaotfload Lua module will be loaded at the start of each run to enable the loading of OpenType fonts.

The fontspec package is being adjusted in a companion release to recognise the new encoding default arrangements.

Note that in practice no font supports the full Unicode range, and so `TU` encoded fonts, unlike fonts specified for `T1`, may be expected to be incomplete in various ways. In the current release the file `tuenc.def` that implements the `TU` encoding-specific commands has made some basic assumptions for (for example) default handling of accent commands, and the set of command names is derived from the command names used for the UTF-8 support in the inputenc package, restricted roughly to the character ranges classically provided by the `T1` and `TS1` encodings, but is part of a longer term plan seen over recent releases to increase support for Unicode based TeX engines into the core LaTeX support.

If for any reason you need to process a document with the previous default `OT1` encoding, you may switch encoding in the usual ways, for example

```
\usepackage[OT1]{fontenc}
```

or you may roll back all the changes for this release by starting the document with

```
\RequirePackage[2016/12/31]{latexrelease}
```

\showhyphens in XeLaTeX

Due to the way XeLaTeX interfaces to font libraries, the standard definition of `\showhyphens` does not work. A variant definition has been available in the contributed xltxtra package, however a (slightly different) definition for `\showhyphens` is now included in XeLaTeX by default. As usual this change will be undone if an earlier release is specified using the latexrelease package.

The fixltx2e package

As described in LaTeX News 22, the fixltx2e package has become obsolete with the new update policy. Since 2015 it has just made a warning and exited. In this release we have re-introduced all the code from the original fixes in the 2014 LaTeX but guarded by `\IncludeInRelease{2015/01/01}`. So for current releases fixltx2e still just makes a warning but for old releases, whether that is an old format, or a format with the version date reset via the latexrelease package, the fixes in the original fixltx2e will be applied.

This improves the ability to run old documents in a way compatible with contemporary formats. If you have a 2014 document that used `\usepackage{fixltx2e}` and you add `\RequirePackage[2014/01/01]{latexrelease}` and process it with the current format then latexrelease will undo most changes made since 2014, but now when the document includes fixltx2e it will act like a 2014 version of the package and apply the code fixes, not just give a warning that the package is obsolete.

The latexbug package

As explained in more detail at the LaTeX Project website[1] a new package, latexbug, has been produced to help produce test files to accompany bug reports on the core LaTeX distribution. This is being published separately to CTAN at the same time as this release. By using the latexbug package you can easily check that the packages involved in the test are all part of the core release. The LaTeX project can not handle bug reports on contributed packages, which should be directed to the package maintainer as given in the package documentation.

Updates to amsmath

The amsmath package has two updates at this release.

- The spacing to the left of the `aligned` and `gathered` environments has been fixed: the spurious thin space is no longer added by default. Package options control this to revert to the original behaviour where required; see the amsldoc guide for further details.
- The large delimiters around generalised fractions (for example in the `\binom` construct) did not work in previous releases if using LuaTeX or XeTeX with OpenType math fonts. This is related to the lack of specific metrics for this use in the OpenType Math table. In principle LuaTeX has two additional named metrics to control the delimiters but these are not initialised by default, and in XeTeX it does not seem possible to make them work at all. So for Unicode TeX systems, a new implementation of `\genfrac` is used at this release that uses `\left\right` internally but parameterised to give spacing as close to the original as possible. The implementation in (pdf)TeX is unaffected.

Updates to tools

The array package has been updated to fix a longstanding but previously unreported issue with unwanted interactions between tables in the page head or foot and the body of the page, as reported in PR tools/4488. There is also an update to the LuaTeX support in bm.

An addendum to the release changes in 2015

In 2015 we announced the introduction of the roll-back/roll-forward concept to manage bug fixes and additions to core LaTeX in a manageable way. We also announced at that time that we now incorporate all fixes from fixltx2e into the kernel (as the old mechanism produced problems instead of improving the situation). Refer to `ltnews22.pdf` for details.

One of the fixes from fixltx2e was for a glaring bug in `\addvspace` that was originally detected in the mid-nineties and back then added to the fixltx2e support package. In certain situations `\addvspace` would result in a page/column break below the baseline of the last line. As a result documents using `\flushbottom` would show a clear misalignment (even more prominent when typesetting in two-column mode).

Starting with release 2015/01/01 this is now finally corrected already in the kernel and not only in fixltx2e. In nearly all circumstances this will either make no difference to existing documents, or it will locally improve the visual appearance of that document without changing anything on other pages. However, by the nature of the change it is also possible that there are further non-local changes to the page breaks due to the different break positions introduced by the fix.

Thus, for documents that have been written before 2015 and that should be preserved unchanged at all costs you may have to add

```
\RequirePackage[2014/01/01]{latexrelease}
```

at the top of the document, to roll back the format to a date before the policy change.

[1] `https://www.latex-project.org/bugs/`

LaTeX3 News

Issue 10, November 2016

There has been something of a gap since the last LaTeX3 News, but this does not mean that work has not been going on. The Team have been working on a number of areas, many of which reflect wider take-up of expl3. There have also been a number of significant new developments in the LaTeX3 "sphere" in the last two years.

l3build: Testing LaTeX packages

Testing has been an important part of the work of the team since they assumed maintenance of LaTeX over twenty years ago. Various scripts have been used over that time by the team for testing, but these have until recently not been set up for wider use.

With the general availability of LuaTeX it is now possible to be sure that every TeX user has a powerful general scripting language available: Lua. The team have used this to create a new general testing system for TeX code, l3build. This *is* designed to be used beyond the team, so is now available in TeX Live and MiKTeX and is fully documented. Testing using l3build makes use of a normalised version of the `.log` file, so can test any aspect of TeX output (e.g., by using `\showbox`) or its algorithms (by displaying results in the `.log`).

Part of the remit for creating l3build was to enable the team to work truly cross-platform and to allow testing using multiple TeX engines (earlier systems were limited to a single engine, normally ε-TeX). The new testing system means we are in a much stronger position to support a variety of engines (see below). It has also enabled us to give useful feedback on development of the LuaTeX engine.

As well as the core capability in testing, l3build also provides a "one stop" script for creating release bundles. The script is sufficiently flexible that for many common LaTeX package structures, setting up for creating releases will require only a few lines of configuration.

In addition to the documentation distributed with l3build, the project website [1, publications in 2014] contains some articles, videos and conference presentations that explain how to use l3build to manage and test any type of (LaTeX) package.

Automating expl3 testing

As well as developing l3build for local use, the team have also set up integration testing for expl3 using the Travis-CI system. This means that *every* commit to the LaTeX3 code base now results in a full set of tests being run. This has allowed us to significantly reduce the number of occasions where expl3 needs attention before being released to CTAN.

Automated testing has also enabled us to check that expl3 updates do not break a number of key third-party packages which use the programming environment.

Refining expl3

Work continues to improve expl3 both in scope and robustness. Increased use of the programming environment means that code which has to-date been underexplored is being used, and this sometimes requires changes to the code.

The team have extended formal support in expl3 to cover the engines pTeX and upTeX, principally used by Japanese TeX users. This has been possible in part due to the l3build system discussed above. Engine-dependent variations between pdfTeX, XeTeX, LuaTeX and (u)pTeX are now well-understood and documented. As part of this process, the "low-level" part of expl3, which saves all primitives, now covers essentially all primitives found in all of these engines.

The code in expl3 is now entirely self-contained, loading no other third-party packages, and can also be loaded as a generic package with plain TeX, *etc.* These changes make it much easier to diagnose problems and make expl3 more useful. In particular it can be used as a programming language for generic packages, that then can run without modifications under different formats!

The team have made a range of small refinements to both internals and expl3 interfaces. Internal self-consistency has also been improved, for example removing almost all use of `nopar` functions. Performance enhancements to the l3keys part of expl3 are ongoing and should result in significantly faster key setting. As keyval methods are increasingly widely used in defining behaviours, this will have an impact on compile times for end users.

Replacing `\lowercase` and `\uppercase`

As discussed in the last LaTeX3 News, the team have for some time been keen to provide new interfaces

which do not directly expose (or in some cases even use) the TEX primitives \lowercase and \uppercase. We have now created a series of different interfaces that provide support for the different conceptual uses which may flow from the primitives:

- For case changing text, \tl_upper_case:n, \tl_lower_case:n, \tl_mixed_case:n and related language-aware functions. These are Unicode-capable and designed for working with text. They also allow for accents, expansion of stored text and leaving math mode unchanged. At present some of the interface decisions are not finalised so they are marked as experimental, but the team expect the core concept to be stable.
- For case changing programming strings, \str_upper_case:n, \str_lower_case:n and \str_fold_case:n. Again these are Unicode-aware, but in contrast to the functions for text are not context-dependent. They are intended for caseless comparisons, constructing command names on-the-fly and so forth.
- For creating arbitrary character tokens, \char_generate:nn. This is based on the \Ucharcat primitive introduced by XETEX, but with the ideas extended to other engines. This function can be used to create almost any reasonable token.
- For defining active characters, \char_set_active_eq:NN and related functions. The concept here is that active characters should be equivalent to some named function, so one does not directly define the active character.

Extending xparse

After discussions at TUG 2015 and some experimentation, the team have added a new argument type, e ("embellishment"), to xparse. This allows arguments similar to TEX primitive sub- and superscripts to be accepted. Thus

```
\DeclareDocumentCommand\foo{e{^_}}
  {\showtokens{"#1"}}
\foo^{Hello} world
```

will show

```
"{Hello}{-NoValue-}".
```

At present, this argument type is experimental: there are a number of models which may make sense for this interface.

A new \parshape model

As part of the development of l3galley, Joseph Wright has proposed a new model for splitting up the functions of the \parshape primitive into three logical elements:

- Margins between the edges of the galley and the paragraph (for example an indented block);
- Cut-out sections running over a fixed number of lines, to support "in place" figures and so forth;
- Running or single-paragraph shape.

There are additional elements to consider here, for example whether lines are the best way to model the length of shaping, how to handle headings, cut-outs at page breaks, *etc.*

Globally optimized pagination of documents

Throughout 2016 Frank Mittelbach has worked on methods and algorithms for globally optimizing the pagination of documents including those that contain floats. Early research results have been presented at BachoTEX 2016, TUG 2016 in Toronto and later in the year at DocEng'16, the ACM Symposium on Document Engineering in Vienna. A link to the ACM paper (that allows a download free of charge) can be found on the project website [1]. The site also holds the speaker notes from Toronto and will host a link to a video of the presentation once it becomes available.

The framework developed by Frank is based on the extended functionality provided by LuaTEX, in particular its callback functions that allow interacting with the typesetting process at various points. The algorithm that determines the optimal pagination of a given document is implemented in Lua and its results are then used to direct the formatting done by the TEX engine.

At the current point in time this a working prototype but not yet anywhere near a production-ready system. However, the work so far shows great potential and Frank is fairly confident that it will eventually become a generally usable solution.

Looking forward

The LuaTEX engine has recently reached version 1.0. This may presage a stable LuaTEX and is likely to result in wider use of this engine in production documents. If that happens we expect to implement some of the more complex functionality (such as complex pagination requirements and models) only for LuaTEX.

References

[1] Links to various publications by members of the LATEX Project Team. https://www.latex-project.org/publications.

A key/value interface for generating LaTeX floats — the keyfloat package

Brian Dunn

Abstract

The keyfloat package provides a key/value user interface for quickly creating figures with a single image each, figures with arbitrary contents, tables, subfloats, rows of floats, floats located [H]ere, floats in the [M]argin, and floats with text [W]rapped around them, using a consistent syntax for all.

Key/value combinations may specify a caption and label, a width proportional to `\linewidth`, a fixed width and/or height, rotation, scaling, a tight or loose frame, an `\arraystretch`, a continued float, additional supplemental text, and an artist/author's name with automatic index entry. When used with the tocdata package, the name also appears in the List of Figures.

Floats may be placed into a multi-row environment, and are typeset to fit within the given number of columns, continuing to the next rows as necessary. Nested sub-rows may be used to generate layouts such as two small figures placed vertically next to one larger figure.

Subfloats are supported by two environments.

As an example, a typical command to include a figure with a framed image of half `\linewidth` could be:

```
\keyfig*[hbp]{f,lw=.5,c={A caption},
    l={fig:label}}{image}
```

1 A problem with floats

When including a figure with a graphics image into a document, the user typically enters something such as:

```
\begin{figure}
\centering
\includegraphics[width=3in]{filename}
\caption{A Figure}
\label{fig:somelabel}
\end{figure}
```

When doing that often enough, it makes sense to factor the common code:

```
\onefigure[3in]{filename}{A Figure}
    {fig:somelabel}
```

Expanding the capability of `\onefigure` via the xparse package can lead to the general case of:

```
\onefigure*[loc](width){filename}
    (add'l text)
    [shortcap]{caption}*[label]
```

Attempting to add additional features such as frames and continued floats hits the limit of nine parameters for a TeX macro, requiring that new features use some kind of change-state macros instead. Attempting to support rows of floats or subfloats only makes things more complicated still.

A key/value system solves the problem of adding more features, does not require much additional typing, is a more self-documenting syntax, and allows a shared syntax with subfloats and groups of floats as well. Thus, the keyfloat package.

2 The keyfloat package

Using keyfloat, the previous example becomes:

```
\keyfig{w=3in,c=A Figure,l=fig:somelabel}
    {filename}
```

The `\onefigure` general case becomes:

```
\keyfig*[loc]{w=width,t={add'l text},
    sc=shortcap,cstar=caption,
    l=label}{filename}
```

2.1 Macros and environments

keyfloat provides macros and environments to create figures and floats:

`\keyfig*[`⟨*loc*⟩`]{`⟨*keys*⟩`}{`⟨*image*⟩`}`
A figure with an image.

`\keyfigbox*[`⟨*loc*⟩`]{`⟨*keys*⟩`}{`⟨*contents*⟩`}`
A figure with arbitrary contents.

`\keyparbox*[`⟨*loc*⟩`]{`⟨*keys*⟩`}{`⟨*contents*⟩`}`
A "figure" without a caption, useful to place uncaptioned text inside a group.

`\keytab*[`⟨*loc*⟩`]{`⟨*keys*⟩`}{`⟨*tabular*⟩`}`
A table.

`keyfigure*[`⟨*loc*⟩`]{`⟨*keys*⟩`}`
A figure environment.

`keytable*[`⟨*loc*⟩`]{`⟨*keys*⟩`}`
A table environment.

2.2 Groups of floats and subfloats

Floats may be gathered into groups, as well as gathered into a subfloat, using these environments:

`keyfloats*[`⟨*loc*⟩`]{`⟨*#cols*⟩`}`
A group of rows and columns of floats.

`keysubfigs*[`⟨*loc*⟩`]{`⟨*#cols*⟩`}{`⟨*keys*⟩`}`
A figure containing a group of rows and columns of subfigures.

`keysubtabs*[`⟨*loc*⟩`]{`⟨*#cols*⟩`}{`⟨*keys*⟩`}`
A table containing a group of rows and columns of subtables.

2.3 Margin float

The tufte-book class offers margin floats. These are used if they are available, otherwise keyfloat provides its own:

`marginfigure[`⟨*offset*⟩`]`
: A figure environment placed into the margin.

`margintable[`⟨*offset*⟩`]`
: A table environment placed into the margin.

2.4 Arranging floats

Rows and columns of floats are created by enclosing `\keyfig` and friends inside a `keyfloats` environment. The number of columns is given, and the floats are dynamically arranged across each row, with leftovers distributed evenly on the last row. These may be nested (Figures 1 to 5, and Table 1).

```
\begin{keyfloats}{2}
\keyfig{lw=1,f,c={First in a group},
    l=fig:firstinrow,
    tl={\cs{raggedright} text}
    }{image}
\keyparbox{}{\centering
    A \cs{keyparbox} describing something.
    \par With several paragraphs.}
\begin{keyfloats}{2}
\keyfig{lw=1,c={Third in a group},
    l=fig:thirdinarow}{image}
\keyfig{lw=1,c={Fourth in a group}}{image2}
\keyfig{lw=1,c={Fifth in a group}}{image}
\keyfig{lw=1,c={Sixth in a group},
    l=fig:sixthinrow}{image2}
\end{keyfloats}
\keytab{c={Seventh in a group},
    l=tab:seventhinrow}
    {\testwidetable}
\end{keyfloats}
```

Subfloats are arranged into rows and columns in a similar manner (Fig. 6). Notice that fig. 6(d) is a foreign table inside a figure.

```
\begin{keysubfigs}{3}
  {c=Subfigures,l=fig:subfigs}
\keyfig{lw=1,f,c={First subfigure},
  l=fig:firstsubfig,t=Some text}{image}
\keyfig{lw=1,f,r=90,c={Second subfigure},
  l=fig:secondsubfig,
  t=Lots of lots of lots of lots of text.}
  {image2}
\begin{keyfloats}{1}
\keyfig{lw=1,f,c={Third subfigure},
  l=fig:thirdsubfig}{image}
\keytab{c={Fourth subfigure},
  l=fig:fourthsubfig}{\testtable}
\keyfig{lw=.5,f,c={Fifth subfigure},
  l=fig:fifthsubfig}{image}
\end{keyfloats}
\end{keysubfigs}
```

\raggedright text

Figure 1: First in a group

A \keyparbox describing something.

With several paragraphs.

Figure 2: Third in a group

Figure 3: Fourth in a group

Table 1: Seventh in a group

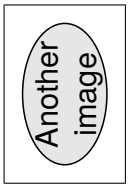
A	B	C
D	E	F

Figure 4: Fifth in a group

Figure 5: Sixth in a group

Some text

(a) First subfigure



Lots of lots of lots of lots of text.

(b) Second subfigure

(c) Third subfigure

(d) Fourth subfigure

A	B
C	D

(e) Fifth subfigure

Figure 6: Subfigures

2.5 Placement of floats

Floats or groups of floats may be placed `[H]`ere, in the `[M]`argin, with text `[W]`rapped around them (wrapfig with optional placement), or with the usual `[htbp]` placement combinations.

Starred floats may be used to create two-column floats.

Mr. First Last III

About the illustration.

Figure 7: An artist's work

2.6 Options controlled by key/values

Most of the keys are one or two letters long, allowing them to be entered quickly.

Continued floats are available to repeat the previous float's number.

Tabular `\arraystretch` may be set per table.

An image inside a figure may be sized, rotated, and placed inside a tight, loose, or custom frame. Boxes of arbitrary contents may be sized and framed. Along with a fixed width or height, contents may also be sized as a fraction of `\linewidth`. Doing so allows them to automatically scale appropriately as they are moved into or out of groups of floats or subfloats.

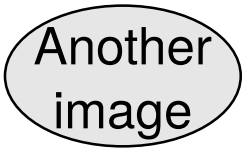
Additional descriptive text may be placed inside the float with left/right/center alignments, and an artist/author's name may be added as well (Fig. 7):

```
\keyfig{ft,lw=1,
  ap=Mr.,af=First,al=Last,as={~III},
  tc={\textit{About the illustration.}},
  c=An artist's work,l=fig:artist}{image}
```

Subfloats may be used to create a collection (Fig. 8):

```
\begin{keysubfigs}{2}{
  c=Artist's collection,
  l=fig:artistcollection,
  t={Some fully-justified text just
  for illustrative purposes, in case you
  have a use for long explanations.
  This text may be the full \cs{linewidth}
  in size. \par
  Multiple paragraphs of text are
  allowed.},
  ap=Prefix,af=First,al=Last,as={, Suffix}
}
  \keyfig{lw=1,c=Artist's First Work}
    {image}
  \keyfig{lw=1,c=Artist's Second Work,
    tl={Commentary about the work.}}
    {image2}
\end{keysubfigs}
```

(a) Artist's First Work



Commentary about the work.

(b) Artist's Second Work

Prefix First Last, Suffix

Some fully-justified text just for illustrative purposes, in case you have a use for long explanations. This text may be the full `\linewidth` in size.

Multiple paragraphs of text are allowed.

Figure 8: Artist's collection

2.7 Customizations

User-redefinable macros are provided for tight and loose frames. A loose frame is meant to add a bit of margin around the object, such as a closely cropped diagram, and is the usual case. A tight frame is useful around a photograph, giving a visual definition to its edge. The user must set a certain LaTeX length for each type of frame, equal to the total width of each frame and margin. These lengths are used to compute the final size of the float contents.

The caption package is used by keyfloat, and customized caption settings may be used for figures, tables, subfigures/tables, and wrapped figures/tables.

As usual, `\floatsep` and `\dblfloatsep` may be used to spread out the floats on the page.

2.8 Examples

The keyfloat documentation has more than thirty examples demonstrating code fragments and the corresponding results, as well as solutions for several special cases, such as frames using mdframed and fancybox.

◇ Brian Dunn
bd (at) bdtechconcepts dot com
http://bdtechconcepts.com

Glisterings: Hanging; Safety in numbers

Peter Wilson

> A glisterin mornin aften draws tae rain.
>
> ANONYMOUS

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

> We must indeed all hang together, or most assuredly, we shall all hang separately.
>
> *Spoken at the signing of the Declaration of Independence*, BENJAMIN FRANKLIN

1 Hanging

1.1 Overhangs

Rui Maciel asked about a notation for the closure of a set, saying that [8]:

When I need to refer to the closure of a set I tend to use the `\bar{}` *command, So, considering the set* Ω *then the closure of that set would be:*

```
$\bar{\Omega}$ ->
```
$\bar{\Omega}$

However, I've noticed that when the symbol used to reference a given set also has a superscript then $\bar{\Omega}^s$ *doesn't look very good. I've also tried* `\overline{}` *instead but it appears even worse*

```
$\overline{\Omega^{s}}$ ->
```
$\overline{\Omega^{s}}$.

Enrico Gregorio recommended [3]:

```
\newcommand*{\closureG}[2][3]{%
 {}\mkern#1mu\overline{\mkern-#1mu#2}}
```

while Bill Hammond said [5] that he found the following to work better, also noting that he used amsmath:

```
\newcommand*{\closureH}[2][3]{%
  \overline{{}\mkern#1mu#2\mkern-#1mu}}
```

In each case the optional argument is the value of an `\mkern` (in mu) applied to the overline to move it sideways; the default is 3.

Dan Luecking felt [7] that there should be two controls over the overline — one to shift the line (which is provided by the previous macros) and a second to adjust the length of the line — and suggested the `\closureL` macro.

```
%% \closureL{right shift}{trim}{symbol}
\newcommand*{\closureL}[3]{%
  \mkern#1mu\mkern#2mu
  \overline{\mkern-#1mu \mkern-#2mu #3%
            \mkern-#2mu \mkern#1mu}%
  \mkern#2mu\mkern-#1mu}
```

Table 1 shows the results of applying the three closure macros to a variety of variables with a range of kerns, along with the result of a vanilla `\overline`.

There is no one ideal value for the `\mkern`; a 'good' value depends on whether the set variable is upright (e.g., Ω) or slanted (e.g., B) and whether or not it has a super- or subscript. Basically it comes down to what you think is most appropriate. In my view I prefer the following:

Upright variable (e.g. Ω**)** `\closureG[0]{}`, `\closureH[0]{}`, `\closureL{0}{0}{}`, which are all equivalent to `\overline{}`

Slanted variable (e.g., B**)** `\closureG[3]{}`, `\closureH[-3]{}`.

I think that `\closureL{0}{3}{}` and `\closureL{3}{0}{}` are close but not quite as good. Something like `\closureL{1}{2}` would seem to give a better result.

As the old saying goes, 'Yer pays yer money and takes yer choice'.

1.2 Paragraphs in equations

'Cooch' wrote [2]:

In a number of the chapters for one of my books, I 'define' a series variables, generally embedded in the form

variable = text to define the variable

For example (the page I'm currently staring at)

```
$\phi^{rst}_{i-1,i}$= the probability that
a particle in state \emph{r} at time
\emph{i}-1 and state \emph{s} at time
\emph{i} is in state \emph{t} at time
\emph{i}+1.
```

. . . I want to force the RHS of the expression to 'wrap' and be indented after the first sentence, to the right of the equal sign. So, that the above looks like:

```
$\phi^{rst}_{i-1,i}$= the probability that
                      a particle in state ...
```

In other words, something analogous to a 'hanging indent' after the first line, but where the indentation is relative to where the equal sign falls.

There were several responses to this and for the following, in order to save space and make the examples easier to read, I have defined

```
\newcommand*{\mathdef}{\phi^{rst}_{i-1,i}=}
\newcommand*{\textdef}{the probability that
  a particle in state $r$ at time $i-1$
  and state $s$ at time $i$ is in state
  $t$ at time $i+1$.}
\newcommand*{\smath}{D_{n}=}
\newcommand*{\stext}{the definition of the
  variable as used herein.}
```

All the respondents disagreed with the use of `\emph` to indicate a math variable. As Enrico Gregorio said [4]:

Table 1: Various closures

‘closure’	Ω	Ω^*	Ω_s	B	B^*	B_s
`\overline{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureG[-3]{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureG[0]{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureG[3]{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureH[-3]{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureH[0]{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureH[3]{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{-3}{-3}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{-3}{0}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{-3}{3}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{0}{-3}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{0}{0}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{0}{3}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{3}{-3}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{3}{0}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$
`\closureL{3}{3}{...}`	$\overline{\Omega}$	$\overline{\Omega^*}$	$\overline{\Omega_s}$	$\overline{B}$	$\overline{B^*}$	$\overline{B_s}$

Don’t use `\emph{i}` for representing a variable: it’s simply `$i$`; notice also the difference between `$i-1$` ($i-1$) and `\emph{i}-1` (*i*-1); the second one is definitely wrong.

Lars Madsen said [9] that he normally used[1]

```
\begin{equation*}
\phi^{rst}_{i-1,i}=
\parbox[t]{<length>}{\raggedright the
probability that a particle in state $r$
at time $i-1$ and ...}
\end{equation*}
```

which, with ⟨*length*⟩ = `0.8\columnwidth` results in

$$\phi^{rst}_{i-1,i} = \text{the probability that a particle in state } r \text{ at time } i-1 \text{ and state } s \text{ at time } i \text{ is in state } t \text{ at time } i+1.$$

Enrico Gregorio provided [4] the following environment:

```
\newenvironment{qdesc}[1]%
  {\par\addvspace{\medskipamount}
   \sbox{0}{$#1$ }\dimen0=\textwidth
   \advance\dimen0 by -\wd0
   \noindent\usebox{0}
   \begin{minipage}[t]{\dimen0}}%
  {\end{minipage}
   \par\addvspace{\medskipamount}}
```

which applied to the example as

```
\begin{qdesc}{\mathdef}
\textdef
\end{qdesc}
```

and results in:

$$\phi^{rst}_{i-1,i} = \text{the probability that a particle in state } r \text{ at time } i-1 \text{ and state } s \text{ at time } i \text{ is in state } t \text{ at time } i+1.$$

Jean-François Burnol presented [1] the following macro:

```
\newcommand{\start}[1]{%
  \setbox0=\hbox{#1}%
  \hangindent\wd0%
  \noindent\box0}

\start{$\phi^{rst}_{i-1,i}$= }\textdef
```

$$\phi^{rst}_{i-1,i} = \text{the probability that a particle in state } r \text{ at time } i-1 \text{ and state } s \text{ at time } i \text{ is in state } t \text{ at time } i+1.$$

Jean-François’ `\start` macro is a TEX version of the LATEX kernel’s command `\@hangfrom`, which the memoir class provides as a user-level macro `\hangfrom{text}` by copying the original definition:

```
\newcommand{\hangfrom}[1]{%
  \setbox\@tempboxa\hbox{{#1}}%
  \hangindent \wd\@tempboxa%
              \noindent\box\@tempboxa}
```

`\hangfrom{<text>}` puts text in a box and makes a hanging paragraph of the following material (somewhat like a description item). Applying this:

```
\hangfrom{$\mathdef$ }\textdef \par
\hangfrom{$\smath$ }\stext
```

[1] `equation*` is from the amsmath package

$\phi^{rst}_{i-1,i}$ = the probability that a particle in state r at time $i-1$ and state s at time i is in state t at time $i+1$.
D_n = the definition of the variable as used herein.

As the last example shows, each 'definition' is treated individually. If it is required that, say, several definition texts should be aligned in a set of definitions then using one of the `tabular` environments could be an advantage. For example:

```
Using \texttt{tabular} \\
\begin{tabular}{lp{0.7\columnwidth}}
$\mathdef$ & \textdef \\
$\smath$ & \stext \\
\end{tabular}

Using \texttt{tabularx} \\
\begin{tabularx}{\linewidth}{lX}
$\mathdef$ & \textdef \\
$\smath$ & \stext \\
\end{tabularx}
```

Using `tabular`

$\phi^{rst}_{i-1,i}$ =	the probability that a particle in state r at time $i-1$ and state s at time i is in state t at time $i+1$.
D_n =	the definition of the variable as used herein.

Using `tabularx`

$\phi^{rst}_{i-1,i}$ =	the probability that a particle in state r at time $i-1$ and state s at time i is in state t at time $i+1$.
D_n =	the definition of the variable as used herein.

Some of the suggestions require a length to be specified for the definition text while others automatically use all the available space. It is really a matter of individual preference which is most suited for a particular desired outcome.

> It was a bright cold day in April, and the clocks were striking thirteen.
>
> *Nineteen Eighty-Four*, George Orwell

2 Safety in numbers

Gordon Haverland posted [6] to the `texhax` mailing list:

I don't suppose there is some easy way to deal with superstitions in LaTeX? I looked around CTAN a bit and nothing jumped out at me.

I suspect the company I am doing work for is superstitious, or customers are. But I ran across an enumerated list where there is no 13th element. What I've done is:

13. Purposely blank.

But is there something else that is more universal?

Heiko Oberdiek responded [10] with a universal solution by changing the definition of `\@arabic`, which is the underlying LaTeX macro for typesetting the value of a counter in arabic form:

```
\makeatletter
\newcommand*{\safe}{%
\renewcommand*{\@arabic}[1]{%
\ifnum##1=13\relax
  12a%
\else
  \ifnum##1=-13\relax
    -12a%
  \else
\expandafter\@firstofone\expandafter{\number##1}%
  \fi
\fi}}
\makeatother
```

Following the `\safe` declaration *every* setting '13' will be typeset as '12a'.

To save space in the following examples I have defined:

```
\makeatletter
\newcommand*{\setlistctr}[1]{%
  \setcounter{\@listctr}{#1}%
  \protected@edef\@currentlabel
    {\csname p@\@listctr\endcsname
     \csname the\@listctr\endcsname}}
\makeatother
```

which can be used to reset the `enumerate` list counter.

Applying Heiko's suggestion to an `enumerate` list as:

```
Standard enumeration:
\begin{enumerate}
\item One \par
\ldots \setlistctr{11}
\item Twelve
\item Thirteen
\item Fourteen
\end{enumerate}

'Safe' enumeration:
\begin{enumerate}\safe
\item One \par
\ldots \setlistctr{11}
\item Twelve
\item Thirteen
\item Fourteen
\end{enumerate}
```

the result is:

Standard enumeration:

1. One

   ...

12. Twelve
13. Thirteen
14. Fourteen

‘Safe’ enumeration:

1. One
   ...
12. Twelve

12a. Thirteen

14. Fourteen

However Gordon had explicitly mentioned the `enumerate` list and I thought that perhaps something specific for that would suit. To that end I defined the `\skipit` macro that ensures that the counter in an `enumerate` skips the value ‘13’, and the macro `\fixitem` to append it to the end of LaTeX‘s internal `\@item` macro.

```
\makeatletter
\newcommand*{\skipit}{%
  \if@nmbrlist
    \ifnum12=\csname c@\@listctr\endcsname
      \refstepcounter\@listctr
    \fi
  \fi}
\let\old@item\@item
\newcommand{\fixitem}{%
  \def\@item[##1]{\old@item[##1]\skipit}}
\makeatother
```

An example of this approach is:

```
‘skipit’ enumeration:
\begin{enumerate}\fixitem
\item One \par
\ldots \setlistctr{11}
\item Twelve
\item Thirteen
\item Fourteen
\end{enumerate}
```

‘skipit’ enumeration:

1. One
   ...
12. Twelve
14. Thirteen
15. Fourteen

With a second level list, though, you might not get what you expect:

```
Standard enumeration:
\begin{enumerate}
\item Including a ‘skipit’ enumeration:
\begin{enumerate}\fixitem
\item One \par
\ldots \setlistctr{11}
\item Twelve
\item Thirteen
\item Fourteen
\end{enumerate}
\item Two \par
\ldots \setlistctr{11}
\item Twelve
\item Thirteen
\item Fourteen
\end{enumerate}
```

Standard enumeration:

1. Including a ‘skipit’ enumeration:
   (a) One
   ...
   (l) Twelve
   (n) Thirteen
   (o) Fourteen
2. Two
   ...

12. Twelve
13. Thirteen
14. Fourteen

References

[1] Jean-François Burnol. Re: Variable definitions / indenting on = sign. `comp.text.tex`, 25 March 2011.

[2] cooch17. Variable definitions / indenting on = sign. `comp.text.tex`, 25 March 2011.

[3] Enrico Gregorio. Re: Math notation for the closure of a set? `comp.text.tex`, 30 March 2011.

[4] Enrico Gregorio. Re: Variable definitions / indenting on = sign. `comp.text.tex`, 25 March 2011.

[5] William Hammond. Re: Math notation for the closure of a set? `comp.text.tex`, 16 April 2011.

[6] Gordon Haverland. [texhax] superstitions. `texhax` mailing list, 23 April 2011.

[7] Dan Luecking. Re: Math notation for the closure of a set? `comp.text.tex`, 19 April 2011.

[8] Rui Maciel. Math notation for the closure of a set? `comp.text.tex`, 30 March 2011.

[9] Lars Madsen. Re: Variable definitions / indenting on = sign. `comp.text.tex`, 25 March 2011.

[10] Heiko Oberdiek. Re: [texhax] superstitions. `texhax` mailing list, 23 April 2011.

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

The optimal value for \emergencystretch

Udo Wermuth

Abstract

As a reaction to authors using very high values for the integer parameter \tolerance in their texts instead of rewriting them if TEX creates overfull lines, Donald E. Knuth introduced in TEX 3.0 the dimen \emergencystretch and a third pass for TEX's line-breaking algorithm. The parameter should only be used in an emergency situation but such a situation can occur, for example, if a typist cannot change the text written by an author and TEX produces overfull lines. Then this parameter comes to the rescue although the output might not look good, as spaces can spread much more than before. This article tries to find all factors that have an impact on the value of \emergencystretch. Besides pure theory, experiments are performed and the impact factors are briefly discussed.

1 Introduction

Since its introduction with TEX 3.0 in 1989 [8, p. 327] the parameter \emergencystretch has asked for attention as no hints about useful values are given in the article. Of course, it could be used to avoid awful lines in a better way than before: "[P]eople [. . .] tried to do this by setting \tolerance = 10000, but the result was terrible because TEX would tend to consolidate all the badness in one truly horrible line." So its usage is a "better way to avoid overfull boxes, for people who don't want to look at their documents to fix unfeasible line breaks manually."

The TEXbook describes this dimen [3, p. 107] and gives its default value in the plain format, 0 pt. I assume that — whatever research went into the design and the implementation — the idea remains that authors should rewrite their texts if they cannot be broken by TEX. The dimen is an instrument for people who have to work with a text which cannot be changed by them.

The number of relevant publications about this parameter appears to be very small. In the first years after its introduction the LATEX3 group looked for volunteers to make experiments with the parameter [12, p. 511]. But it seems that no one has published such experiments or research. A check with several well-known textbooks to find a hint about useful settings and not merely the description of this parameter resulted in just one hit: In [1], p. 333, it is written that "a likely value seems to be around 5 pt." A search in *TUGboat* archives produces again only one hit: In [13], p. 139, the following statement is made: "However, a sound rule of thumb is to set it to 1 em (based on the primary text font); this value, strange as it may seem, appears equally suitable for both wide and narrow measures." (His "pragmatic approach" can result in high values for \tolerance.)

Of course a single value cannot always solve the problem and make TEX find line breaks without overfull lines. It might nevertheless be seen as an upper bound on what is seen as tolerable for the visual output. A value of 5 pt appears to be small but in a line with only one stretchable interword space it creates a hole in the text as it stretches the space to twice its maximal value in cmr10. Therefore, whatever value for the parameter \emergencystretch is used, the output should be checked to see if it does not create awful looking lines. The author of a text should think about a textual change before eliminating overfull lines with \emergencystretch.

Here is a short overview of the structure of this article: Section 2 sketches briefly and somewhat vaguely aspects of glue and establishes the notation that helps to write about spaces and stretchability. Section 3 presents an experiment with many overfull lines and shows how they can be removed by assigning appropriate values to the dimension \emergencystretch. The situation is analyzed from a theoretical point of view in section 4. Section 5 applies the formula created in section 4 to some cases of experiment 1 and presents numerical results as well as ideas for TEX macros to do the calculations. The next three sections look at the various parameters that occur in the formula for \emergencystretch stated in section 4. The theoretical results are extended in section 9 and applied to a second experiment in section 10. Removing overfull lines is only one application of \emergencystretch; another is discussed in section 11. The last section summarizes the results and gives a rule of thumb for the calculation of \emergencystretch.

2 Notational conventions

A paper about the dimension \emergencystretch should explain the material that is able to stretch or shrink: glue. Therefore the basic principles of glue in texts and in math mode are reviewed first. To handle all the different forms a consistent notation for numbers, dimensions, and skips is needed. So let's start with some notational conventions and definitions.

Conventions. I use the following notation for variables and functions; variables are written in lowercase, functions in uppercase letters:

1. math italic Latin for dimensions, the most frequently used variables and functions;
2. numbers are written with Greek letters;
3. boldface Latin is used for glue, i.e., a triple of dimensions, for example, $\mathbf{g} = (g^\circ, g^+, g^-)$ for glue, that has the natural width g° with the ability to stretch by g^+ and to shrink by g^-;
4. boldface Greek is used for pairs of numbers and triples of such pairs.

A typewriter font is used for the input: String variables are written in uppercase and single command or character input get letters in lowercase.

When glue is added the values of the corresponding components are added; multiplication and division by an integer is also done by multiplying or dividing each component by that integer. These operations are represented in TEX by the commands `\advance`, `\multiply`, and `\divide` for skips.

Some important elements receive fixed names:

e is the value of `\emergencystretch`;
f_ν denotes the `\fontdimen` ν of the current font;
h stands for the `\hsize`;
k represents a kern;
m is the current value of `\mathsurround` at the end of a formula, i.e., at the math-shift character that ends the math mode;
o is the value by which an overfull line is too wide;
ϵ specifies the *environmental condition* that explains how spaces get their width (see below);
τ represents the value of `\tolerance`;
π stands for a penalty;
$\mathbf{l} = (l^\circ, l^+, l^-)$ is the `\leftskip`;
$\mathbf{r} = (r^\circ, r^+, r^-)$ is the `\rightskip`;
$\mathbf{s} = (s^\circ, s^+, s^-)$ represents the `\spaceskip`;
$\mathbf{x} = (x^\circ, x^+, x^-)$ stands for the `\xspaceskip`;
$\mathbf{z} = (0\,\mathrm{pt}, 0\,\mathrm{pt}, 0\,\mathrm{pt})$ is the zero glue; in the `plain` format it is called `\z@skip`;
$\mathbf{Z} = (0, 0)$ is the pair of two zeros.

Important functions are:

$W(\mathtt{T})$ is the width of the input `T` with an unspecified stretch or shrink amount; otherwise the subscript "nw" or "mt" are used for the natural width and the maximal tight width, resp.;
$L(\mathtt{S\&T})$ stands for the change of width that the concatenation of `S` and `T` differs from the sum of the individual widths because of ligatures or kerning, i.e., $W_{\mathrm{nw}}(\mathtt{ST}) = W_{\mathrm{nw}}(\mathtt{S}) + W_{\mathrm{nw}}(\mathtt{T}) + L(\mathtt{S\&T})$;
$\Phi(\mathtt{T})$ returns the value of the space factor that is active at the end of input `T`;
$\mathbf{\Omega}(\mathtt{T})$ stands for a triple of pairs — each pair gives the amounts of stretchability and shrinkability of one of the three infinite glue orders in input `T`;
$\mathbf{G}_\epsilon(\sigma, \mathtt{W})$ is the finite glue that stems from the input `W` for white space, given that the space factor is σ at the start of the input;
$\mathbf{S}_\epsilon(\mathtt{T})$ represents all finite glue contained in `T`.

The first three functions should be clear enough from the given description. They are easily computed by TEX. The last two functions compute glue. Together with the fourth function they are explained and precisely defined in the next subsections together with a few more specialized functions.

The rest of this section describes and defines these functions; skip to section 3 if you are not interested in the details.

Glue in paragraphs. In general an input `T` can be seen as a sequence of text parts $\mathtt{T}_\kappa$, which do not have any author-entered glue except if it is already set, for example, in an `\hbox`, and input $\mathtt{W}_\kappa$ that represents the glue w_κ° plus w_κ^+ minus w_κ^-:

$$\mathtt{T} = \mathtt{T}_0\mathtt{W}_0\mathtt{T}_1\mathtt{W}_1 \ldots \mathtt{T}_{\omega-1}\mathtt{W}_{\omega-1}\mathtt{T}_\omega. \qquad (*)$$

The text elements $\mathtt{T}_\kappa$ contain more than characters that are typeset, for example, TEX control sequences, implicit kerns, mathematics, or boxes. An assumption is made: The input can be processed unconditionally, that means control sequences like `\if`, `\unkern`, etc. are resolved and token lists are expanded. This is not a real restriction but avoids subcases and is reasonable for user input. Each entered white space sequence is related to a glue item $\mathtt{W}_\kappa = \mathtt{w}_{\kappa,1}\mathtt{w}_{\kappa,2} \ldots \mathtt{w}_{\kappa,\mu_\kappa}$, whose elements might be not only spaces but also penalties and dimens from a kern or a math-on/math-off switch. For example, the author might have entered an italic correction $\mathtt{w}_{\kappa,1}$ and then a space $\mathtt{w}_{\kappa,2}$: The natural width of the glue item w_κ° is the sum of the kern and the natural width of a space. Penalties are also allowed in $\mathtt{W}_\kappa$, though they are not white space. Then a kern followed by a tie is covered by the description too.

If leaders are used then the skip part is listed as glue input, since the box part just fills the created white space with some pattern [3, p. 223].

The glue function. A function is needed to find a unified way to write about finite glue in a text. TEX has several mechanisms to deal with it [3, pp. 75–77] which result in many different cases in equations about glue. Therefore, the *environmental condition*, called ϵ, is introduced. It is a number defined as

$$\epsilon := \begin{cases} 0, & \mathbf{s} = \mathbf{z} = \mathbf{x}; \\ 1, & \mathbf{s} = \mathbf{z} \neq \mathbf{x}; \\ 2, & \mathbf{s} \neq \mathbf{z} = \mathbf{x}; \\ 3, & \mathbf{s} \neq \mathbf{z} \neq \mathbf{x}. \end{cases}$$

So $\epsilon = 0$ means that the normal interword spaces are used, odd values stand for a nonzero `\xspaceskip`,

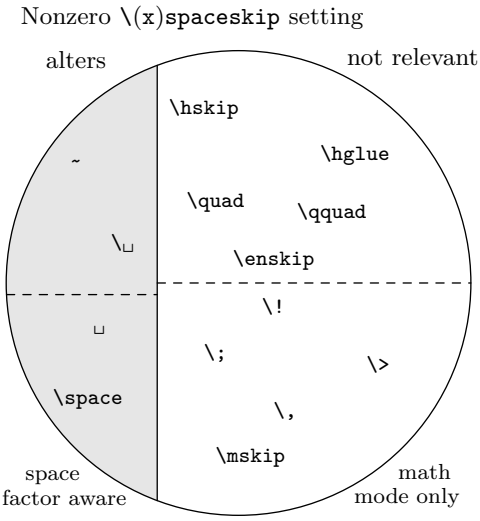


Figure 1: Input methods in `plain` TEX for glue influenced by nonzero `\(x)spaceskip` (shaded area) and glue that is space factor-aware (lower left area)

and values 2 and 3 signal that the `\spaceskip` is nonzero.

TEX knows several kinds of white space. Some are aware of the environment, while others ignore it; let's call the former kind *eaglue*. For example, the normal spaces, the control word `\space`, the control space and the tie belong to eaglue but author-entered white space through `\hskip` is not eaglue. Further, normal spaces and control spaces are different as the first is affected by the space factor before the white space [3, p. 76]. To distinguish between these two kinds of eaglue the first one is called *sfglue* in the following discussion.

Figure 1 gives an overview of the different kinds of finite glue, including muglue which is used only in math mode. The TEX control sequences for infinite glue, such as `\hfil`, are not listed, since such glue is collected by the function $\mathbf{\Omega}$ described below. Technically, not only glue counts, although all values are treated as glue in this discussion. A user can enter an explicit kern with the italic correction, or TEX inserts the value of `\mathsurround` when it reads a math-shift character.

It is possible to have several glue items in a sequence; sometimes this second glue is ignored and sometimes it counts: Two `\hskips` create glue in which each component is the sum of the corresponding values of the skips, but two normal spaces count usually only as a single space. When white space is placed in a sequence the space factor must be considered for all items. For example, in `:\hglue0pt plus1pt \space\space` the last `\space` is still influenced by the space factor that occurs after the colon.

So it requires several cases to define a function that returns the glue value for any glue item `w` when the space factor σ is applied under the environmental condition ϵ:

$$\mathbf{G}_\epsilon(\sigma,\mathtt{w}) := \begin{cases} (f_2, \sigma f_3/1000, 1000 f_4/\sigma), \\ \quad \mathtt{w} \text{ is sfglue, } \epsilon < 2,\ \sigma < 2000; \\ (s^\circ, \sigma s^+/1000, 1000 s^-/\sigma), \\ \quad \mathtt{w} \text{ is sfglue, } \epsilon = 2; \\ (s^\circ, \sigma s^+/1000, 1000 s^-/\sigma), \\ \quad \mathtt{w} \text{ is sfglue, } \epsilon = 3,\ \sigma < 2000; \\ (f_2 + f_7, \sigma f_3/1000, 1000 f_4/\sigma), \\ \quad \mathtt{w} \text{ is sfglue, } \epsilon = 0,\ \sigma \geq 2000; \\ (x^\circ, \sigma x^+/1000, 1000 x^-/\sigma), \\ \quad \mathtt{w} \text{ is sfglue, } \epsilon \text{ is odd, } \sigma \geq 2000; \\ (f_2, f_3, f_4), \\ \quad \mathtt{w} \text{ is eaglue, not sfglue, } \epsilon < 2; \\ (s^\circ, s^+, s^-), \\ \quad \mathtt{w} \text{ is eaglue, not sfglue, } \epsilon \geq 2; \\ (w^\circ, v^+, v^-), \\ \quad \mathtt{w} \text{ is explicitly entered glue,} \\ \quad \texttt{\textbackslash hskip} \text{ or the like;} \\ \quad v^+ := w^+ \text{ and } v^- := w^- \text{ if they} \\ \quad \text{are finite, otherwise } 0\,\text{pt}; \\ (k, 0\,\text{pt}, 0\,\text{pt}), \\ \quad \mathtt{w} \text{ is a kern or mkern with width } k; \\ (m, 0\,\text{pt}, 0\,\text{pt}), \\ \quad \mathtt{w} \text{ is a math-shift character; } m \\ \quad \text{is the value of } \texttt{\textbackslash mathsurround}; \\ (0\,\text{pt}, 0\,\text{pt}, 0\,\text{pt}), \\ \quad \mathtt{w} \text{ is ignored or not white space.} \end{cases}$$

As usual the three components of $\mathbf{G}_\epsilon(\sigma,\mathtt{w})$ have the names $G^\circ_\epsilon(\sigma,\mathtt{w})$, $G^+_\epsilon(\sigma,\mathtt{w})$, and $G^-_\epsilon(\sigma,\mathtt{w})$.

Before the sum of the individual parts of $\mathtt{W}_\kappa = \mathtt{w}_{\kappa,1}\mathtt{w}_{\kappa,2}\ldots\mathtt{w}_{\kappa,\mu_\kappa}$ can be built, one technicality needs to be addressed: The value of `\spacefactor` can be changed. When $\mathtt{w}_{\kappa,0}$ is the empty string, the definition is

$$\mathbf{G}_\epsilon(\Phi(\mathtt{T}_\kappa), \mathtt{W}_\kappa) := \sum_{\iota=1}^{\mu_\kappa} \mathbf{G}_\epsilon(\Phi(\mathtt{T}_\kappa\mathtt{w}_{\kappa,1}\ldots\mathtt{w}_{\kappa,\iota-1}), \mathtt{w}_{\kappa,\iota}).$$

To state the amount of stretchability or shrinkability of the input `T` of the model of eq. (∗) in one function, the following summations are necessary:

$$\sum_{\kappa=0}^{\omega-1} G^+_\epsilon(\Phi(\mathtt{T}_\kappa), \mathtt{W}_\kappa) \text{ and } \sum_{\kappa=0}^{\omega-1} G^-_\epsilon(\Phi(\mathtt{T}_\kappa), \mathtt{W}_\kappa).$$

This looks very complicated, but is a simple function if the default settings of `plain` TEX are considered. In this case only a counting of spaces and punctuation marks in a single line `L` is necessary if the author hasn't entered `\hskip`s. Let

$\nu_{\mathtt{c}}(\mathtt{L})$:= number of occurrences of character `c` in `L`

and define an extension for more than one character

$$\nu_{\mathtt{c}_1\ldots\mathtt{c}_\omega}(\mathtt{L}) := \sum_{\kappa=1}^{\omega} \nu_{\mathtt{c}_\kappa}(\mathtt{L}).$$

The counts of the "relevant" punctuation marks, i.e., those punctuation marks not preceded by an uppercase letter, not at end of line, and not followed by a control space or tie, and the `plain` TEX settings give total stretchability and shrinkability:

$$\big(\nu_{\sqcup}(\mathtt{L}) + \frac{5}{4}\nu_{,}(\mathtt{L}) + \frac{3}{2}\nu_{;}(\mathtt{L}) + 2\nu_{:}(\mathtt{L}) + 3\nu_{.?!}(\mathtt{L})\big) f_3,$$

$$\big(\nu_{\sqcup}(\mathtt{L}) + \frac{4}{5}\nu_{,}(\mathtt{L}) + \frac{2}{3}\nu_{;}(\mathtt{L}) + \frac{1}{2}\nu_{:}(\mathtt{L}) + \frac{1}{3}\nu_{.?!}(\mathtt{L})\big) f_4,$$

where $\nu_{\sqcup}(\mathtt{L})$ numbers the spaces that are not preceded by a relevant punctuation mark in `L`. Note the terms f_3 and f_4, which are `\fontdimen`s 3 and 4, are replaced by s^+ and s^- if $\epsilon = 2$.

Infinite glue. Inspired by [4, §822], infinite glue is written as a triple of pairs of numbers. Each pair represent one of the three orders of infinite glue: fil, fill, and filll. So the amount of infinite glue in `g` is $\boldsymbol{\Omega}(\mathtt{g}) = (\boldsymbol{\Omega}_1(\mathtt{g}), \boldsymbol{\Omega}_2(\mathtt{g}), \boldsymbol{\Omega}_3(\mathtt{g}))$ and each pair has two numbers: $\boldsymbol{\Omega}_\iota(\mathtt{g}) = (\Omega_\iota^+(\mathtt{g}), \Omega_\iota^-(\mathtt{g}))$.

For example, the glue specification `g = \hskip` χ pt `plus` 2α `fil minus` $-\beta$ `fill` has two different orders of infinite glue, therefore $\boldsymbol{\Omega}_1(\mathtt{g}) = (2\alpha, 0)$, $\boldsymbol{\Omega}_2(\mathtt{g}) = (0, -\beta)$, and $\boldsymbol{\Omega}_3(\mathtt{g}) = \mathbf{Z}$.

To define for `T` of eq. (∗)

$$\boldsymbol{\Omega}(\mathtt{T}) = (\boldsymbol{\Omega}_1(\mathtt{T}), \boldsymbol{\Omega}_2(\mathtt{T}), \boldsymbol{\Omega}_3(\mathtt{T}))$$

a definition of $\boldsymbol{\Omega}_\iota(\mathtt{T})$ for $1 \le \iota \le 3$ is needed. With

$$\Omega_\iota^+(\mathtt{w}) := \begin{cases} \alpha, & \text{if } \mathbf{w} \text{ contains the amount } \alpha \\ & \text{of infinite stretchability of order } \iota \\ 0, & \text{otherwise} \end{cases}$$

$$\Omega_\iota^-(\mathtt{w}) := \begin{cases} \alpha, & \text{if } \mathbf{w} \text{ contains the amount } \alpha \\ & \text{of infinite shrinkability of order } \iota \\ 0, & \text{otherwise} \end{cases}$$

$\boldsymbol{\Omega}_\iota(\mathtt{W}_\kappa)$ when $\mathtt{W}_\kappa = \mathtt{w}_{\kappa,1}\mathtt{w}_{\kappa,2}\ldots\mathtt{w}_{\kappa,\mu_\kappa}$ is defined as

$$\begin{aligned}\boldsymbol{\Omega}_\iota(\mathtt{W}_\kappa) &= (\Omega_\iota^+(\mathtt{W}_\kappa), \Omega_\iota^-(\mathtt{W}_\kappa)) \\ &:= \left(\sum_{\chi=1}^{\mu_\kappa} \Omega_\iota^+(\mathtt{w}_{\kappa,\chi}), \sum_{\chi=1}^{\mu_\kappa} \Omega_\iota^-(\mathtt{w}_{\kappa,\chi})\right)\end{aligned}$$

so that

$$\begin{aligned}\boldsymbol{\Omega}_\iota(\mathtt{T}) &= (\Omega_\iota^+(\mathtt{T}), \Omega_\iota^-(\mathtt{T})) \\ &:= \left(\sum_{\kappa=0}^{\omega-1} \Omega_\iota^+(\mathtt{W}_\kappa), \sum_{\kappa=0}^{\omega-1} \Omega_\iota^-(\mathtt{W}_\kappa)\right).\end{aligned}$$

Glue in math mode. TEX is often used to typeset mathematics, so the glue that is present in math mode should be analyzed too. Only inline formulas are treated here. (Overfull lines in display math mode are a different topic, not handled in this article; see [3], pp. 195–197.)

When TEX operates in math mode the space factor and normal spaces have no meaning: TEX inserts the spacing according to its own rules.

In math mode TEX typesets the formulas in *styles*. Eight styles are defined: the four basic styles are scriptscript style *SS*, script style *S*, text style *T*, and display style *D*, each with two versions [3, pp. 140–141]. They are usually sorted as

$$SS' < SS < S' < S < T' < T < D' < D$$

so that it makes sense to represent them by the numbers 0 to 7. Inside the formulas TEX operates with *atoms*. There are thirteen types, but only eight are important for spacing as the others are transformed into these eight (see [3], p. 158 and Appendix G). Again the types are sorted

$$\begin{aligned}&\text{Ord} < \text{Op} < \text{Bin} < \text{Rel} \\ &\quad < \text{Open} < \text{Close} < \text{Punct} < \text{Inner} \\ &\qquad < \text{Over} < \text{Under} < \text{Acc} < \text{Rad} < \text{Vcent}\end{aligned}$$

so that the numbers 0 to 12 can be assigned to the atoms. The names stand for atoms of types ordinary, large operator, binary operator, relation, opening, closing, punctuation, inner, overline, underline, accented, radical, and vcenter; examples of atoms for the first eight types are given in Fig. 2, first column. Note, however, it is not the symbol that counts but its usage: In `$+1$` the '+' is not a binary operator and therefore not of type Bin.

In math mode TEX doesn't use single fonts but font triples combined in a *family*; there are up to 16 families. A triple of fonts consists of a font for normal text symbols, one for subscripts, and one for sub-subscripts; the first font is called the `\textfont`. In `plain` TEX, family 1 contains the math italic letters, family 2 the math symbols, and family 3 large symbols. The fonts in families 2 and 3 need special `\fontdimen` parameters but only one of them is of interest for our glue concerns: `\fontdimen6` of family 2's `\textfont` (named f_6^{t2}), the quad width of the font, plays an important role. It is used to convert the *muglue*, which is measured in mu, into glue measured in pt: $1\,\text{mu} = f_6^{t2}/18$.

As mentioned above, TEX inserts glue in math mode and ignores normal white space but, of course, an author can explicitly enter `\hskip` or `\kern` or `\mskip` or `\mkern` commands; the latter two use muglue. In the model (∗) each explicitly entered glue is represented by a $\mathtt{W}_\kappa$ but inserted glue in math mode does not create such a glue token. The input $\mathtt{T}_\kappa$ may contain material in math mode which stretches or shrinks because of glue inserted by TEX.

			Insert the specified glue after Atom if it is followed by an atom of type							
Ex.	Atom type		0	1	2	3	4	5	6	7
a	Ord	0		$\mathbf{n}^1$	$\mathbf{m}$	$\mathbf{t}$				$\mathbf{n}$
$\sum$	Op	1	$\mathbf{n}^1$	$\mathbf{n}^1$		$\mathbf{t}$				$\mathbf{n}$
$+$	Bin	2	$\mathbf{m}$	$\mathbf{m}$			$\mathbf{m}$			$\mathbf{m}$
$=$	Rel	3	$\mathbf{t}$	$\mathbf{t}$			$\mathbf{t}$			$\mathbf{t}$
$($	Open	4								
$)$	Close	5		$\mathbf{n}^1$	$\mathbf{m}$	$\mathbf{t}$				$\mathbf{n}$
$:$	Punct	6	$\mathbf{n}$	$\mathbf{n}$		$\mathbf{n}$	$\mathbf{n}$	$\mathbf{n}$	$\mathbf{n}$	$\mathbf{n}$
$\frac{1}{2}$	Inner	7	$\mathbf{n}$	$\mathbf{n}^1$	$\mathbf{m}$	$\mathbf{t}$	$\mathbf{n}$		$\mathbf{n}$	$\mathbf{n}$

[1] applied in styles 0–7; otherwise 4–7 only

Figure 2: TEX's *space table* [3, p. 170], with examples for the atoms

There are three amounts of glue that TEX inserts into formulas:

- **n** is the glue created by the command `\mskip\thinmuskip` in pt, i.e., it is muglue expressed in pt;
- **m** is like **n** but `\medmuskip` is used;
- **t** is like **n** but `\thickmuskip` is used.

What type gets inserted is determined by the *space table* shown in Fig. 2. Most glue is inserted only in styles T', T, D', and D.

The glue of these three types is inserted around selected atoms only if certain conditions are met. So first a counting of atom pairs is defined:

$$\mu(\chi, \alpha_1\alpha_2, \mathtt{T}) := \text{number of times that the atom sequence } \alpha_1\alpha_2 \text{ occurs in style } \chi \text{ inside text } \mathtt{T}.$$

The *bracket notation* (a.k.a. *Iverson's convention*)

$$[\,\text{statement}\,] := \begin{cases} 1 & \text{statement is true} \\ 0 & \text{statement is false} \end{cases}$$

helps to describe the glue inside math mode in $(*)$ with three functions.

$$\begin{aligned}\mathbf{N}(\mathtt{T}) := \sum_{\kappa=0}^{\omega}\Bigg(&\sum_{\chi=0}^{7}\big(\mu(\chi,\alpha1,\mathtt{T}_\kappa)\mathbf{n}\,[\alpha\in\{0,1,5,7\}] \\ &\qquad+\mu(\chi,10,\mathtt{T}_\kappa)\big) \\ &+\sum_{\chi=4}^{7}\big(\mu(\chi,\alpha7,\mathtt{T}_\kappa)\mathbf{n}\,[\alpha\in\{0,1,5\}] \\ &\qquad+\mu(\chi,6\alpha,\mathtt{T}_\kappa)\mathbf{n}\,[\alpha\in\{0,1,3,4,5,6,7\}] \\ &\qquad+\mu(\chi,7\alpha,\mathtt{T}_\kappa)\mathbf{n}\,[\alpha\in\{0,4,6,7\}]\big)\Bigg).\end{aligned}$$

The formula for **m** looks only at styles 4–7:

$$\begin{aligned}\mathbf{M}(\mathtt{T}) := \sum_{\kappa=0}^{\omega}\sum_{\chi=4}^{7}\big(&\mu(\chi,2\alpha,\mathtt{T}_\kappa)\mathbf{m}\,[\alpha\in\{0,1,4,7\}] \\ &+\mu(\chi,\alpha2,\mathtt{T}_\kappa)\mathbf{m}\,[\alpha\in\{0,5,7\}]\big).\end{aligned}$$

And for **t**, $\mathbf{T}(\mathtt{T})$ is computed in a similar way:

$$\begin{aligned}\mathbf{T}(\mathtt{T}) := \sum_{\kappa=0}^{\omega}\sum_{\chi=4}^{7}\big(&\mu(\chi,3\alpha,\mathtt{T}_\kappa)\mathbf{t}\,[\alpha\in\{0,1,5,7\}] \\ &+\mu(\chi,\alpha3,\mathtt{T}_\kappa)\mathbf{t}\,[\alpha\in\{0,1,4,7\}]\big).\end{aligned}$$

At least with the defaults of `plain` TEX, the split into three functions makes some sense as only $\mathbf{M}(\mathtt{T})$ can stretch and shrink; $\mathbf{T}(\mathtt{T})$ cannot shrink and $\mathbf{N}(\mathtt{T})$ can neither stretch nor shrink. For the applications of this paper the shrinkability and stretchability is of interest, so the function $\mathbf{N}(\mathtt{T})$ is never used as long as `\thinmuskip` is not changed.

Again this looks more complicated than it is in order to cover all theoretical aspects: $\mathbf{M}(\mathtt{T})$ counts the number of Bin atoms in text and display style and this number is multiplied by $2\mathbf{m}$. Similarly, $\mathbf{T}(\mathtt{T})$ counts the number of Rel atoms that don't follow a Punct atom; this count is multiplied by $2\mathbf{t}$. The identification of the relevant atoms is easily described for the styles T' and T that appear in paragraphs if the style isn't explicitly changed with `\displaystyle`: A Bin or Rel atom is typeset in text style if it is neither raised nor lowered with respect to the baseline.

To capture inserted glue at a line break the glue function is extended to math mode. In styles 4–7, TEX inserts a penalty after Bin and Rel atoms to create an opportunity to break formulas.

$$\mathbf{G}_\epsilon(\sigma,\mathtt{m}) := \begin{cases} \mathbf{m}, & \mathtt{m} \text{ is a Bin atom in styles 4–7} \\ \mathbf{t}, & \mathtt{m} \text{ is a Rel atom in styles 4–7.} \end{cases}$$

Summary: Glue in the input. In the model $(*)$ the infinite glue is captured by $\boldsymbol{\Omega}(\mathtt{T})$.

The finite glue of explicitly entered glue `W` that follows the input `T` is $\mathbf{G}_\epsilon(\Phi(\mathtt{T}),\mathtt{W})$. Three functions are defined to represent the finite glue that TEX inserts into input `T` which contains material in math mode: $\mathbf{T}(\mathtt{T})$, $\mathbf{M}(\mathtt{T})$, and $\mathbf{N}(\mathtt{T})$. Therefore:

$$\begin{aligned}\mathbf{S}_\epsilon(\mathtt{T}) &:= \text{finite glue in } \mathtt{T} \\ &= \sum_{\kappa=0}^{\omega-1}\mathbf{G}_\epsilon(\Phi(\mathtt{T}_\kappa),\mathtt{W}_\kappa) + \mathbf{T}(\mathtt{T}) + \mathbf{M}(\mathtt{T}) + \mathbf{N}(\mathtt{T})\end{aligned}$$

when $\mathtt{T} = \mathtt{T}_0\mathtt{W}_0\mathtt{T}_1\mathtt{W}_1\ldots\mathtt{T}_{\omega-1}\mathtt{W}_{\omega-1}\mathtt{T}_\omega$ as defined in $(*)$. Of course, $\mathbf{S}_\epsilon(\mathtt{T}) = (S^\circ_\epsilon(\mathtt{T}), S^+_\epsilon(\mathtt{T}), S^-_\epsilon(\mathtt{T}))$.

3 An experiment

I use a well-known text [3, p. 24] for the first experiment. The text is typeset with a smaller `\hsize` than the column width in order to produce overfull boxes. Then the value of the dimension parameter `\emergencystretch` is increased up to the point where an overfull box disappears.

For example, when `\hsize` = 100 pt, five overfull boxes are produced. Then the value of the dimen

`\emergencystretch` is increased in steps of 0.1 pt, and at `\emergencystretch` = 1.5 pt TEX is able to typeset the text with only two overfull lines.

An experiment starts with a boldface line showing the number of the experiment for reference and it ends with this end-of-experiment marker: □.

Experiment 1: TEX definitions

```
\hsize=100pt \overfullrule=1pt
```

TEX input

```
Once upon a time, in a distant galaxy called
\"O\"o\c c, there lived a computer named
R.~J. Drofnats.

Mr.~Drofnats---or ``R. J.,'' as he preferred to
be called---was happiest when he was at work
typesetting beautiful documents.
```

TEX output

i) `\emergencystretch`: left col. 0.1 pt, right 0.2 pt

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

ii) `\emergencystretch`: left 1.4 pt, right 1.5 pt

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

iii) `\emergencystretch`: left 9.2 pt, right 9.3 pt

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

iv) `\emergencystretch`: left 11.1 pt, right 11.2 pt

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats.

Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents. □

During the experiment seven overfull lines occur. Five exist at the beginning, and two are created during the experiment. We'll name the five cases a) to e); the overfull line that is created in the first paragraph is referenced by b′), the other by d′).

The result of the experiment is captured in the following summary, stating in the first line the number of the experiment and its parameters and in the second line the measured values for the dimension `\emergencystretch`. Values separated by a slash mean: The resolution of one overfull line created another, which was resolved with the second value.

Experiment 1 continued: Results

a) parameters; b) used `\emergencystretch`

1a): `\hsize`=100 pt

b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt □

A few dimensions. TEX reports by how much the overfull lines are too wide.

$W(\texttt{called}) = 25.00005\,\text{pt}$ juts $5.88911\,\text{pt}$ (1a)
$W(\texttt{Drof-}) = 22.94449\,\text{pt}$ juts $2.13905\,\text{pt}$ (1b)
$W(\texttt{``R.}) = 15.13892\,\text{pt}$ juts $12.36128\,\text{pt}$ (1c)
$W(\texttt{happiest}) = 36.7223\,\text{pt}$ juts $2.33350\,\text{pt}$ (1d)
$W(\texttt{doc-}) = 18.61115\,\text{pt}$ juts $10.61136\,\text{pt}$ (1e)

where $W(\texttt{text})$ stands for the width of `text` as defined in the previous section. For a single word or word fragment this is always the natural width, i.e., there is no stretching or shrinking. When we explicitly mention that the natural width of a text with white space is computed then we use the subscript "nw" with W.

As mentioned above, later two more lines become overfull:

$W_{\text{nw}}(\texttt{R. J.}) = 21.38892\,\text{pt}$ juts $1.97240\,\text{pt}$ (1b′)
$W(\texttt{work}) = 21.13893\,\text{pt}$ juts $2.08344\,\text{pt}$ (1d′)

The overfull lines are resolved by various means: in a) and c) only stretchability is needed, in d) a word gets hyphenated, d′) is similar to a) and c) but some fixed width material is added at the left side of the line, variable sized material is added to b), in b′) variable sized material is moved to the next line,

and e) is similar to that but a word is hyphenated instead of breaking at glue.

In order to have all dimensions at hand for later calculations, here are the values of the material moved to the next line in cases with hyphenation and the material added to some lines at the left:

$$W_{\text{nw}}(\texttt{a com}) = 26.11116\,\text{pt} \quad \text{is added} \qquad (2b)$$
$$W(\texttt{est}) = 12.27779\,\text{pt} \quad \text{is moved} \qquad (2d)$$
$$W(\texttt{pi}) = 8.33336\,\text{pt} \quad \text{is added} \qquad (2d')$$
$$W_{\text{nw}}(\texttt{tiful doc-}) = 40.00009\,\text{pt} \quad \text{is moved} \qquad (2e)$$

The badness values. With the help of the parameter `\tracingparagraphs`, the badness values of the line-breaking algorithm can be found [14]. The lines of the paragraph without using `\emergencystretch` and the paragraphs at the right of runs i) to iv) have the following badness values:

$\overline{7}$, *, $\overline{15}$, *, 0 and *, 17, *, 171, *, 0 (3)

$\overline{7}$, *, $\overline{15}$, *, 0 and *, 15, 190, *, *, 0 (3i)

$\overline{7}$, *, $\overline{15}$, *, 0 and 184, $\overline{17}$, 0, 113, $\overline{2}$, 0 (3ii)

$\overline{7}$, 198, 0, *, 0 and 4, $\overline{17}$, 0, 15, $\overline{2}$, 0 (3iii)

$\overline{7}$, 136, 0, 200, 0 and 3, $\overline{17}$, 0, 11, $\overline{2}$, 0 (3iv)

The symbol $*$ stands for the infinite badness of the overfull lines. Such a line has the fitness class *tight* [3, p. 97]. Finite badness values of lines containing glue that shrinks are written with a bar above them. In this way all fitness classes can be identified.

In this list the badness values for lines in the fitness classes very loose and loose change to lower values when the `\emergencystretch` increases. The values of some decent lines change too and then it is known that the white space in such a line has to stretch. Only tight lines and decent lines, in which the glue shrinks, keep a constant badness value. As indicated by the bar the 7 and the 2 in (3iv) belong to decent lines in which the glue shrinks.

But the "real" badness values obtained with `\hbadness` [14, p. 367] in the last run are

$\overline{7}$, 4660, 7, 10000, 0 and 1264, $\overline{17}$, 0, 206, $\overline{2}$, 0. (3iv′)

4 Some theory

Do we need to perform this stepwise increment in order to obtain the values for `\emergencystretch`, or is there a way to compute the values?

First, let's think about infinite glue. TEX throws an error if it finds infinite shrinkability in a paragraph. Infinite stretchability in a line makes it nearly impossible to generate an overfull line as TEX can break at any glue, penalty or hyphenation possibility. Therefore, in this analysis all glue in the text is assumed to be finite.

As was noted above TEX can get rid of an overfull line in many different ways. It is too complex to handle all the cases at once. It is better to start with a simple model. Therefore, let's make the following assumption: The overfull line is changed only by moving material to the next line. In other words, no new material is added to the previously overfull line, so that only the cases a), b′), c), and d) of experiment 1 are considered in this section.

A formula for `\emergencystretch`. The badness is a heuristic based on the amount by which the available spaces have to shrink or to stretch in order to make the line fill a predefined length. Sometimes the formula for the badness is stated as

$$100 \times \left(\frac{\text{used} \begin{matrix}\text{stretch}\\ \text{shrink}\end{matrix} \text{ability in the line}}{\text{available} \begin{matrix}\text{stretch}\\ \text{shrink}\end{matrix} \text{ability in the line}} \right)^3 \qquad (4)$$

but this is only an *approximation* [3, p. 97]. (Of course, the same word in the numerator and denominator must be selected.) It's not only that the badness is always an integer ≤ 10000, but the computation given in §108 of [4] computes it without the "need to squeeze out the last drop of accuracy." The section further states that the routine is "capable of computing at most 1095 distinct values." For the purpose of the heuristic this computation is sufficient. Nevertheless it should be remembered that the badness value is not computed by a continuous function as the formula might suggest.

As the case of an overfull line is analyzed, formula (4) is used for the stretchability of a line. Let β be the badness, u the dimension of used stretchability, a the available stretchability in the line, and e the value of `\emergencystretch`. Then for e large enough to remove the overfull line

$$\beta \approx 100 \left(\frac{u}{a+e} \right)^3 .$$

Of course, $a + e \neq 0\,\text{pt}$. But to see an impact of `\emergencystretch` a line must have some stretchability, i.e., $a > 0\,\text{pt}$.

The badness of a line is difficult to estimate. As mentioned, it can be seen in the data written to the log file if `\tracingparagraphs` is set to 1.

The badness β of a non-overfull line is a fraction of the `\tolerance` $\tau > 0$; so for φ with $0 \le \varphi \le 1$ the equation

$$\beta = \varphi\tau$$

holds. Replacing the left hand side of the approximation with the right hand side of the equation gives

$$\varphi\tau \approx 100 \left(\frac{u}{a+e} \right)^3 \quad \text{or} \quad \sqrt[3]{\frac{\varphi\tau}{100}} \approx \frac{u}{a+e}$$

The optimal value for `\emergencystretch`

and if $\varphi \neq 0$

$$e \approx \sqrt[3]{\frac{100}{\varphi\tau}}\, u - a. \tag{5}$$

Therefore three values a, u, and φ must be found to get an approximation of e to resolve a given overfull line. In such a line $\varphi > 0$, of course.

As stated here, the values are not obvious to calculate or even to guess but they can be factored to a level which allows estimation of their values. The values a and u can be computed from trace output shown by TEX but not when an overfull line is reported — this is discussed later.

The plan for the analysis. It's probably best to look at the situation when the overfull line gets the right amount for `\emergencystretch` so that a break creating an acceptable line occurs. Therefore the first question is: Where can TEX break a line? On p. 96 of [3], five cases are listed. A line break might be at

1. glue if a non-discardable item (not glue, kern, penalty or a math switch) appears before the glue;
2. a kern if it is followed by glue;
3. a math-off if it is followed by glue;
4. a penalty;
5. a hyphen, either inserted by TEX or an explicit hyphen present in the text.

The first three items are combined in this analysis into one case (named "break at glue") using the definitions made in section 2. The glue might be entered by the author or it is inserted by TEX after a binary operation or a relation in math mode. Number 4 can also be added except when the penalty is not followed by glue. So the second case is a break after a penalty that is not followed by glue. The third case represents number 5, the discretionary break.

Case 1: Break at glue. A before and after comparison of the line contents shows the situation when the overfull line gets the right amount for the dimension `\emergencystretch` so that a break at the last breakable white space can occur.

In the following analysis, we assume the input `L` doesn't start and the input `M` doesn't end with glue that can be dropped at the beginning or end of a line. Such glue is ignored by TEX and it makes the description much easier if such glue isn't present in the first place.

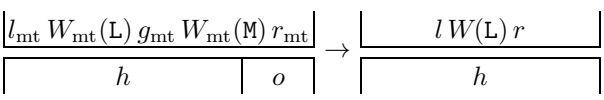


Figure 3: Resolve overfull line with break at glue

At the left side of Fig. 3 some input `L` of width $W_{\mathrm{mt}}(\mathtt{L})$ is typeset, with `\leftskip` at the left, then it is followed by glue **g** of width g_{mt} at a place with a certain `\spacefactor` $\Phi(\mathtt{L})$ and some material `M` that contains no breakable glue and that is followed by `\rightskip`. The width of the material is named $W_{\mathrm{mt}}(\mathtt{M})$. The white space in the texts, the glue and the skips on both sides are shrunk to their maximum as the line is overfull and therefore it is *maximally tight*; this is indicated by the subscript "mt" to W and other variables. The length of the line at the left is the sum of the `\hsize` h (it may be the width of a `\parshape` or a `\hangindent`) and the amount that the line is too wide, let's call it o.

The right hand side contains also some information. The natural width of the three items in the line, $l^\circ + W_{\mathrm{nw}}(\mathtt{L}) + r^\circ$, must be smaller than h because of the following observation: TEX would not create an overfull line if the stretchability in the line would allow a break at the glue **g**; at the left the amount of stretchability was too small to resolve the problem. The width $l^\circ + W_{\mathrm{nw}}(\mathtt{L}) + r^\circ$ can only stretch to the width h with the help of e, so it must be smaller than the width $l + W(\mathtt{L}) + r$.

The available stretchability of the resolved overfull line (the right side of Fig. 3) lies in the stretchability of the two glue items **l** and **r** and in the stretchability of the text `L`. All these elements were named in section 2: The available stretchability on the right hand side of Fig. 3 is the sum of the stretchability of the `\leftskip`, l^+, of the text `L`, $S^+_\epsilon(\mathtt{L})$, and of the `\rightskip`, r^+:

$$a = l^+ + S^+_\epsilon(\mathtt{L}) + r^+. \tag{6}$$

The used stretchability is the next difficult-to-guess value. As explained above in the right hand side of Fig. 3 the text is typeset with some amount of stretch. This is the used stretchability u. So the following equations hold:

$$h = l + W(\mathtt{L}) + r = l^\circ + W_{\mathrm{nw}}(\mathtt{L}) + r^\circ + u.$$

This leads to an equation for u: $u = h - l^\circ + W_{\mathrm{nw}}(\mathtt{L}) + r^\circ$. But it is too soon to stop here. A term like $W_{\mathrm{nw}}(\mathtt{L})$ is so content-dependent that it should be analyzed further. In (6) the value $S^+_\epsilon(\mathtt{L})$ is used. This is more or less the number of spaces in `L`. It is an important characteristic and not dependent on all elements of the input `L`. $W_{\mathrm{nw}}(\mathtt{L})$ is different, as hardly any other text will have the same width as `L`.

Ok, the analysis continues and the equation for the right hand side is reordered

$$l^\circ + W_{\mathrm{nw}}(\mathtt{L}) + r^\circ = h - u.$$

The left hand side gives another equation

$$h + o = l_{\mathrm{mt}} + W_{\mathrm{mt}}(\mathtt{L}) + g_{\mathrm{mt}} + W_{\mathrm{mt}}(\mathtt{M}) + r_{\mathrm{mt}}\,.$$

where $W_{\mathrm{mt}}(\mathtt{L})$ typesets the text L with the minimal amount of white space as explained above. Of course, $l_{\mathrm{mt}} = l^\circ - l^-$ and $r_{\mathrm{mt}} = r^\circ - r^-$. For the second term the equation $W_{\mathrm{mt}}(\mathtt{L}) = W_{\mathrm{nw}}(\mathtt{L}) - S^-_\epsilon(\mathtt{L})$ holds. And $g_{\mathrm{mt}} = G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g})$ with the function $\mathbf{G}_\epsilon$ of section 2.

When all these equations are applied to the equation of the left hand side it becomes:

$$\begin{aligned} h + o &= l^\circ - l^- + W_{\mathrm{nw}}(\mathtt{L}) - S^-_\epsilon(\mathtt{L}) + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) \\ &\quad - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) + W_{\mathrm{mt}}(\mathtt{M}) + r^\circ - r^- \\ &= l^\circ + W_{\mathrm{nw}}(\mathtt{L}) + r^\circ - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &\quad + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) + W_{\mathrm{mt}}(\mathtt{M}). \end{aligned}$$

Now the first three summands are replaced by $h - u$ from the equation of the right hand side:

$$\begin{aligned} h + o &= h - u - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &\quad + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) + W_{\mathrm{mt}}(\mathtt{M}). \end{aligned}$$

The goal is reached if u is moved to the left side and o to the right:

$$\begin{aligned} u &= W_{\mathrm{mt}}(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &\quad + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g}). \end{aligned}$$

All values except $W_{\mathrm{mt}}(\mathtt{M})$ are known and the width of the material M that is moved to the next line is much easier to estimate than the value of u, especially if it is of fixed width. Otherwise the material that is moved contains one or more ties. As $W_{\mathrm{mt}}(\mathtt{M})$ is the width of the material with all white spaces shrunk to maximum the following equation with the natural width $W_{\mathrm{mt}}(\mathtt{M}) = W_{\mathrm{nw}}(\mathtt{M}) - S^-_\epsilon(\mathtt{M})$ can be applied to the result. In total this gives

$$\begin{aligned} u &= W_{\mathrm{nw}}(\mathtt{M}) - S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &\quad + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g}). \end{aligned} \tag{7}$$

Case 2: Break at penalty. As mentioned above the only situation that must be considered here is a break at a penalty that is not followed by glue. Let's call the penalty π; it separates inputs L and M such that no kerning or ligatures must be taken into account.

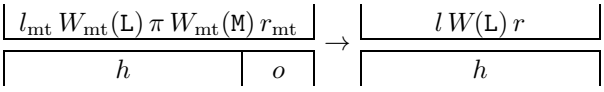


Figure 4: Resolve overfull line with break at penalty

The resulting line at the right in Fig. 4 looks identical to the right hand side in Fig. 3. So (6) holds without a change. The first difference occurs when the formula for the left hand side is written down; now the summand g_{mt} disappears. Or one can say the glue **g** of Fig. 3 must be replaced by **z** and all formulas stay the same. Therefore the equation for u becomes

$$u = W_{\mathrm{nw}}(\mathtt{M}) - S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^-.$$

Using the bracket notation of section 2, an abbreviation for the case when the line break occurs at glue is defined

$$\Gamma := [\text{break occurs at glue}]$$

so that the cases 1 and 2 can be combined:

$$\begin{aligned} u &= W_{\mathrm{nw}}(\mathtt{M}) - S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &\quad + \big(G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g})\big)\Gamma. \end{aligned} \tag{8}$$

Case 3: Break at hyphen. Another form of removing an overfull line is given in Fig. 5. Here the material that sticks out in the overfull line is not moved completely to the next line. It is a discretionary break and only the fragment of width $W(\mathtt{M})$ is moved; material with width $W(\mathtt{K}')$ is kept on the line together with the hyphen character of width $W(\mathtt{-})$. (The format `plain` TEX assigns the value of `\defaulthyphenchar` to `\hyphenchar`. Here $W(\mathtt{-})$ is used instead of $W($`\char\hyphenchar\font`$)$.) As the characters of a font can interact through ligatures and kerns, the sum of the two widths is not necessarily the width of the concatenated strings. The notation $L(\mathtt{S\&T})$ is used to denote the change of width (compared to the sum) either through a ligature or kern when the strings S and T are concatenated, i.e., $W_{\mathrm{nw}}(\mathtt{ST}) = W_{\mathrm{nw}}(\mathtt{S}) + W_{\mathrm{nw}}(\mathtt{T}) + L(\mathtt{S\&T})$ as explained in section 2.

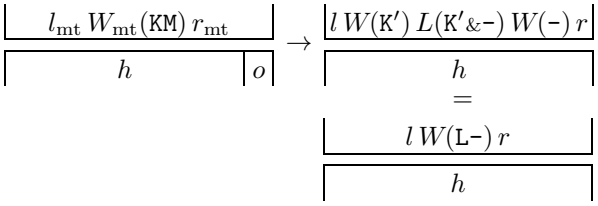


Figure 5: Resolve overfull line with break at hyphen

If the discretionary break occurs at an explicit hyphen then $\mathtt{K} = \mathtt{K}'\mathtt{-}$; otherwise, $\mathtt{K} = \mathtt{K}'$. The distinction can be handled via the abbreviations

$$\Theta := [\text{the line ends with an inserted hyphen}]$$

$$\Xi := [\text{the line ends with an explicit hyphen}]$$

to avoid long statements in bracket notation in the formulas.

Using the first abbreviation the widths of K and K′ fulfill the following equation:

$$\begin{aligned} W_{\mathrm{nw}}(\mathtt{K}) + \big(L(\mathtt{K\&-}) + W(\mathtt{-})\big)\Theta &= \\ W_{\mathrm{nw}}(\mathtt{K}') + L(\mathtt{K}'\&\mathtt{-}) + W(\mathtt{-}). \end{aligned} \tag{9}$$

Note that K′ might end in a hyphen, for example, if the break occurs at an em-dash, and therefore the summand $L(\mathtt{K}'\&\mathtt{-})$ is important.

The available stretchability comes from the `\leftskip`, the `\rightskip`, and the text K′. But

as Fig. 5 shows, the concatenation of this string together with the fixed width hyphen is L- and therefore all the stretchability comes from L, the text that remains in the line: Equation (6) is still valid.

The used stretchability requires a bit more complicated equivalence of the left and right hand side equations that were discussed in case 1. Figure 5 together with an application of (9) gives

$$\begin{aligned} h &= l^\circ + W_{\mathrm{nw}}(\mathtt{L\text{-}}) + r^\circ + u \\ &= l^\circ + W_{\mathrm{nw}}(\mathtt{K}') + L(\mathtt{K}'\&\text{-}) + W(\text{-}) + r^\circ + u \\ &= l^\circ + W_{\mathrm{nw}}(\mathtt{K}) + \big(L(\mathtt{K}\&\text{-}) + W(\text{-})\big)\Theta + r^\circ + u \end{aligned}$$

or

$$l^\circ + W_{\mathrm{nw}}(\mathtt{K}) + r^\circ = h - \big(L(\mathtt{K}\&\text{-}) + W(\text{-})\big)\Theta - u.$$

The left hand side gives

$$\begin{aligned} h + o &= l_{\mathrm{mt}} + W_{\mathrm{mt}}(\mathtt{KM}) + r_{\mathrm{mt}} \\ &= l_{\mathrm{mt}} + W_{\mathrm{mt}}(\mathtt{K}) + L(\mathtt{K}\&\mathtt{M}) + W_{\mathrm{mt}}(\mathtt{M}) + r_{\mathrm{mt}} \\ &= l^\circ + W_{\mathrm{nw}}(\mathtt{K}) + r^\circ - l^- - S^-_\epsilon(\mathtt{K}) - r^- \\ &\qquad + L(\mathtt{K}\&\mathtt{M}) + W_{\mathrm{mt}}(\mathtt{M}). \end{aligned}$$

With the equation for the right hand side and with $S^-_\epsilon(\mathtt{K}) = S^-_\epsilon(\mathtt{L})$ this equals

$$\begin{aligned} h + o = h &- \big(L(\mathtt{K}\&\text{-}) + W(\text{-})\big)\Theta - u \\ &- l^- - S^-_\epsilon(\mathtt{L}) - r^- + L(\mathtt{K}\&\mathtt{M}) + W_{\mathrm{mt}}(\mathtt{M}) \end{aligned}$$

and therefore with $W_{\mathrm{mt}}(\mathtt{M}) = W_{\mathrm{nw}}(\mathtt{M}) - S^-_\epsilon(\mathtt{M})$

$$\begin{aligned} u = W_{\mathrm{nw}}(\mathtt{M}) &- S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &- \big(L(\mathtt{K}\&\text{-}) + W(\text{-})\big)\Theta + L(\mathtt{K}\&\mathtt{M}). \end{aligned}$$

The last step is to remove the reference to K. When the hyphen is inserted $L(\mathtt{K}\&\text{-}) = L(\mathtt{L}\&\text{-})$ and $L(\mathtt{K}\&\mathtt{M}) = L(\mathtt{L}\&\mathtt{M})$. If the hyphen is explicit, then $L(\mathtt{K}\&\mathtt{M}) = L(\mathtt{L}\text{-}\&\mathtt{M})$. Using Ξ the equation for u can be stated without K as

$$\begin{aligned} u = W_{\mathrm{nw}}(\mathtt{M}) &- S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &+ \big(L(\mathtt{L}\&\mathtt{M}) - L(\mathtt{L}\&\text{-}) - W(\text{-})\big)\Theta \\ &+ L(\mathtt{L}\text{-}\&\mathtt{M})\,\Xi. \end{aligned} \tag{10}$$

Combining the cases. The equations (8) and (10) can be combined:

$$\begin{aligned} u = W_{\mathrm{nw}}(\mathtt{M}) &- S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &+ \big(G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g})\big)\,\Gamma \\ &+ \big(L(\mathtt{L}\&\mathtt{M}) - L(\mathtt{L}\&\text{-}) - W(\text{-})\big)\Theta \\ &+ L(\mathtt{L}\text{-}\&\mathtt{M})\,\Xi \end{aligned} \tag{11}$$

and (6) is used in all three cases for a.

The factor φ. Two of the three variables have been transformed into "simpler" forms (6) and (11); at least they are simpler to estimate.

The value φ can be set to 1 with the following reasoning: The overfull line is removed as soon as possible, so the maximum allowed stretchability will be used that results in the badness τ. See the data in (3iii), (3iv), (3ii), and (3i) which show the badness values after the overfull line disappears for the cases a), b′), c), and d), resp. That φ is sometimes less than 1 in experiment 1 can be explained by the fact that the increment for the stretchability was done in steps by 0.1 pt instead of 1 sp.

For example, if, let's say, 0.00001 pt is needed as additional stretchability when e_ι has already been applied, the situation can be described by the following approximation using $\tau = 200$:

$$200 \approx 100 \left(\frac{u + 0.00001\,\mathrm{pt}}{a + e_\iota + 0.00001\,\mathrm{pt}}\right)^3,$$

but

$$\alpha \approx 100 \left(\frac{u + 0.00001\,\mathrm{pt}}{a + e_\iota + 0.1\,\mathrm{pt}}\right)^3$$

is computed with the strategy of experiment 1 where $e_{\iota+1} = e_\iota + 0.1\,\mathrm{pt}$. If we divide the left side of the second equation by the left side of the first equation, and the right side of the second equation by the right side of the first equation, the quotients are

$$\frac{\alpha}{200} \approx \left(\frac{a + e_\iota + 0.00001\,\mathrm{pt}}{a + e_\iota + 0.1\,\mathrm{pt}}\right)^3.$$

With $a + e_\iota = 5\,\mathrm{pt}$ the quotient on the right hand side is $(5.00001\,\mathrm{pt}/5.1\,\mathrm{pt})^3 \approx 0.98^3 = 0.941192$, so that $\alpha \approx 188$. And with $a + e_\iota = 10\,\mathrm{pt}$ α gets no larger than 194.

Again, this argument is only valid if $a > 0\,\mathrm{pt}$, as otherwise TEX cannot make use of the additional stretchability, i.e., if the line has no place to stretch additional stretchability cannot change the output.

The result. With $\varphi = 1$ and equations (6) and (11) the optimal value of the dimen \emergencystretch e is given for the abovementioned cases by

$$\begin{aligned} e \approx \sqrt[3]{\frac{100}{\tau}}\Big(&W_{\mathrm{nw}}(\mathtt{M}) - S^-_\epsilon(\mathtt{M}) - o - l^- - S^-_\epsilon(\mathtt{L}) - r^- \\ &+ \big(G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}) - G^-_\epsilon(\Phi(\mathtt{L}), \mathtt{g})\big)\,\Gamma \\ &+ \big(L(\mathtt{L}\&\mathtt{M}) - L(\mathtt{L}\&\text{-}) - W(\text{-})\big)\Theta \\ &+ L(\mathtt{L}\text{-}\&\mathtt{M})\,\Xi\Big) \\ &- l^+ - S^+_\epsilon(\mathtt{L}) - r^+ \end{aligned} \tag{12}$$

if $l^+ + S^+_\epsilon(\mathtt{L}) + r^+ > 0\,\mathrm{pt}$.

5 Numerical calculations

Approximation (12) looks rather complicated with its many parameters but some of them are zero when others are nonzero. Several parameters are determined by the font used and some are specified via the line-breaking parameters. Here are the relevant values for the situation of experiment 1.

1. First, the environmental condition ϵ is 0, meaning that the font-related parameters have to be used. These have the following values in `cmr10`:

1a. `\fontdimen2` $= f_2 = 3.33333\,\mathrm{pt}$
1b. `\fontdimen3` $= f_3 = 1.66666\,\mathrm{pt}$
1c. `\fontdimen4` $= f_4 = 1.11111\,\mathrm{pt}$
1d. `\fontdimen7` $= f_7 = 1.11111\,\mathrm{pt}$
1e. and it means that $\mathbf{s} = \mathbf{z}$ as well as that $\mathbf{x} = \mathbf{z}$.

The `\tolerance` equals 200 in `plain` TEX when the emergency pass is performed so

2. $\sqrt[3]{100/\tau} = \sqrt[3]{0.5} \approx 0.794$.

The `plain` format leaves both `\leftskip` and `\rightskip` zero.

3. $\mathbf{l} = \mathbf{z}$, i.e., $l^+ = 0\,\mathrm{pt}$ and $l^- = 0\,\mathrm{pt}$
4. $\mathbf{r} = \mathbf{z}$, i.e., $r^+ = 0\,\mathrm{pt}$ and $r^- = 0\,\mathrm{pt}$

Two values can be determined for all `cm` fonts:

5. $L(\texttt{T\&-}) = 0\,\mathrm{pt}$ for all `T` not ending in a hyphen,
6. $L(\texttt{S-\&T}) = 0\,\mathrm{pt}$ for any text `S` and all `T` not beginning with a hyphen.

A hyphen has no kerning with any other character and ligatures are built only with other hyphens. But a hyphen is never inserted when explicit hyphens are present to break the line.

The width of a hyphen in `cmr10` is:

7. $W(\texttt{-}) = 3.33333\,\mathrm{pt}$.

The following parameters are not used in the calculations as no math typesetting occurs in the experiment. Nevertheless the values are given for completeness:

8. `\fontdimen6\textfont2` $= f_6^{\mathrm{t}2} = 10.00002\,\mathrm{pt}$ as the `\textfont2` is `cmsy10`;
9. `\medmuskip` $= (4\,\mathrm{mu}, 2\,\mathrm{mu}, 4\,\mathrm{mu})$ and therefore $\mathbf{m} = (4, 2, 4) \times 10.00002/18\,\mathrm{pt}$ which gives the glue $(2.22223\,\mathrm{pt}, 1.11111\,\mathrm{pt}, 2.22223\,\mathrm{pt})$;
10. similarly, `\thickmuskip` $= (5\,\mathrm{mu}, 5\,\mathrm{mu}, 0\,\mathrm{mu})$ so $\mathbf{t} = (5.55557\,\mathrm{pt}, 5.55557\,\mathrm{pt}, 0\,\mathrm{pt})$.

The next parameters depend upon the content of the line and the type of the break. The value of o is shown by TEX in the message about the overfull line. From this message the possible type of break, i.e., Γ, Θ, or Ξ can be determined.

	case a)	case b′)	case c)	case d)
11. o/pt =	5.88911	1.97240	12.36128	2.3335
12. Ξ =	0	0	0	0
13. Θ =	0	0	0	1
14. Γ =	1	1	1	0

Then `L` and $\Phi(\texttt{L})$ as well as two values of the glue $\mathbf{g} = (g^\circ, g^+, g^-)$ where the break occurs if $\Gamma = 1$ are calculated. The values of the $\mathbf{S}_\epsilon$ function can be counted by the method described in section 2.

	case a)	case b′)	case c)	case d)
15. (end of) `L` =	galaxy	named	or	happi
16. $\Phi(\texttt{L})$ =	1000	1000	1000	n/a
17. g° =	f_2	f_2	f_2	n/a
18. g^- =	f_4	f_4	f_4	n/a
19. $S_\epsilon^+(\texttt{L})$ =	$3f_3$	$2f_3$	f_3	$2f_3$
20. $S_\epsilon^-(\texttt{L})$ =	$3f_4$	$2f_4$	f_4	$2f_4$

The text `M` is shown by TEX in the message about the overfull line, but the value $W_{\mathrm{nw}}(\texttt{M})$ must be either measured or guessed. For experiment 1 the values have been documented in (1) and (2).

	case a)	case b′)	case c)	case d)
	`M` = called	R. J.	"R.	est
21. $W_{\mathrm{nw}}(\texttt{M})$/pt =	25.00005	21.38892	15.13892	12.27779
22. $S_\epsilon^-(\texttt{M})$ =	0	f_4	0	0
23. $L(\texttt{L\&M})$ =	n/a	n/a	n/a	0 pt

Although 23 parameters have been listed, yet more are involved which are hidden in other parameters. One is the set of `\sfcode`s for all the characters, which are used in the calculation of $S_\epsilon^-()$ and $S_\epsilon^+()$. Another is the font size (or the width of the characters), which is important for $W_{\mathrm{nw}}(\texttt{M})$.

As mentioned above, a and u can be derived from the trace data written by TEX, but only after the overfull line has been resolved. The output from `\tracingoutput` can be used: The sum of the stretchability of a line gives a and the multiplication with the *glue set* value computes u ([3], p. 75 and p. 79, resp.). The values are listed in lines A and B, their product u rounded up to five places is given in C.

	case a)	case b′)	case c)	case d)
A. stretch/pt =	4.99998	3.33332	1.66666	3.33332
B. glue set =	3.59998	5.49165	2.33325	1.31664
C. A ∗ B/pt =	17.99983	18.30543	3.88873	4.38878
D. a/pt (6) =	4.99998	3.33332	1.66666	3.33332
E. u/pt (11) =	17.99963	18.30541	3.88875	4.38874
F. e/pt (12) ≈	9.286	11.196	1.4198	0.15
G. e/pt ex. 1 =	9.3	11.2	1.5	0.2

The calculated values in line F agree with the stepwise measured data in row G. In the rest of this article, the computations round up the value of e to one decimal place; a higher precision is not needed.

Macros. When the parameter `\showboxbreadth` is large enough, say 100 for lines with 60–70 characters, the complete overfull line is written by TEX to the log file and there all stretch and shrink values of the glue can be found. But since the glue set ratio is reported as −1.0 in an overfull line, only the value of a can been determined from the message.

The optimal value for `\emergencystretch`

Nevertheless, macros can be written to make the calculation from the message shown. In this subsection a brief description for the design of a set of macros based on (12) is given.

Except for the cube root the calculation uses only simple arithmetic, which is available in TEX. For the cube root I use the formula $\sqrt[3]{\alpha^3+\beta} \approx \alpha + \beta/(3\alpha^2)$. The number of parameters makes the calculations a little bit complicated but TEX can do it, especially as it knows all the font parameters, the `\tolerance`, and the width of strings. A set of macros can be designed that receives the data about the moved and the kept material, the type of the expected line break, and the overhang to perform the calculation to get e.

Usually only five values must be specified as parameters or by macro names — three of them are displayed by TEX and two are known by the user. The other parameters can be calculated by TEX, although a user should be able to change them.

The following four macros are used, for example, to compute the value of `\emergencystretch` in case a) of experiment 1:

```
1. \dataEtextM(called)
2. \dataEtextL(in a distant galaxy)
3. \breakEglue(1000)
4. \calcEoverhang(5.88911pt)
```

1. The first macro receives the text `M`. It allows the calculation of $W_{\mathrm{nw}}(\mathtt{M})$ and $S^{-}_{\epsilon}(\mathtt{M})$.
2. The second macro gets the text `L` and it determines its stretch and shrink units $S^{+}_{\epsilon}(\mathtt{L})$ and $S^{-}_{\epsilon}(\mathtt{L})$.

   ▪ Now a is known and for u only the three terms which are multiplied by Γ, Ξ, and Θ as well as o are missing.
3. One of the following five macros is called to specify the type of break:
   a) `\breakEhyphen`,
   b) `\breakEexhyphen`,
   c) `\breakEmath` with a parameter for the atom (Bin or Rel) after which the break occurs, and a glue specification for user-entered glue,
   d) `\breakEotherglue` — used when the glue is, for example, an `\hskip` — which gets the three dimen values g°, g^{+}, and g^{-}; or it is a break with glue `z` at penalty and
   e) `\breakEglue` that has one parameter: $\Phi(\mathtt{L})$.

   ▪ Now $u + o$ is known.
4. The last macro receives the value o as parameter. It calculates u, the factor with the cube root and determines e rounded up to one decimal place.
5. All macros use the `plain` TEX defaults for their calculation. As mentioned above, to change the font, `\leftskip`, etc., some macros are written that can be called before the first macro to specify different values.

This makes it easy to find e for the original five cases of overfull lines in experiment 1:

Case a): 9.3 pt Case c): 1.5 pt
Case b): 10.7 pt Case d): 0.2 pt
Case e): 5.6 pt

and for the two created overfull lines:

Case b′): 11.2 pt Case d′): 6.7 pt

although the theory of section 4 does not apply to the cases b), e), and d′). So their computed values do not agree with the measured ones.

6 Line-breaking parameters

In the approximation (12) for e many different values are used, and it seems useful to discuss some of them. As `\tolerance` plays a very prominent rôle, and is the only parameter with a non-linear relationship to `\emergencystretch`, the group of line-breaking parameters is analyzed first.

The `\tolerance`. The dimen `\emergencystretch` was introduced to avoid large values of `\tolerance`. The latter is more of a document-wide parameter, while the former should be applied to a single paragraph. An increase of `\tolerance` in order to lower the value of `\emergencystretch` is therefore not a good idea: Tight lines cannot become tighter and loose lines benefit from `\emergencystretch`.

When (5) is written as a function of the parameter `\tolerance` for an overfull line

$$f(\xi) = \sqrt[3]{100}u\,\xi^{-1/3} - a, \quad \xi > 0,$$

it is a monotone decreasing function as in a situation with an overfull line $u > a > 0$ and its derivative is < 0. For $\xi \to \infty$ the limit is $-a$. As usual, values above 10000 are of no use in the application and the `\emergencystretch` cannot be a negative distance.

With $\xi_0 = 100$, which is the `plain` TEX default value of `\pretolerance`, the function value $f(\xi_0)$ is $u - a > 0\,\mathrm{pt}$ so there must be a ξ_1 for which the function becomes zero. This is $\xi_1 = 100(u/a)^3$, which is the approximation for the badness. Or in other words: With the real badness values, as in (3iv′), the required `\emergencystretch` is 0 pt. The value $f(\tau)$ computes the right hand side of (5) so it is the additional stretchability e that is needed to resolve the overfull line.

A concrete numerical example might help to understand this better. For case d) the values $u = 4.38878\,\mathrm{pt}$ and $a = 3.33332\,\mathrm{pt}$ were stated above.

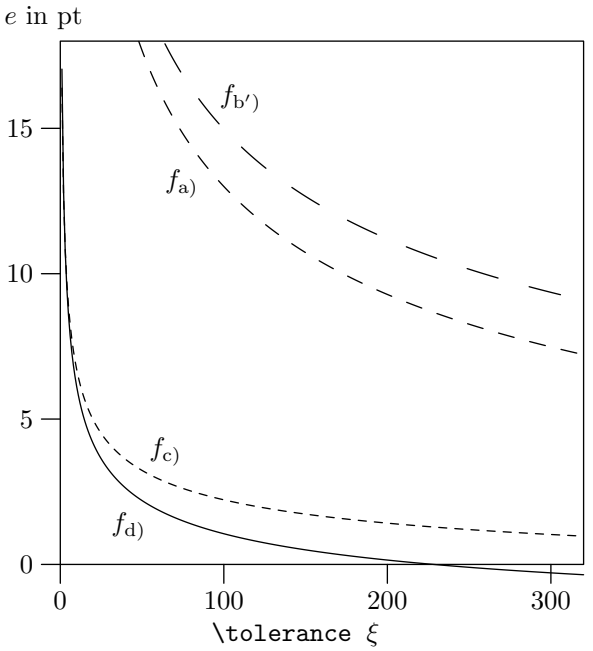


Figure 6: Graphs of function f for the cases discussed in section 4

As $\sqrt[3]{100} \approx 4.642$ the function becomes

$$f_{\mathrm{d})}(\xi) = 20.37091\,\xi^{-1/3}\,\mathrm{pt} - 3.33332\,\mathrm{pt}.$$

The other three cases a), b′), and c) have similar formulas. The graphs of all four functions are drawn in Fig. 6. It shows the typical curve of these functions: A small ξ has a high value but the function values drop quickly as ξ is increased. This effect slows down and a larger ξ reduces the required `\emergencystretch` by only a small amount. The value of $\xi = \tau = 200$ gives e.

Hyphenation. In the context of the removal of overfull lines the assumption was made that the hyphenation of words is a valid option. This might not always be the case. There are several primitive TEX commands that prevent hyphenation completely or for certain words only: The assignment of 10000 to the penalty `\hyphenpenalty` and its companion `\exhyphenpenalty` suppresses hyphenation completely as does the assignment `\hyphenchar` $= -1$. And `\uchyph` $= 0$ suppresses hyphenation of words that start with an uppercase letter. All these settings influence the possibility of hyphenating certain words and therefore the values that are available to remove overfull lines with `\emergencystretch` might increase. If hyphenation is suppressed only Γ can be 1, Θ and Ξ must be 0 in (12).

The commands act like switches, not continuous functions. As the final result of experiment 1 doesn't hyphenate a word which starts with an uppercase letter, the switch `\uchyph` $= 0$ only changes the initial situation ("Drofnats" cannot be hyphenated) but the data for `\emergencystretch` doesn't change, as the following experiment shows. Its results are found by the procedure used in experiment 1.

Experiment 2: Description
Suppress hyphenation of words that start with an uppercase letter.

TEX definitions

```
\uchyph=0
```

Result (experiment 1 vs. experiment 2)
a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
2a): `\uchyph`=0
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt □

The next two experiments show that the penalties `\hyphenpenalty` and `\exhyphenpenalty` act as switches. The badness of an overfull box is reported as 1000000 [3, p. 229]. An overfull line is so bad in TEX's opinion that no finite setting for the hyphenation parameters should make a difference.

Experiment 3: Description
All penalties are set to the maximum finite value and all demerits are set to a high value.

TEX definitions

```
\hyphenpenalty=9999 \exhyphenpenalty=9999
\linepenalty=9999 \finalhyphendemerits=1000000
\adjdemerits=1000000 \doublehyphendemerits=1000000
```

Result (experiment 1 vs. experiment 3)
a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
3a): `\...penalty`=9999 and `\...demerits`=1000000
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt □

The situation is different when hyphenation is completely suppressed, since the final result of experiment 1 contains hyphenated words.

Experiment 4: Description
Hyphenation of all words is suppressed.

TEX definitions

```
\hyphenpenalty=10000 \exhyphenpenalty=10000
```

Result (experiment 1 vs. experiment 4)
a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
4a): `\hyphenpenalty`=`\exhyphenpenalty`=10000
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 15.2 pt 5.6 pt □

The necessary values for e are found with (12) when only the break at glue or penalty is considered although TEX will still show hyphenation points in the messages for the overfull lines. Note: case e) is now covered by (12); the computation at the end of section 5 gives e.

The optimal value for `\emergencystretch`

7 Character- and font-related parameters

Some of the parameters of (12) are directly related to the properties of characters. Among them are the `\sfcode` and, closely related, the `\spacefactor`. But the largest group of parameters in (12) are related to the font used; for example, there are the `\fontdimen` 2, 3, 4, and 7 and the ligtable entries $L(\texttt{L\&-})$, $L(\texttt{L\&M})$, and $L(\texttt{L-\&M})$. And with $W_{\mathrm{nw}}(\texttt{M})$ the width of a character string occurs. Some of the listed values can be changed by an author with the help of TEX; others must be handled as constants.

The `\spacefactor`. The integer `\spacefactor` is an important characteristic for horizontal mode and the `\sfcode` assigned to each character modifies it. (The assignments are made by INITEX and the format, so the `\sfcode` is not directly a font-related parameter.) In (12) the four parameters $S_\epsilon^-(\texttt{M})$, $S_\epsilon^-(\texttt{L})$, $S_\epsilon^+(\texttt{L})$, and $\Phi(\texttt{L})$ are affected by all of these. In this subsection the relation of `\spacefactor` to the formula for `\emergencystretch` is analyzed; a TEX command to change the `\sfcode`s is discussed in the next subsection.

Experiment 5: Description
Set `\spacefactor` = 1200 if it is 1000 before a space.

TEX definitions

```
\def\1{\ifnum\spacefactor<1200 \count255=1200
       \else \count255=\spacefactor
       \fi \spacefactor=\count255 }
```

TEX input

```
Once\1 upon\1 a\1 time,\1 in\1 a\1 distant\1
galaxy\1 called\1 \"O\"o\c c,\1 there\1 lived\1
a\1 computer\1 named\1 R.\1~J.\1 Drofnats.

Mr.\1~Drofnats---or\1 ``R.\1 J.,''\1 as\1 he\1
preferred\1 to\1 be\1 called---was\1 happiest\1
when\1 he\1 was\1 at\1 work\1 typesetting\1
beautiful\1 documents.
```

Result (experiment 1 vs. experiment 5)
a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
5a): `\spacefactor`=1200 instead of 1000
b): 8.3 pt 8.3 pt/10.5 pt 1.5 pt 0.0 pt/1.5 pt 1.5 pt

As expected, the increase by 20% of the stretchability per space reduces the required value for the `\emergencystretch`. As $0.2f_3 = 0.2 \times 1.66666\,\mathrm{pt} = 0.333332\,\mathrm{pt}$ the values of cases a) and b) with three spaces can be up to $\approx 1\,\mathrm{pt}$ smaller and this is what the experiment shows. Of course the amount in the resolved line is stretched to its maximum and the additional stretchability is used to its maximum too.

For case b′) with two spaces it is only $\approx 0.7\,\mathrm{pt}$ and case c) stays at 1.5 pt as no space was changed. Case d) vanishes and the cases d′) and e) do not change as the added material is more important than the change in the stretchability of the spaces.

The `\sfcodes`. A `plain` TEX control sequence that changes the `\sfcode`s of the punctuation marks `. ? ! : ; ,` is `\frenchspacing`. It makes a space after a punctuation mark equal to a normal space for any text `T`; let's call this space **g**. But in our cases its use will not change anything because there are so few punctuation marks in the text: three commas and seven periods. The space factor of the periods has no influence on the text as two of the seven periods are at the end of the paragraphs, two are followed by a tie which resets the value to 1000, and three follow an uppercase letter so that the value is changed from 999 to 1000. And the three commas are never part of an overfull line.

Therefore, experiment 1 can be treated as if the command `\frenchspacing` was given and so the equalities $S_\epsilon^-(\texttt{L}) = \nu_{\sqcup}(\texttt{L})g^-$ and $S_\epsilon^+(\texttt{L}) = \nu_{\sqcup}(\texttt{L})g^+$ hold. Approximation (12) can be stated as a function of $\xi = \nu_{\sqcup}(\texttt{L})$, the number of spaces in `L`:

$$\begin{aligned} g(\xi) = &-\left(\sqrt[3]{\frac{100}{\tau}}g^- + g^+\right)\xi \\ &+ \sqrt[3]{\frac{100}{\tau}}\Big(W_{\mathrm{nw}}(\texttt{M}) - S_\epsilon^-(\texttt{M}) - o - l^- - r^- \\ &\qquad + \big(g^\circ - g^-\big)\Gamma \\ &\qquad + \big(L(\texttt{L\&M}) - L(\texttt{L\&-}) - W(\texttt{-})\big)\Theta \\ &\qquad + L(\texttt{L-\&M})\,\Xi\Big) - l^+ - r^+. \end{aligned}$$

It is a linear equation with negative slope. For `plain` TEX and `cmr10` $\sqrt[3]{100/\tau}\,g^- + g^+ \approx 2.43\,\mathrm{pt}$. So each space gives a relief of this amount when the command `\frenchspacing` is active.

Let's perform a concrete calculation to find the value of the intercept. For case d) three other terms are nonzero: $W_{\mathrm{nw}}(\texttt{M}) = 12.27779\,\mathrm{pt}$ by (2d), $o = 2.3335\,\mathrm{pt}$ by (1d), and as $\Theta = 1$ $L(\texttt{L\&M}) - L(\texttt{L\&-}) - W(\texttt{-}) = 0\,\mathrm{pt} - 0\,\mathrm{pt} - 3.33333\,\mathrm{pt} = -3.33333\,\mathrm{pt}$, so $0.794 \times (12.27779 - 2.3335 - 3.33333)\,\mathrm{pt} \approx 5.2491\,\mathrm{pt}$.

Similar equations can be created for the other three cases of section 4, as shown in Fig. 7 (next page). A small dot on the lines shows how many spaces are present in that case. As a parameter to g this gives the value of e because no other input can stretch or shrink. These spaces must additionally stretch; each space widens by `\emergencystretch` divided by the number of spaces in the line.

The `\fontdimen`s. The `\fontdimen` parameters are read by TEX from a font's tfm file. Although they

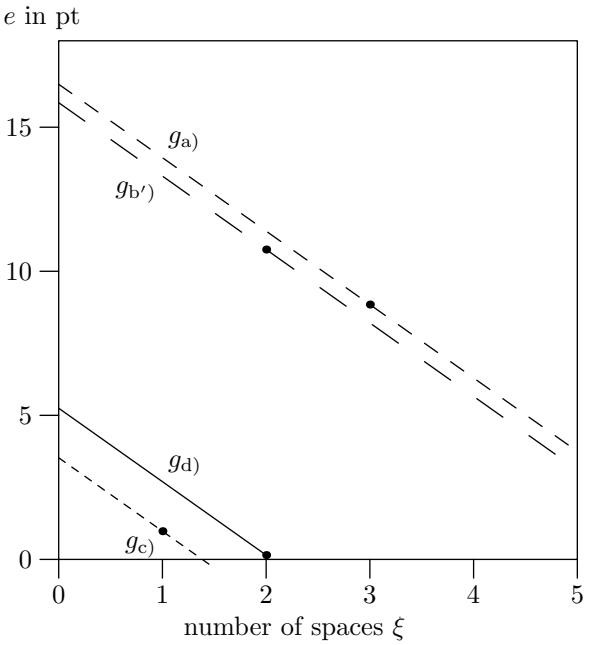


Figure 7: Graphs of function g for the cases discussed in section 4

can be changed by an author, this is not recommended except in very unusual situations. I consider them to be constants. Here is a list for several fonts:

f_ν	cmr10	cmbx10	cmr12	cmtt10
2	3.33333 pt	3.83331 pt	3.91663 pt	5.24995 pt
3	1.66666 pt	1.91666 pt	1.95831 pt	0.0 pt
4	1.11111 pt	1.27777 pt	1.30554 pt	0.0 pt
7	1.11111 pt	1.27777 pt	1.30554 pt	5.24995 pt
6	10.00002 pt	11.49994 pt	11.74988 pt	10.4999 pt

The `\fontdimen` parameters change when different sizes of the same font are used, as shown in the column for `cmr12` compared to the values for `cmr10`. For the *Computer Modern* fonts the relations $f_3 = f_2/2$, $f_4 = f_2/3$, and $f_7 = f_4$ seem to hold except for the monospaced font `cmtt10`. But it is better to express this relationship in terms of f_6, the quad width, also known as 1 em: $f_2 = f_6/3$, $f_3 = f_6/6$, $f_4 = f_6/9$, and $f_7 = f_6/9$.

The ligtable. The values must be treated as constants. For the font `cmr10` $L($`L&-`$) = L($`L-&M`$) = 0$ pt as explained in section 5. The dimension $L($`L&M`$)$ can be taken from Fig. 8. The values are small but they can be negative or positive.

The easiest way to get the values is to look at the *property list* of a font, which can be generated from the tfm file with the utility TFtoPL [7]. But note, sometimes letter pairs might have more than one entry, for example, in `cmr10` two values for the letter pair "ka" are specified [6, p. 37]. The first value counts, as explained in [5, p. 317]; so it does not

$L($`C&l`$)$ when letter `l` follows input of column `C`

C	−0.55556 pt	−0.27779 pt	0.27779 pt	0.55554 pt
a		v w y		j
b		v w x y	c d e o q	j
c		c k		
f		f i l		
ff		i l		
g			j	
h		b t u v w y		
k	a	c e o		
m		b t u v w x y		
n		b t u v w x y		
o		v w x y	c d e o q	j
p		v w x y	c d e o q	j
t		w y		
u		w		
v	a	c e o		
w		a c e o		
y		a e o		

Figure 8: Measured ligature and kerning values of lowercase letters in `cmr10`

seem to be a "mistake" [2, p. 322], but rather an optimization [11].

8 Paragraph shape parameters

The last group of parameters in (12) is formed by **l** and **r**, the `\leftskip` and the `\rightskip`. Sometimes these skips are used only with their natural width, for example, in the command `\narrower` but in (12) only the stretchability and shrinkability of the glue counts.

It might have been a surprise that the dimension `\hsize` has no influence on the required value of `\emergencystretch`. At least there is a basic relationship between the two as

$$0 \le \frac{\text{optimal value of } \texttt{\textbackslash emergencystretch}}{\text{value of } \texttt{\textbackslash hsize}} \le 1.$$

Therefore, the parameter `\hsize` is also analyzed.

No influence: `\hsize`. Of course, a change of the `\hsize` means different line breaks and that influences other parameters. Often overfull lines go away when the `\hsize` is changed. In this sense this dimension has influence. But when a concrete linebreaking situation is given — which is the precondition for the analysis that leads to (12) — it doesn't have any influence.

In the next experiment a "comparable" linebreaking situation for a larger `\hsize` is used. By "comparable" I mean a situation in which the line breaks are the same as with the previous `\hsize`. Therefore the content must be changed: Each line

gets more or less in its middle part a rule with a length equal to the increment of `\hsize`.

Experiment 6: Description
The `\hsize` is increased by 40 pt. As input, an instrumented version of the text of experiment 1 is used: In each line the control symbol `\0`, which represents a 40 pt long rule, is added before one word.

TEX definitions

```
\hsize=140pt \def\0{\hbox to 40pt{\hrulefill}}
```

TEX input

```
Once upon \0a time, in \0a distant galaxy
called \"O\"o\c c, there \0lived a computer
\0named R.~J. Drofnats.

Mr.~Drofnats\0---or ``R. J.,'' as \0he preferred
to be \0called---was happiest when \0he was at
work \0typesetting beautiful documents.
```

TEX output

Once upon _____a time,
in _____a distant galaxy called| $o = 5.88911$ pt
Ööç, there _____lived a com-
puter _____named R. J. Drof-| $o = 2.13905$ pt
nats.
Mr. Drofnats_____—or "R.| $o = 12.36128$ pt
J.," as _____he preferred to
be _____called—was happiest| $o = 2.3335$ pt
when _____he was at work
_____typesetting beautiful doc-| $o = 10.61136$ pt
uments.

Result (experiment 1 vs. experiment 6)
a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
6a): `\hsize`=140 pt with instrumented content
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt □

As expected, the experiment generates the identical values for eliminating overfull lines.

Ragged-right setting. As mentioned above the `\fontdimen` values should only be changed in very unusual situations; an example is given in [3], p. 355. If a change of the interword space is required the internal glue registers `\spaceskip` and `\xspaceskip` can be used as explained in section 2. An application occurs when text is typeset ragged-right with the help of a stretchable `\rightskip`.

The `plain` macro `\raggedright` initializes the ragged-right settings:

```
\def\raggedright{\rightskip=0pt plus2em
  \spaceskip=.3333em \xspaceskip=.5em\relax}.
```

The interword spaces are fixed and the `\rightskip` has a large value of stretchability: 2 em is $12 f_3$ in `cmr10`. So it is not surprising that the text of experiment 1 can be typeset ragged right without any overfull lines. But with an `\hsize` of 108 pt case a) returns.

Experiment 7: Description
Change the `\hsize` to 108 pt and typeset the text ragged right.

TEX definitions

```
\hsize=108pt \raggedright
```

TEX output

Once upon a time,
in a distant galaxy called| $o = 2.33217$ pt
Ööç, there lived a com-
puter named R. J. Drof-
nats.
Mr. Drofnats—or
"R. J.," as he preferred
to be called—was happi-
est when he was at work
typesetting beautiful
documents.

Result (experiment 1 vs. experiment 7)
a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
7a): `\hsize`=108 pt and `\raggedright`
b): 0.6 pt □

The theory developed in section 4 applies as all the different space types were included in the analysis; (12) computes the value: 0.6 pt.

9 More theory

Three cases of experiment 1 remain: b), d′), and e). In these cases text is not only moved to the next line but the preceding line hands over new material to the previously overfull line. This changes the picture completely as one of the basic elements on which the theory of section 4 is based is no longer true: The line that appears after applying the minimal `\emergencystretch` is no longer stretched to its maximum. See (3ii) for the case e) where the new badness is 2 and the white space *shrinks* as (3iv′) shows. I doubt that there is a precise approximation to calculate e from the output that contains the overfull line only if new material is added to the line.

As the cases b) and b′) show, it can happen that an overfull line is transformed into another overfull line before the computed `\emergencystretch` resolves the original problem. And the newly created overfull line needs a larger additional stretchability than the original one. Therefore a second calculation of e is sometimes necessary. The repeated application of the theory developed in section 4 is required.

A procedure. Compute with (12) all values for the overfull lines. Use the smallest value and apply it to the paragraph. For the selected overfull line and the other overfull lines the following possibilities exist.

1: Text is moved from the overfull line to the next line, no material comes from the previous line. This scenario was analyzed in section 4. There the problem is solved for the selected overfull line. Two results are possible:
   i) The computed value e works and no other overfull line is created — success. The cases a), b′), and c) belong to this scenario. Other overfull lines might disappear too, this happens in the cases d′) and e).
   ii) The computed value e works for the selected line, but a new overfull line is created. This is the case d); the procedure must be repeated.

2: Text is moved from the selected overfull line and new material is added to it. The computed value e, which would work for the original line, is not relevant to the changed line.
   i) The overfull line is gone and no other overfull line is created — success, but a smaller value of e might be successful too. No case of this kind occurs in experiment 1.
   ii) The overfull line is gone, but a new overfull line is created. This is the case b). The procedure must be repeated.

The procedure terminates as either the number of overfull lines is decreased or the new overfull line occurs later in the paragraph.

The remaining cases of experiment 1 are handled by this procedure. The only scenario that must be looked at further is 2i) as in other cases at most the procedure must be restarted.

The main question is: Why is some new material moved to the overfull line? Of course, TEX can use the value of `\emergencystretch` to stretch one or more previous lines with less text so that it finds line breaks which avoid the overfull line.

The missing scenario. This scenario is described more formally in order to have a clear definition. As the way in which the lines are broken, i.e., whether at glue, at penalty, at an explicit hyphen, or at an inserted hyphen, is not important in the following discussion that part is ignored. Assume that the line $\mathtt{L}_0\mathtt{M}_0$ is overfull and the method of section 4 has computed e_0. But when `\emergencystretch` is set to that value the former overfull line doesn't become $\mathtt{L}_0$ but $\mathtt{M}_{-1}\mathtt{L}_0$. Let $-\omega$ be the first preceding line that has not received material from its previous line; such a line must exist as the first line of the paragraph qualifies.

So the lines $\mathtt{L}_{-\omega}\mathtt{M}_{-\omega}$, $\mathtt{L}_{-\omega+1}\mathtt{M}_{-\omega+1}$, …, $\mathtt{L}_0\mathtt{M}_0$, in which the last line is overfull, are transformed into the lines $\mathtt{L}_{-\omega}$, $\mathtt{M}_{-\omega}\mathtt{L}_{-\omega+1}$, …, $\mathtt{M}_{-1}\mathtt{L}_0$.

Some of the new lines must stretch to benefit from e; let the index sequence $\kappa_{-\mu}, \ldots, \kappa_0$ represent the indices of lines that stretch. We have to calculate an e_{κ_ι} for all of these lines — although except for one they might not be overfull — to stretch them by the expected amount. The maximum dimension of these calculations is sufficient for `\emergencystretch` instead of the originally calculated e_0 to make all transitions happen.

Therefore two formulas are needed. One that computes e_0 from the transition $\mathtt{L}_0\mathtt{M}_0 \to \mathtt{M}_{-1}\mathtt{L}_0$; it can only give a useful result if $\mathtt{M}_{-1}\mathtt{L}_0$ stretches. The second formula computes the required amount of `\emergencystretch` that is needed to transform a non-overfull line into another: $\mathtt{L}_{\kappa_\iota}\mathtt{M}_{\kappa_\iota} \to \mathtt{M}_{\kappa_\iota-1}\mathtt{L}_{\kappa_\iota}$. The computation is valid only if $\mathtt{M}_{\kappa_\iota-1}\mathtt{L}_{\kappa_\iota}$ stretches.

The first formula. Figure 9 shows how the cases that were discussed in section 4 change when new material is added to the right side. Note the new material `N` might end in glue. It must be the empty string if `L` stands for the first line of a paragraph. So there is no problem with an `\indent` or `\noindent`.

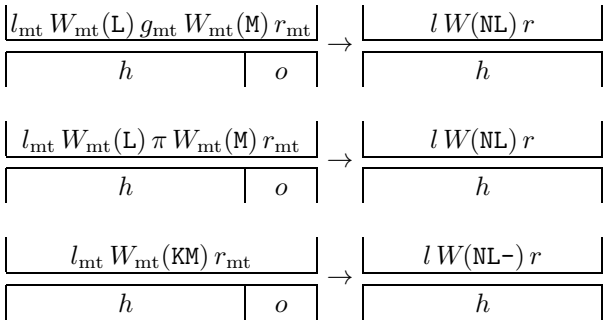


Figure 9: Resolve overfull line with new material

The right hand sides must stretch and they must need `\emergencystretch`, i.e., each real badness ϱ must be greater than τ to benefit from the following theory.

As only the right hand sides change, all equations for the left hand sides developed in section 4 are still valid. In the equations for the right hand sides the input `L` must be replaced by `NL`. The equation $W_{\mathrm{nw}}(\mathtt{NL}) = W_{\mathrm{nw}}(\mathtt{N}) + L(\mathtt{N\&L}) + W_{\mathrm{nw}}(\mathtt{L})$ shows that only two new terms occur in these equations; the equation is valid as `L` doesn't begin with ignored spaces as explained above.

Equation (6) changes to

$$\begin{aligned} a &= l^+ + S_\epsilon^+(\mathtt{NL}) + r^+ \\ &= l^+ + S_\epsilon^+(\mathtt{N}) + S_\epsilon^+(\mathtt{L}) + r^+. \end{aligned} \tag{13}$$

The optimal value for `\emergencystretch`

Equation (10) is no longer valid, now the computation is:

$$\begin{aligned} h &= l + W(\texttt{NL}) + r \\ &= l^\circ + W_{\text{nw}}(\texttt{NL}) + r^\circ + u \\ &= l^\circ + W_{\text{nw}}(\texttt{L}) + r^\circ + W_{\text{nw}}(\texttt{N}) + L(\texttt{N\&L}) + u. \end{aligned}$$

The equations for the left hand sides stay the same and with the techniques of section 4 the equivalent of (8) is:

$$\begin{aligned} u = W_{\text{nw}}(\texttt{M}) - S^-_\epsilon(\texttt{M}) - o - l^- - S^-_\epsilon(\texttt{L}) - r^- \\ - W_{\text{nw}}(\texttt{N}) - L(\texttt{N\&L}) \\ + \big(G^\circ_\epsilon(\Phi(\texttt{L}), \texttt{g}) - G^-_\epsilon(\Phi(\texttt{L}), \texttt{g})\big)\Gamma. \end{aligned}$$

The first two cases are handled; the last one is not difficult but the argument must involve the strings K and K′. One more figure might help:

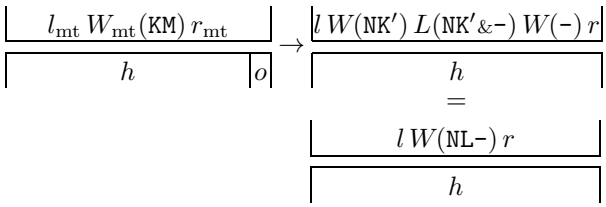


Figure 10: Break at hyphen and add new material

The replacement of (9) must deal with NK and NK′. The relationship becomes

$$\begin{aligned} W_{\text{nw}}(\texttt{NK}) + \big(L(\texttt{NK\&-}) + W(\texttt{-})\big)\Theta = \\ W_{\text{nw}}(\texttt{NK}') + L(\texttt{NK}'\texttt{\&-}) + W(\texttt{-}) \end{aligned}$$

and for the right hand side the new formula is

$$\begin{aligned} h &= l^\circ + W_{\text{nw}}(\texttt{NL-}) + r^\circ + u \\ &= l^\circ + W_{\text{nw}}(\texttt{NK}') + L(\texttt{NK}'\texttt{\&-}) + W(\texttt{-}) + r^\circ + u \\ &= l^\circ + W_{\text{nw}}(\texttt{NK}) + \big(L(\texttt{NK\&-}) + W(\texttt{-})\big)\Theta + r^\circ + u \\ &= l^\circ + W_{\text{nw}}(\texttt{N}) + L(\texttt{N\&K}) + W_{\text{nw}}(\texttt{K}) \\ &\quad + \big(L(\texttt{NK\&-}) + W(\texttt{-})\big)\Theta + r^\circ + u \end{aligned}$$

so

$$\begin{aligned} l^\circ + W_{\text{nw}}(\texttt{K}) + r^\circ = h - \big(L(\texttt{NK\&-}) - W(\texttt{-})\big)\Theta - u \\ - W_{\text{nw}}(\texttt{N}) - L(\texttt{N\&K}). \end{aligned}$$

The equation for the left hand side remains valid and its first three summands are replaced by the right hand side of the previous equation:

$$\begin{aligned} h + o = h - \big(L(\texttt{NK\&-}) + W(\texttt{-})\big)\Theta - u \\ - W_{\text{nw}}(\texttt{N}) - L(\texttt{N\&K}) - l^- - S^-_\epsilon(\texttt{K}) - r^- \\ + L(\texttt{K\&M}) + W_{\text{nw}}(\texttt{M}) - S^-_\epsilon(\texttt{M}). \end{aligned}$$

The final rearrangement involves the replacement of K by L as was done before.

$$\begin{aligned} u = W_{\text{nw}}(\texttt{M}) - S^-_\epsilon(\texttt{M}) - o - l^- - S^-_\epsilon(\texttt{L}) - r^- \\ - W_{\text{nw}}(\texttt{N}) - L(\texttt{N\&L}) \\ + \big(L(\texttt{L\&M}) - L(\texttt{NL\&-}) - W(\texttt{-})\big)\Theta \\ + L(\texttt{L-\&M})\,\Xi. \end{aligned}$$

Summary. Two new summands are added compared to (10), and all the formulas can be combined as in section 4 to get the equivalent of approximation (12):

$$\begin{aligned} e \approx \sqrt[3]{\frac{100}{\tau}}\Big(W_{\text{nw}}(\texttt{M}) - S^-_\epsilon(\texttt{M}) - o - l^- - r^- \\ - S^-_\epsilon(\texttt{L}) - W_{\text{nw}}(\texttt{N}) - L(\texttt{N\&L}) \\ + \big(G^\circ_\epsilon(\Phi(\texttt{L}), \texttt{g}) - G^-_\epsilon(\Phi(\texttt{L}), \texttt{g})\big)\,\Gamma \\ + \big(L(\texttt{L\&M}) - L(\texttt{NL\&-}) - W(\texttt{-})\big)\Theta \\ + L(\texttt{L-\&M})\,\Xi\Big) \\ - l^+ - S^+_\epsilon(\texttt{N}) - S^+_\epsilon(\texttt{L}) - r^+ \end{aligned} \tag{14}$$

if $\varrho > \tau$ and if there is stretchability in the line, i.e., $l^+ + S^+_\epsilon(\texttt{N}) + S^+_\epsilon(\texttt{L}) + r^+ > 0\,\text{pt}$.

The second formula. Now the situation changes completely as — to speak in the terms that we have been using — the left hand side is not overfull.

Case 1: Break at glue. Let's look at the situation again in the form of a simple picture, Fig. 11, the companion of Fig. 3.

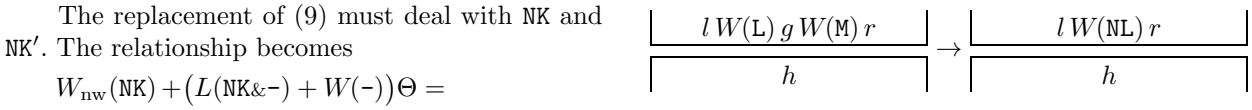


Figure 11: Break at glue and add new text

As before the text N might end in a glue item; the glue **g** might be user-entered or TEX-inserted.

Much of the information from the above Fig. 3 is lost. But on both sides the badness values are now finite, and as observed in the previous subsection for the first formula, the right hand side must stretch; further, it must stretch so much that the additional stretchability of `\emergencystretch` is required. In other words: The real badness ϱ of the right hand side must be larger than `\tolerance`. The glue of the left side has no restriction, i.e., it can shrink, stretch or use its natural width. To distinguish the common variable names for the right and the left sides the subscripts ρ and λ are used for a and u.

The right hand sides have been analyzed for the first formula and the value a_ρ is given by (13); as before, both sides give an equation for h:

$$\begin{aligned} h &= l^\circ + W_{\text{nw}}(\texttt{N}) + L(\texttt{N\&L}) + W_{\text{nw}}(\texttt{L}) + r^\circ + u_\rho. \\ &= l^\circ + W_{\text{nw}}(\texttt{L}) + G^\circ_\epsilon(\Phi(\texttt{L}), \texttt{g}) + W_{\text{nw}}(\texttt{M}) + r^\circ \\ &\quad + \delta u_\lambda. \end{aligned}$$

The factor $\delta \in \{-1, +1\}$ represents the fact that the value u_λ has to be added if the line on the left stretches otherwise it must be subtracted.

The equations can be rearranged to find the equation for u_ρ:

$$u_\rho = W_{\mathrm{nw}}(\mathtt{M}) - W_{\mathrm{nw}}(\mathtt{N}) - L(\mathtt{N\&L}) + \delta u_\lambda + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g}).$$

The finite badness values allow to write the following approximations (compare to (5)):

$$\rho \approx 100 \left(\frac{u_\rho}{a_\rho + e}\right)^3 \text{ or } u_\rho \approx \sqrt[3]{\frac{\rho}{100}}\,(a_\rho + e).$$

As above $\rho = \tau$ to get the minimal e:

$$e \approx \sqrt[3]{100/\tau}\,u_\rho - a_\rho. \qquad (**)$$

Therefore

$$e \approx \sqrt[3]{\frac{100}{\tau}} \Big(W_{\mathrm{nw}}(\mathtt{M}) - W_{\mathrm{nw}}(\mathtt{N}) - L(\mathtt{N\&L}) + \delta u_\lambda + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g})\Big) - l^+ - S^+_\epsilon(\mathtt{N}) - S^+_\epsilon(\mathtt{L}) - r^+$$

if stretchability is present in the line.

The term δu_λ could be analyzed further, but that leads to a lot of subcases. I think it is best to keep it in the formula.

Case 2: Break at penalty. This case is handled exactly like the previous one. If Γ is introduced, as in section 4, then the combined equation becomes

$$e \approx \sqrt[3]{\frac{100}{\tau}} \Big(W_{\mathrm{nw}}(\mathtt{M}) - W_{\mathrm{nw}}(\mathtt{N}) - L(\mathtt{N\&L}) + \delta u_\lambda + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g})\Gamma\Big) - l^+ - S^+_\epsilon(\mathtt{N}) - S^+_\epsilon(\mathtt{L}) - r^+.$$

Case 3: Break at hyphen. Again the formula for a_ρ is given by (13).

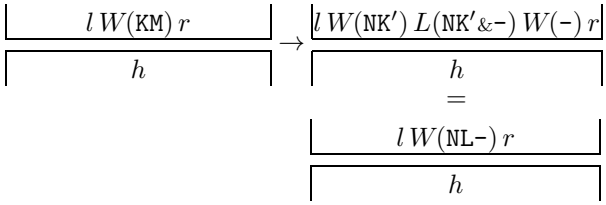


Figure 12: Break at hyphen and add new text

The right hand side was analyzed above for the first formula and the left hand side gives the equation:

$$h = l^\circ + W_{\mathrm{nw}}(\mathtt{K}) + L(\mathtt{K\&M}) + W_{\mathrm{nw}}(\mathtt{M}) + r^\circ + \delta u_\lambda.$$

Therefore

$$u_\rho = W_{\mathrm{nw}}(\mathtt{M}) - W_{\mathrm{nw}}(\mathtt{N}) + L(\mathtt{K\&M}) - L(\mathtt{N\&K'}) - \big(L(\mathtt{NK\&\text{-}}) + W(\mathtt{-})\big)\Theta + \delta u_\lambda$$

and with the replacement of K by L as it was done for the first formula it becomes

$$u_\rho = W_{\mathrm{nw}}(\mathtt{M}) - W_{\mathrm{nw}}(\mathtt{N}) - L(\mathtt{N\&L}) - \big(L(\mathtt{NL\&\text{-}}) - L(\mathtt{L\&M}) + W(\mathtt{-})\big)\Theta + L(\mathtt{L\text{-}\&M})\,\Xi + \delta u_\lambda.$$

Summary. The previous equation can be applied to (∗∗). The result isn't shown here as all cases can be combined directly as before into one equation. The approximation of e for the second formula is

$$e \approx \sqrt[3]{100/\tau}\,\Big(W_{\mathrm{nw}}(\mathtt{M}) - W_{\mathrm{nw}}(\mathtt{N}) - L(\mathtt{N\&L}) + \delta u_\lambda + G^\circ_\epsilon(\Phi(\mathtt{L}), \mathtt{g})\Gamma + \big(L(\mathtt{L\&M}) - L(\mathtt{NL\&\text{-}}) - W(\mathtt{-})\big)\Theta + L(\mathtt{L\text{-}\&M})\,\Xi\Big) - l^+ - S^+_\epsilon(\mathtt{N}) - S^+_\epsilon(\mathtt{L}) - r^+ \qquad (15)$$

if $\varrho > \tau$ and $-l^+ - S^+_\epsilon(\mathtt{N}) - S^+_\epsilon(\mathtt{L}) - r^+ > 0\,\mathrm{pt}$.

10 A second experiment

The theory is completely developed. The following experiment contains aspects not seen before, for example, an overfull line in math mode and a break at an explicit hyphen, and it shows a complete cycle to get rid off all overfull lines in a paragraph according to the theory of section 9.

Experiment 8: Description
Combine both paragraphs of experiment 1 and add a sentence with mathematics. Reduce `\hsize` by 2 pt.

TEX definitions

```
\hsize=98pt
```

TEX output

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats. Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents. In one text he proved $e^{i\pi}+1 = 0$.

$e_1 \approx 9.1\,\mathrm{pt}$ by (12)
$e_2 \approx 1.8\,\mathrm{pt}$ by (12)
$e_3 \approx 4.1\,\mathrm{pt}$ by (12)
$e_4 \approx 4.1\,\mathrm{pt}$ by (12)
$e_5 \approx 0.6\,\mathrm{pt}$ by (12)

As in experiment 1 there are five overfull lines. To remove them the procedure of section 9 that is based on the results of section 4 can be used. As described, e_5, the smallest of the five values, can be applied first, which of course resolves the last overfull line and only this line. Then the application of the next smallest value, e_2, resolves the second and fourth overfull lines and transfers the third one into a new overfull line. And so on.

But with the formulas of the previous section the maximum value can be used directly and if it is too large the value can be corrected; so the largest value e_1 is assigned to `\emergencystretch`.

Experiment 8 continued: TEX definitions

```
\emergencystretch=9.1pt
```

TEX output

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats. Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents. In one text he proved $e^{i\pi} + 1 = 0$.

$e_6 \approx 9.6\,\text{pt}$ by (12)
$e_7 \approx -0.0\,\text{pt}$ by (12)

The value of e_1 was too small to solve all problems; new overfull lines were created.

Note the value of e_7 is given as $-0.0\,\text{pt}$ — an impossible result. The reason lies in the content of the line. The space after "`Mr.`" in line 5 is entered as a tie (see the TEX input of experiment 1) so the line cannot be broken there. And "`Drofnats---`" cannot be hyphenated as it contains, in TEX's view, an explicit hyphen. Therefore the line break must occur after the end-of-sentence period. With the theory of section 4 this results in a line that would contain only "`Drofnats.`", i.e., the line has no stretchability. Therefore approximation (12) cannot compute a valid value for the `\emergencystretch` — expressed by the impossible value $-0.0\,\text{pt}$.

But as the other line reports a valid dimension e_6, this can be used in the next run to remove at least this overfull line.

Experiment 8 continued: TEX definitions

```
\emergencystretch=9.6pt
```

TEX output

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats. Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents. In one text he proved $e^{i\pi} + 1 = 0$.

$e_8 \approx 22.5\,\text{pt}$ by (12)

There is still one overfull line. The value 9.6 pt was too small but line 5 now gives a usable value as two spaces occur in the line after a break at the end-of-sentence period. The value e_8 removes the overfull line.

Experiment 8 continued: TEX definitions

```
\emergencystretch=22.5pt
```

TEX output

Once upon a time, in a distant galaxy called Ööç, there lived a computer named R. J. Drofnats. Mr. Drofnats—or "R. J.," as he preferred to be called—was happiest when he was at work typesetting beautiful documents. In one text he proved $e^{i\pi} + 1 = 0$.

$e_9 \approx 8.2\,\text{pt}$ by (15)
$e_{10} \approx 7.7\,\text{pt}$ by (15)
$e_{11} \approx 15.4\,\text{pt}$ by (15)
$e_{12} \approx 16.2\,\text{pt}$ by (15)
$e_{13} \approx -0.0\,\text{pt}$ by (14)

This is the scenario 2i) of the procedure in section 9. The check with the four preceding lines of the previously overfull line shows that the value 16.2 pt is sufficient to resolve this problem as the input "`named `" is moved to that line. The output for the `\emergencystretch` set to 16.2 pt isn't shown as it is identical to the version with 22.5 pt.

Note the value e_{13} cannot be computed as the former overfull line is transformed into a line of badness 0. Formula (14) needs a line that stretches and has a real badness larger than `\tolerance`.

11 Another application

In [14, example 5], `\emergencystretch` — together with `\looseness` to have a trigger for a third line-breaking pass — was used to remove a stack of hyphens. The first three lines of a paragraph end with an inserted hyphen. In the example the value of `\emergencystretch` was set to the width of the string "`Ar`", i.e., to 11.41669 pt to remove the stack. The text is taken from [9].

Experiment 9: Description

Remove a stack of hyphens via `\emergencystretch`; first show the original line breaking and then give the additional stretchability the amount 11.4 pt. Force TEX to use a third pass with the setting `\looseness` = 1.

TEX output

So instead, I worked only at Stanford, at the Artificial Intelligence Laboratory with the very primitive equipment there. We did have television cameras, and my publisher, Addison-Wesley, was very helpful — they sent me the original press-printed proofs of my book, from which *The Art of Computer Programming* had been made. The process in the 60s ...

So instead, I worked only at Stanford, at the Artificial Intelligence Laboratory with the very

primitive equipment there. We did have television cameras, and my publisher, Addison-Wesley, was very helpful — they sent me the original press-printed proofs of my book, from which *The Art of Computer Programming* had been made. The process in the 60s ... ∎

No overfull lines are output, but the second formula of section 9 calculates e for a transition that doesn't involve overfull lines. Here are the values for the first three lines of the second paragraph of the above output:

1st line: $e_1 \approx -0.0\,\text{pt}$ by (15)

2nd line: $e_2 \approx 6.9\,\text{pt}$ by (15)

3rd line: $e_3 \approx -0.0\,\text{pt}$ by (15)

So let's try what happens when the value of the dimen `\emergencystretch` is set only to 6.9 pt.

Surprise — nothing happens! The output looks identical to the first paragraph. Well, maybe it is not that much of a surprise as one of the principles on which the derived formula (15) is based is violated: TEX does not have to avoid an overfull line, and therefore it uses the line breaks that result in the minimal sum of the line demerits. The total for the text without the three hyphens is nearly 36,000 demerits higher than for the shown paragraph with $\approx$ 11.4 pt additional stretchability. Note that the second line, which determines the value of e, gets a badness of 200 by (15), the default `\tolerance`.

Either the penalty for hyphenation must be increased, for example, to 200 to compensate for the value of `\tolerance`, or `\emergencystretch` must be increased because then the lines that stretch get lower badness values, as observed in experiment 1. The correct amounts are not easy to determine. For example, setting `\hyphenpenalty` to 128 solves the problem, or setting `\emergencystretch` to the minimal 10.9 pt to create the output as shown above, or to 9.8 pt to get a solution that removes the three hyphens but creates a new one.

12 Final remarks

In this article theoretical results show how to compute the optimal value for `\emergencystretch` in the case when TEX produces an overfull line. Approximation (12) is always successful as long as the line contains stretchability and (14) and (15) extend the result but need an additional condition: the badness of the new line must be larger than the given `\tolerance`. (This condition is automatically fulfilled when (12) can be used.) The analysis provides some insight into the factors that influence the value for the additional stretchability.

One question has not been addressed yet: Why is a minimal value useful? As explained in [14], and observed in experiment 1, TEX uses the badness value computed from the available stretchability for its line-breaking decision and the stretchability that comes from `\emergencystretch`. High values for the latter assign to every stretchable line a low badness value, and thus TEX starts to prefer stretched lines in the paragraph during the line-breaking procedure, resulting in an excess of spaced-out lines.

Everyone who wants to use a nonzero value for `\emergencystretch` should be aware that an acceptable value depends on the available glue in a line — or in other words: the number of spaces in the line. A high average number of spaces in a text can tolerate higher values of `\emergencystretch`. And a line break with an inserted hyphen is an advantage. Although the following advice is not obeyed in all of the experiments in this article I recommend to apply a positive `\emergencystretch` only to a single paragraph at a time.

A rule of thumb. The accuracy with which e was computed in this article is not required, though. This allows us to create a *rule of thumb* to make the theory useful in everyday applications. A close look at (12) shows that many of the summands are zero or small if the `plain` TEX defaults and `cmr10` is used. Only four values are needed:

1. the amount that the line is too wide: o pt,
2. the number of characters in `M` which are moved to the next line,
3. the number of spaces in `L`, i.e., in the rest of the line,
4. the type of break: glue, penalty, explicit or inserted hyphen.

The rule of thumb is not very simple, maybe it is too complicated to be easily remembered:

$$4e \approx (15 \times \text{chars in } \texttt{M} - 11 \times \text{spaces in } \texttt{L} - 3 \times o + 8 - 18 \text{ if a hyphen is inserted})\,\text{pt}. \qquad \text{(RoT)}$$

Of course, if the number of spaces in `L` is zero, the computed value for e makes no sense as the input `L` has no stretchability.

For example, in experiment 1 the message

```
Overfull \hbox (5.88911pt too wide) in paragraph
\tenrm in a dis-tant galaxy called|
```

is shown and therefore $o \approx 5.89$, 6 characters are moved and three spaces remain when "called" is moved at a break at glue. Now use (RoT) to compute $4e \approx (90 - 33 - 17.67 + 8)\,\text{pt} = 47.33\,\text{pt}$ and therefore $e \approx 11.8\,\text{pt}$.

The other cases are treated in the same way.

The results compared to the measured values of experiment 1 are:

a) parameters; b) used `\emergencystretch`
1a): `\hsize`=100 pt
b): 9.3 pt 9.3 pt/11.2 pt 1.5 pt 0.2 pt/1.5 pt 1.5 pt
RoT a): calculated values with the *rule of thumb*
b): 11.8 pt 10.8 pt/13.7 pt 4.9 pt 1.4 pt/4.4 pt 6.2 pt

All computed values are large enough to remove the overfull line. Nevertheless there can be cases, for example, `M` is "WWW", when the factor 15 might be too small — although it looks as if the RoT computes values that are often too large. The simplification compared to (12), (14), and (15) comes with a price.

Aesthetics. The results given in this article are rather technical, and do not consider any aesthetic aspects. Of course, aesthetics cannot be measured objectively, as different individuals will prefer different aspects: hyphenation or not, justified or ragged-right text, etc.

This paper is already too long to discuss this, but one more TEX parameter might be briefly mentioned: `\hfuzz`. This has nothing to do with line-breaking decisions, it is used to suppress the warning message for overfull lines, nothing more. As long as an overfull line juts less than `\hfuzz` into the margin, TEX does not output a warning message. The `plain` format sets `\hfuzz` to 0.1 pt. Note that in section 10 the line preceding the overfull line, for which e_1 is calculated, is overfull by 0.02792 pt. It is not mentioned by TEX as the value is less than 0.1 pt. Sometimes higher values, for example, 1 pt, are suggested. But an overfull line that is 1 pt too wide is often easily detected by the eye.

An author may change his text to improve aesthetics but of course typesetting problems will always be present. Whatever the author or typesetter prefers or dislikes in a specific situation a manual intervention is sometimes necessary. Never forget: The visual output counts; it must always be checked, especially when `\emergencystretch` is used.

References

[1] Malcolm Clark, *A plain TEX primer*, Oxford University Press, 1992

[2] Bogusław Jackowski, Piotr Strzelczyk and Piotr Pianowski, "GUST e-foundry font projects," *TUGboat* **37**:3 (2016), 317–336
tug.org/TUGboat/tb37-3/tb117jackowski.pdf

[3] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984
The text of experiment 1 can be found in
ctan.org/tex-archive/systems/knuth/dist/lib/story.tex

[4] Donald E. Knuth, *TEX: The Program*, Volume B of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986

[5] Donald E. Knuth, *The METAFONTbook*, Volume C of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986

[6] Donald E. Knuth, *Computer Modern Typefaces*, Volume E of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986

[7] Donald E. Knuth, "The TFtoPL processor," in *TEXware*, Stanford Computer Science Report *STAN-CS-86-1097*, Stanford, California: Stanford University, 1986 (The first TFtoPL program was designed by Leo Guibas in the summer of 1978)
ctan.org/pkg/texware

[8] Donald E. Knuth, "The New Versions of TEX and METAFONT," *TUGboat* **10**:3 (1989), 325–328; "Erratum: The New Versions of TEX and METAFONT *TUGboat* Vol. 10, No. 3," *TUGboat* **11**:1 (1990), 12; reprinted as Chapter 29 in [10], 563–570
tug.org/TUGboat/tb10-3/tb25knut.pdf
tug.org/TUGboat/tb11-1/tb27erratum.pdf

[9] Donald E. Knuth, "CSTUG, Charles University, Prague, March 1996: Questions and Answers with Prof. Donald E. Knuth," *TUGboat* **17**:4 (1996), 355–367; reprinted as Chapter 32 in [10], 601–624
tug.org/TUGboat/tb17-4/tb53knuc.pdf

[10] Donald E. Knuth, *Digital Typography*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 78, 1999

[11] Jerry Leichter, "Re: Strange kern in cmr10," *TEXhax Digest* **89**:70 (1989), 25 July 1989
ctan.org/tex-archive/info/digests/texhax/89/texhax.70.gz

[12] Frank Mittelbach, Chris Rowley, Michael Downes, "Volunteer work for the LATEX3 project," *TUGboat* **13**:4 (1992), 510–513
tug.org/TUGboat/tb13-4/tb37mitt-l3.pdf
ctan.org/tex-archive/info/ltx3pub/vol-task.tex

[13] Philip Taylor, "A Pragmatic Approach to Paragraphs," *TUGboat* **14**:2 (1993), 138–140
tug.org/TUGboat/tb14-2/tb39taylor-para.pdf

[14] Udo Wermuth, "Tracing paragraphs," *TUGboat* **37**:3 (2016), 358–373
tug.org/TUGboat/tb37-3/tb117wermuth.pdf

◇ Udo Wermuth
Babenhäuser Straße 6
63128 Dietzenbach
Germany
u dot wermuth (at) icloud dot com

The Treasure Chest

This is a selection of the new packages posted to CTAN (`ctan.org`) from November 2016–March 2017, with descriptions based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. More information about any package can be found at `ctan.org/pkg/`*pkgname*. A few entries which the editors subjectively believe to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions.

We hope this column and its companions will help to make CTAN a more accessible resource to the TEX community. See also `ctan.org/topic`. Comments are welcome, as always.

◇ Karl Berry
`tugboat (at) tug dot org`

fonts

`arimo` in `fonts`
: Arimo fonts (Arial-compatible), with LaTeX support.

`arphic-ttf` in `fonts`
: Chinese Arphic fonts in TrueType format.

* `baskervillef` in `fonts`
: Design similar to Fry's Baskerville, with math support. [See article in this issue of *TUGboat*.]

`gandhi` in `fonts`
: Gandhi fonts, with LaTeX support.

* `gofonts` in `fonts`
: Go fonts designed by Bigelow&Holmes for the Go language project. [See item in this issue's editorial.]

`missaali` in `fonts`
: Late medieval OpenType textura font.

`montserrat` in `fonts`
: Montserrat fonts, with LaTeX support.

`paduak` in `fonts`
: TrueType font supporting the Myanmar script.

`ptex-fontmaps` in `fonts`
: Font maps and configuration for CJK fonts and (u)`ptex`, superseding `jfontmaps`.

`tinos` in `fonts`
: Tinos fonts (compatible with Times New Roman), with LaTeX support.

`txuprcal` in `fonts`
: Upright calligraphic based on `txfonts`.

`variablelm` in `fonts`
: Font definitions for variable Latin Modern fonts.

graphics

`awesomebox` in `graphics`
: Add admonition blocks using FontAwesome icons.

`callouts` in `graphics/pgf/contrib`
: Simple annotations and notes inside a Ti*k*Z picture.

`karnaughmap` in `graphics/pgf/contrib`
: Karnaugh maps in Ti*k*Z, with up to six variables and implicants.

`pst-shell` in `graphics/pstricks/contrib`
: Seashells in 3D using PSTricks.

`scsnowman` in `graphics/pgf/contrib`
: Snowman variants in Ti*k*Z.

`stanli` in `graphics/pgf/contrib`
: Structural analysis library for Ti*k*Z.

`tikz-kalendar` in `graphics/pgf/contrib`
: Typeset calendars with Ti*k*Z.

`tikzpeople` in `graphics/pgf/contrib`
: People-shaped nodes, such as 'chef', for Ti*k*Z.

info

* `docsurvey` in `info`
: Survey of LaTeX documentation. [See article in this issue of *TUGboat*.]

`forest-quickstart` in `info`
: Introduction to `forest`, for linguistic trees.

`mendex-doc` in `info`
: Mendex index processor manual.

`platexcheat` in `info`
: Japanese translation of Winston Chang's LaTeX cheat sheet, with additions.

* `undergradmath` in `info`
: Cheat sheet for writing math in LaTeX.

language/japanese

`jlreq` in `language/japanese`
: Document class supporting Japanese text layout requirements.

macros/latex

See also LaTeX news items in this issue of *TUGboat*, or `latex-project.org/news`.

`latexrelease,...` in `macros/latex/base`
: Use new TU encoding by default in XeLaTeX and LuaLaTeX, instead of OT1, among other changes and fixes.

`l3kernel,...` in `macros/latex/contrib`
: LaTeX3 packages now support (u)pTeX, among much other work.

macros/latex/contrib

`apxproof` in `macros/latex/contrib`
: Defer proofs to an appendix.

`conv-xkv` in `macros/latex/contrib`
: Support alternative key/value syntaxes.

`css-colors` in `macros/latex/contrib`
: Define the 143 named web-safe colors.

`delimset` in `macros/latex/contrib`
: Declare sets of delimiters with adjustable sizes.

`dtxdescribe` in `macros/latex/contrib`
: Additional object types in `dtx` files.

`eqnalign` in `macros/latex/contrib`
: Make the `eqnarray` environment behave like `align`.

`fgruler` in `macros/latex/contrib`
: Rulers on the page foreground or within text.

`fnspe` in `macros/latex/contrib`
: Math notation macros and shortcuts, developed for FNSPE CTU in Prague.

`footmisx` in `macros/latex/contrib`
: `footmisc` with support for `hyperref`.

`grayhints` in `macros/latex/contrib`
: Initial text in PDF form fields, via JavaScript.

`gtrlib-largetrees` in `macros/latex/contrib`
: Library for `genealogytree`, aimed at large trees.

`halloweenmath` in `macros/latex/contrib`
: Math symbols based on traditional Halloween iconography (pumpkins, witches, etc.).

`iscram` in `macros/latex/contrib`
: Class for ISCRAM articles (International Conference on Information Systems for Crisis Response and Management).

`keyfloat` in `macros/latex/contrib`
: Key/value interface for generating (sub)floats. [See article in this issue.]

`lion-msc` in `macros/latex/contrib`
: Class for theses at the Leiden Institution of Physics.

* `lwarp` in `macros/latex/contrib`
: Convert LaTeX to HTML5. [See article in this issue.]

`mpostinl` in `macros/latex/contrib`
: Embed MetaPost figures in a LaTeX 2$_\varepsilon$ document.

`mucproc` in `macros/latex/contrib`
: Class for "Mensch und Computer" conference contributions.

`numspell` in `macros/latex/contrib`
: Spelling cardinal and ordinal numbers in several languages.

`oplotsymbl` in `macros/latex/contrib`
: Unusual symbols used in scientific plots, etc.

`pxtatescale` in `macros/latex/contrib`
: Graphics driver support for pTeX vertical direction scaling.

`pythonhighlight` in `macros/latex/contrib`
: Highlighting Python code, based on `listings`.

`soup` in `macros/latex/contrib`
: Generate alphabet soup (word search) puzzles.

`studenthandouts` in `macros/latex/contrib`
: Manage and style student handout projects.

`wtref` in `macros/latex/contrib`
: Namespaces and scopes for LaTeX cross references.

`yaletter` in `macros/latex/contrib`
: Flexible macros for letters, envelopes, label sheets.

macros/latex/contrib/biblatex-contrib

`biblatex-archaeology` in `m/l/c/biblatex-contrib`
: Support for German humanities, especially the German Archaeological Institute.

`biblatex-arthistory-bonn` in `m/l/c/biblatex-contrib`
: Support for art historians.

macros/luatex/latex

`luahyphenrules` in `macros/luatex/latex`
: Loading patterns in `lualatex` independent of `babel`.

`novel` in `macros/luatex/latex`
: Class for authors of original fiction, particularly for print-on-demand.

macros/xetex/latex

`simple-resume-cv` in `macros/xetex/latex`
: Simple resume/CV template for XeLaTeX.

`simple-thesis-dissertation` in `macros/xetex/latex`
: Simple thesis/dissertation template for XeLaTeX.

`unicode-bidi` in `macros/xetex/latex`
: Mixing RTL and non-RTL without markup.

support

`fribidixetex` in `support`
: XeTeX preprocessor for supporting the Unicode bidirectional algorithm.

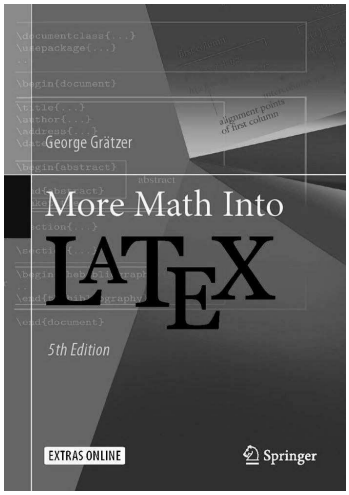
`gladtex` in `support`
: Python program to enable use of LaTeX math in HTML.

`tlaunch` in `support`
: Windows GUI to run networked TeX Live.

Book review: *More Math Into LaTeX*, 5th edition by George Grätzer

Jim Hefferon

George Grätzer, *More Math Into LaTeX*, 5th edition. Springer. 609 pp. Paperback, US$67.99. ISBN 978-3319237954.



Although *More Math Into LaTeX* by George Grätzer is a classic, I happen not to have studied it. So with the appearance of the fifth edition, I grabbed the chance. I was not disappointed.

The audience for this book is professionals such as working mathematicians or graduate students. The presentation is organized, thorough, and clear. The coverage is wide: certainly not everything about LaTeX is in here but it does cover, or at least provide a pointer to, the great majority of what an author would want to know to prepare a journal article, a book, or a presentation.

A technical book must balance being an introduction against being a reference. Professor Grätzer finesses the question by having a first chapter called *Short course*. This gets a beginner up and running. The rest of the book is suitable for dipping into when you have a specific problem, although you absolutely can read it as well.

This seems to me to be an excellent approach. Certainly if a graduate student preparing to write a thesis were asked to first master a six hundred page technical work, that would be discouraging. But here the short course is forty pages, eminently reasonable. The student could, in an afternoon, read the introduction, become familiar with the rest, and produce a first document or two.

It is an approach that expects that the reader is busy working on something else and I suspect most readers will be grateful for it.

More Math Into LaTeX has six Parts, and seven appendices. First is the short course mentioned above. Then comes the heart of the book, *Text and Math*. I'll talk more about this below. Part III, *Document Structure*, discusses such things as front and back matter, and also the `amsart` class. Then Part IV on *PDF Documents* covers hyperlinks, and also includes a chapter on presentations and a chapter introducing Ti*k*Z. Part V on *Customization* covers writing your own commands, including list environments. Finally, in Part VI, Professor Grätzer discusses things most relevant to *Long Documents*, such as BibTeX and *MakeIndex*, as well as tables of contents, etc.

The appendices include symbol tables for mathematics as well as for text, which I found useful almost immediately. I noted that contact information for the TeX Users Group is also included, with a suggestion that the reader may want to join. That's great to see, of course.

The longest Part of the book, nearly two hundred pages, and the one that will be the most often-thumbed is the second one, *Text and Math*.

This Part also typifies the feel of the entire book. It is filled with information. The chapter names will give you a sense of the extensive coverage: Typing text, Text environments, Typing math, More math, and Multiline math displays.

Just as one instance, there is a table with all of the font family switching commands. I can never remember the darn things and so right away onto that page went a sticky note, to ease finding it next time, in the heat of writing.

This Part is also filled with examples. I find that in working with students learning LaTeX, they start from examples. The ones given in *More Math Into LaTeX* are ones that come up in practice, which gets these students started, and also conveys to them that LaTeX is something they will find useful. That's very well done.

Professor Grätzer also gives advice, tips. There are documents in the LaTeX world where in my opinion the author moves past suggestions and into intrusiveness. Not so here. While I didn't find that every tip I read accords with my own sense of best practice, I did find that they are all reasonable.

One thing about this Part, and about the book as a whole, is its good sense of where users have problems. Anyone who has spent time in online TeX forums knows that there are themes to the questions that people ask. *More Math Into LaTeX* does a fine job heading off these questions, before they become a frustration.

There are many major LaTeX packages that the book does not cover. Just as one example, I'll cite *SIunits* (`http://ctan.org/pkg/siunits`) as something that a person may well find useful in preparing professional technical documents and that I don't find here. But no book can cover everything, and Professor Grätzer is wise not to try. The subject is just too extensive, and so this is not a deficiency.

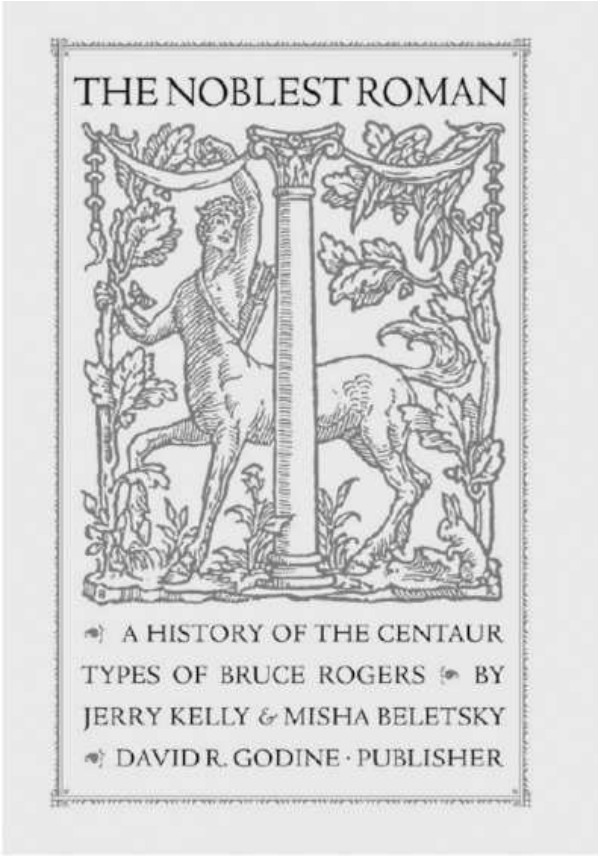
This is an excellent choice, for an individual or for an institutional library. It spans a great deal of the subject of LaTeX and does so without losing the reader. And, with this latest edition, it is up to date with recent developments. This classic continues on.

◇ Jim Hefferon
`jhefferon (at) smcvt dot edu`

Book review: *The Noblest Roman: A History of the Centaur Types of Bruce Rogers* by Jerry Kelly and Misha Beletsky

Boris Veytsman

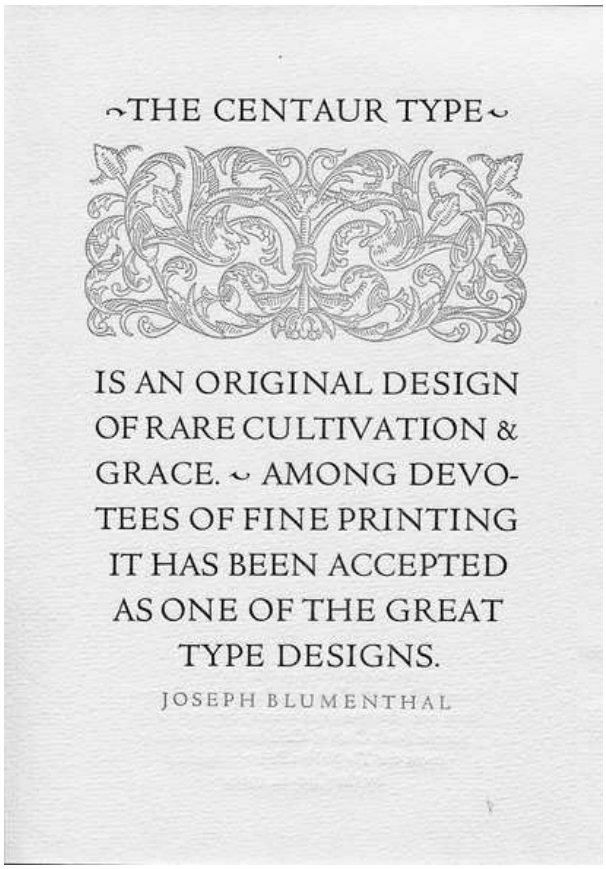
Jerry Kelly and Misha Beletsky, *The Noblest Roman: A History of the Centaur Types of Bruce Rogers*. David R. Godine, Publisher, Boston. 2016. 128 pp. Hardcover, US$45.00. ISBN 978-1-56792-582-1.



The history of fonts and font design is an important part of our heritage. The TEX community, in a tradition started by Knuth, appreciates the fine aspects of typography, and has always been keenly interested in fonts.

The *Centaur* font by Bruce Rogers is a classical font created at the beginning of the last century based on the immortal designs by Nicolas Jenson in the 15th century. The first full version of this font was cast in 1915. The book by Jerry Kelly and Misha Beletsky is a tribute to the centennial of this typographic treasure.

There are poets' poets and painters' painters: the writers and artists who foremost influenced their colleagues and inspired their creativity. Bruce Rogers probably deserves the title of typographers' typographer: his fame among book artists became, as another great typographer and Rogers' rival, Stanley Morison called it, "a BR cult". The quotation below by Joseph Blumenthal, beautifully typeset by Jerry Kelly for the Book Club of California edition of *The Noblest Roman*, well describes the opinion of professionals about *Centaur:*



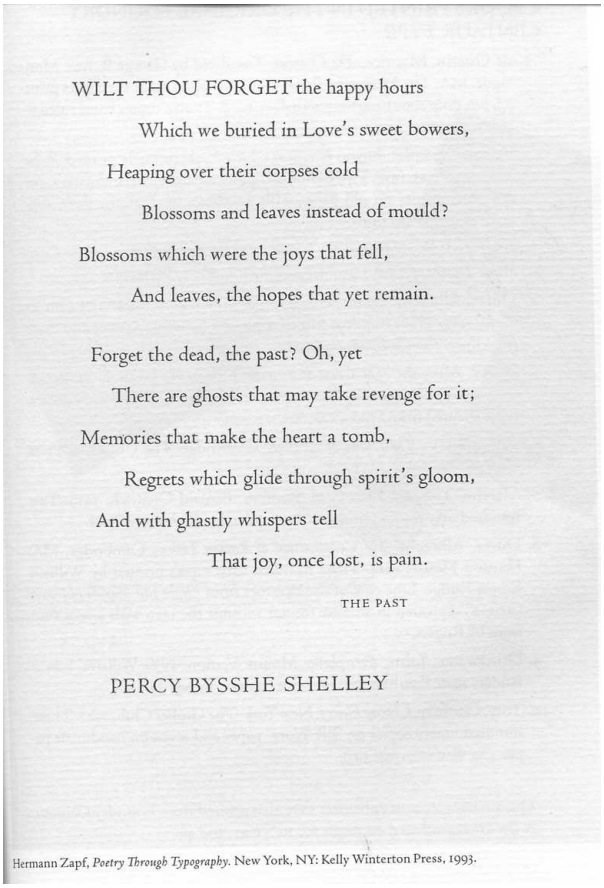
Kelly and Beletsky quote similar praise for the font by such masters as D. B. Updike and Robert Bringhurst.

Interestingly enough, this acclaim among artists, as often happens, did not translate into commercial success. As the authors note, during the entire production run of Monotype Centaur from 1929 to 1980 only 754 sets of matrices were sold — fifty times less than Times New Roman and ten times less than Monotype Garamond. The font is not well known by the general public. Maybe this is why we do not have free Centaur digital fonts for use with TEX; certainly such fonts would be a great contribution to the TEX typographic arsenal.

The authors note that the high regard for Centaur among typographers created many legends about its history. The incomplete and often unreliable recollections by Bruce Rogers himself, amplified by other narrators, lead to a situation where "little can

be taken at face value". Kelly and Beletsky took great pains to check details using archives, letters and other little explored sources. They performed a great service in elucidating the true story of Centaur. The book explores the relations between Rogers and his employees and contemporaries, the evolution of the design, and many interesting details, such as the attempt to create a version of the font for the Justowriter typesetting typewriter (1948).

While the book text is very interesting, its illustrations are truly superb. *The Noblest Roman* reproduces many pages of books typeset with Centaur. Below is the page from a book designed by Hermann Zapf:



WILT THOU FORGET the happy hours
Which we buried in Love's sweet bowers,
Heaping over their corpses cold
Blossoms and leaves instead of mould?
Blossoms which were the joys that fell,
And leaves, the hopes that yet remain.

Forget the dead, the past? Oh, yet
There are ghosts that may take revenge for it;
Memories that make the heart a tomb,
Regrets which glide through spirit's gloom,
And with ghastly whispers tell
That joy, once lost, is pain.

THE PAST

PERCY BYSSHE SHELLEY

Hermann Zapf, *Poetry Through Typography*. New York, NY: Kelly Winterton Press, 1993.

Besides comprehensive type samples, *The Noblest Roman* includes a list of books typeset with the original Centaur as well as detailed notes, a bibliography and a carefully compiled index.

The book is beautifully typeset by Jerry Kelly, whose work has been reviewed several times before in these pages. It is a masterpiece of the typographic art. The book is printed in two colors, with black and red marginal notes. The marginalia contain succinct biographies of typographers of this time, small illustrations and captions for larger illustrations in text:

Albrecht Dürer, *The Great Procession*. New York, privately printed at the Museum Press, 1919.

THE GREAT PROCESSION FROM THE CHURCH OF OUR LADY AT ANTWERP AS SEEN BY ALBRECHT DÜRER

WILLIAM IVINS, JR. (1881–1961) was appointed the first curator of prints and drawings at The Metropolitan Museum of Art in 1916. He retired in 1946, having built the collection up to more than 130,000 items. He published hundreds of articles, catalogs, and books, including his influential *How Prints Look*. He served as president of The Grolier Club from 1932 to 1935. Ivins's wife, Florence, created some illustrations for pamphlets designed by Rogers. Ivins served as acting director of The Metropolitan Museum of Art from 1933 to 1938, at a time when Henry Watson Kent was secretary to the museum.

Kent often turned to Rogers for help with designing Museum Press materials. While living in New York in 1916, Rogers was on a retainer with the museum and worked there a few days each week.[84] One of the most recognizable results of the Kent/Rogers collaboration was the adaptation, around 1925, of the letter *M* from Fra Luca da Pacioli's 1494 manual on the construction of roman letters[85] for use as the museum's logo.[86] This logo, which was in continuous use until 2016, may be the oldest and one of the most widely recognized elements of institutional identity in the history of graphic design.

The Centaur type is another lasting monument to the partnership of Kent and Rogers. Kent recognized the great potential of this font, and had The Metropolitan Museum of Art purchase the foundry type for exclusive use in their publications, aside from Rogers's use of the type for his own purposes. The selection of Centaur as a proprietary type for this august institution could not be more appropriate. Similar to the architectural face of the museum, the 1902 Beaux-Arts façade designed by Richard Morris Hunt, the type aspires to the classical ideal. But where the building,

M designed by Luca de Pacioli (from *Divina Proportione*, Venice: Paganinus, 1509). Adapted by Bruce Rogers for The Metropolitan Museum of Art's logo. The designer of this logo has been obscure, but it seems to have been Rogers's idea since it was included in the retrospective show of Rogers's work at The Grolier Club in 1939.

[47

To tell the truth, I am not sure why other notes, such as bibliographic references, are relegated to the back of the book rather than being typeset in the margins as well.

The book is typeset in Centaur (of course!) using three recent digital revivals of the font: the body text in the font by Jerry Kelly, the captions in the version by Toshi Omagari (used for the first time in this book), and displays in Monotype Centaur. The body text is justified with ragged right marginalia.

The publisher, David R. Godine, is well known for his books of highest quality, also often mentioned in these *TUGboat* review pages.

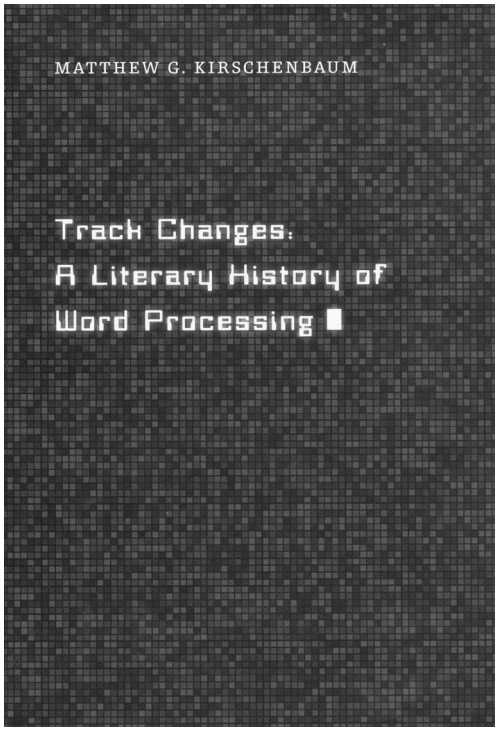
I thoroughly enjoyed this book and heartily recommend it to typophiles in the TEX community.

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

Book review: *Track Changes* by Matthew G. Kirschenbaum

David Walden

Matthew G. Kirschenbaum, *Track Changes: A Literary History of Word Processing*. The Belknap Press of Harvard University Press, 2016, xvi+344 pp. Hardcover, US$29.95. ISBN 978-0674417076.



In the Acknowledgments section of Matthew Kirschenbaum's *Track Changes*, he notes that in December 2011 he gave a lecture on his research at the New York Public Library, and a *New York Times* reporter wrote about his interest in the history of word processing.[1] The resulting "burst of publicity [flooded his inbox] with tips, anecdotes, contacts, and suggestions". Much of this helped the author develop his book; and, although the book is now published, the author says that "this is still very much an active project, and I'd be grateful to hear from any readers who have additional information".

The author is a professor in the English Department of the University of Maryland who "is a devoted scholar of all things digital and literary". In this book he studies the steps and impact on their writing of authors moving from pens or typewriters to word processors, from the earliest word processing kludges of equipment before word processing was really available as a product and continuing through writers who cling to "boutique" word processing systems (e.g., Scrivener and WriteRoom) in spite of the "hegemony" of Microsoft Word. While there is a sociological slant to the book, anyone who (a) has lived through the era of the development of interactive editors and word processors or (b) who is a reader of fiction (particularly genre fiction), will find in the book many authors, books, and systems to remember and a plethora of previously unknown information.

The book's style falls somewhere between creative non-fiction and an academic monograph. It is not based on a few characters used to drive a narrative ahead (like most books by, e.g., Tracy Kidder and Michael Lewis); rather, it's a collection of stories about hundreds of individual authors. Although it has 80 pages of notes and the chapter titles appear to be working on moving along an academic thesis, the book is easy and fun to read.

Knuth and TEX are mentioned once in the book, on page 30 where author Neal Stephenson is described as transcribing his *Baroque Cycle*, originally drafted longhand, into TEX using Emacs; then, when the publisher wanted to typeset the book in the QuarkXPress, Stephenson wrote a conversion program in Lisp.[2,3] Perhaps the longest of the hundreds of authors mentioned in the book is the story of author Len Deighton's adoption of early word processing technology (i.e., IBM's MT/ST system).

The book also has a nice index of systems and author names, which lets one see if one's favorite author is mentioned in the book or to look up history on an early word processing system such as WordStar. *Track Changes* is a pretty useful reference book as well as being fun reading. I am glad to have a copy.

To get a better feel for *Track Changes*, you can "Look Inside" on the book's Amazon page. Also take a look at the author's website[4] to see other essays and videos as he continues his work at the intersection of the literary and the digital worlds. I am finding it all quite fascinating. At minimum, I can recommend borrowing *Track Changes* from your library.

Notes

[1] Jennifer Schuessler, "The Muses of Insert, Delete and Execute", 2011-12-26, `tinyurl.com/times-schuessler`

[2] Somewhat irrelevant aside: If you have not read Stephenson's *In the Beginning was the Command Line*, I highly recommend it still, even though it is now quite dated: `http://cryptonomicon.com/beginning.html`

[3] I am wondering about other fiction authors who may have used TEX et al. to do their composing. If you know of any, you might communicate what you know to author Kirschenbaum (see the last sentence of the first paragraph).

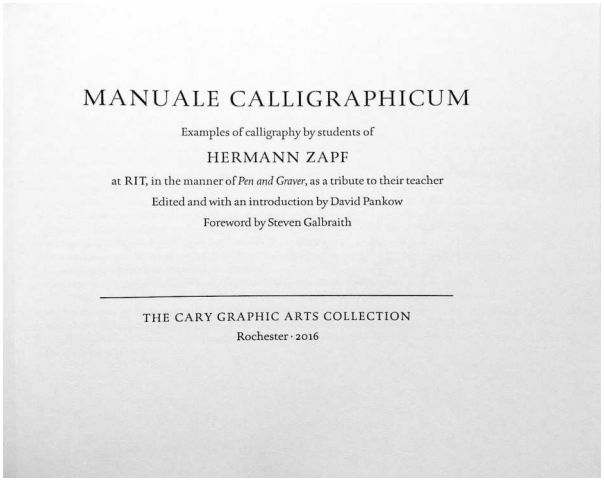
[4] `https://mkirschenbaum.wordpress.com/`

◇ David Walden
`walden-family.com/texland`

Book review: *Manuale Calligraphicum. Examples of Calligraphy by Students of Hermann Zapf*, David Pankow, ed.

Boris Veytsman

Manuale Calligraphicum. Examples of Calligraphy by Students of Hermann Zapf, David Pankow, ed. RIT Cary Graphic Arts Collection; Rochester, 2016, 60pp, ill. Hardcover, paper over boards with vellum spine. US$195.00.



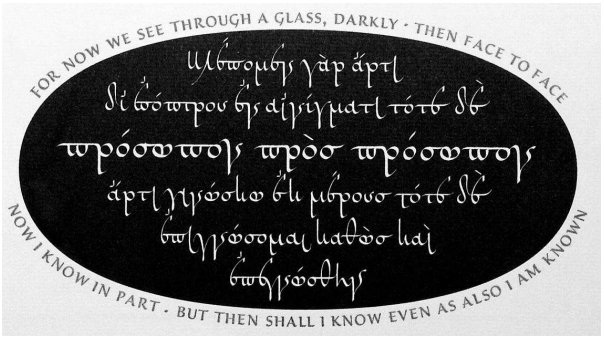


Figure 1: Kris Holmes's contribution

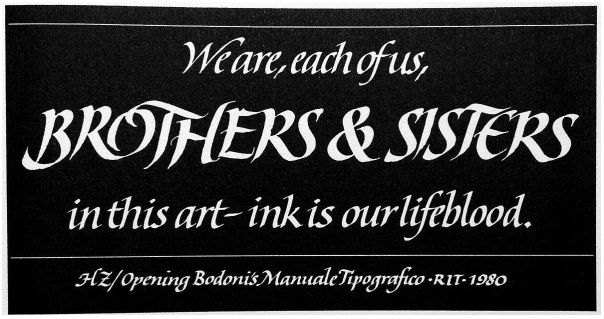


Figure 2: Steven Skaggs's contribution

For ten years, from 1979 to 1988, Hermann Zapf taught summer master classes on calligraphy at the Rochester Institute of Technology. This book is a tribute to him by his former students. It contains nineteen specimens — or, rather, nineteen pieces of calligraphy art by Larry and Marsha Brady, Annie Cicale, Rick Cusick, Claude Dieterich A., Reggie Ezell, Peter Fraterdeus, Kris Holmes, Jerry Kelly, Peter Noth, Marcy Robinson, Ina Saltz, Steven Skaggs, John Stevens and Julian Waters. A TEX user probably knows some of these names from DEK's *3:16 Bible Texts Illuminated*, and, of course, from the Lucida fonts by Bigelow and Holmes. However, I would argue that calligraphy by itself is important for anybody interested in the philosophy behind TEX and METAFONT. One of the origins of our letters is the art of scribes, who for centuries developed the technique of combining legibility and beauty using pen and ink (another being the stone cutters making the monumental inscriptions). A careful study of calligraphy helps open the world of good typesetting — and is enjoyable in itself.

Besides the illustrations the book has an introduction by David Pankow, a foreword by RIT Cary Graphic Arts Collection curator Steven Galbright, and artists' biographical sketches. It was designed by renowned typographer Jerry Kelly. The spartan title page and Kelly's Rilke typeface of the text contribute to the general impression of a tasteful book. It is printed on a letter press by Bradley Hutchinson on Hahnemühle paper with vellum spine in 325 copies.

The book was designed as a luxury item, thus its rather steep price. Still, since the Web version of this review is going to appear before 2016 holiday season, it might give you a gift idea for a calligraphy lover dear to you.

On the other hand, I wonder whether RIT Cary Graphic Arts Collection might consider a trade edition of this book: it might benefit many students of calligraphy and fonts.

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

Seminar review: *Presenting data and information* by Edward Tufte, November 9, 2016, Arlington, VA

Boris Veytsman

Edward Tufte, a prominent advocate of beauty and clarity in data presentation, is rather well known to the TEX community. There are several packages on CTAN inspired by his ideas, for example, *tufte-latex* and *sparklines*. Tufte's essay about the cognitive style of PowerPoint (2003) was quoted by many TEXnicians at that time (I must say, however, that we now have TEX tools quite capable of reproducing PowerPoint style). His web site (`edwardtufte.com`) and Twitter (`@EdwardTufte`) have been sources of many interesting ideas and beautiful pictures.

For quite some time, I have wanted to attend his seminar (`edwardtufte.com/tufte/courses`), and this November my employer (Harris Corp.) kindly agreed to pay for it. (Disclaimer: this review reflects only my personal opinion, and does not necessarily reflect opinions of my employer, other Harris employees or officers. This review cannot be construed as an endorsement or recommendation by Harris or any other corporate entity.)

Despite a rather steep price ($420.00, which includes four books by Tufte in paperback editions, coffee & tea, but does not include parking or lunch), the big ballroom of the Hyatt at DCA airport was packed. This is a good sign: there *is* an interest in high quality data presentation by the public.

The presentation itself, as a matter of course, followed Tufte's recommendations. It started with an hour of silent reading, it emphatically did not use slides with bullets, and it did not assume that the attention span of the audience was close to that of a two year old. Many of the topics covered would be familiar to those who have read Tufte's books: from the famous map by Charles Joseph Minard to the beautiful examples of the "small multiples" graphs. There were also many new topics, including a very interesting analysis of the styles of Web sites. Despite my previous knowledge of Tufte's ideas, the presentation turned out useful for me: it is one thing to read about the general principles, but quite another thing is to see them in action.

Edward Tufte did not limit himself to typography or graphic design. Rather, he showed a holistic approach to the presentation of data and information. He talked about such things as preparation before the presentation, ways to deal with anxiety and technology malfunction, and many other topics that are important for a successful practitioner, but often overlooked.

Tufte discussed both sides of a presentation: the point of view of a presenter, and the point of view of an audience. There was plenty of advice on both how to show information and how to understand what was shown — and recognize when the presenter is not forthright with the truth.

This leads to another dimension of the seminar, one which was rarely mentioned explicitly, but definitely was present: the moral dimension. Tufte explained how to show the story behind the data, always assuming this was a true story. It was not only that he did not teach how to mislead the audience and conceal the truth; rather, his methods are not compatible with such subterfuge (more on this below). Data presentation as practiced by Tufte is not a neutral trade, which can be used for any goal, being it telling the truth or telling lies. Honesty is hardwired in the approach.

The seminar used projections, videos and music. The current jargon suggests the word "multimedia". However, this word now often means some haphazardly selected fragments joined together in a more or less arbitrary fashion. Unlike those "multimedia presentations", this one embedded the non-speech elements with tact and taste, creating an organic whole. Perhaps the closest analogy would be a theater play set by a good director, rehearsed to perfection and skillfully executed. Indeed, Tufte's presentations are one person shows. It is strange that among the arts discussed by the author (typography, painting, music, literature) the thespian art was not mentioned — it would seem to be very relevant to the ideas espoused in the seminar.

While it is impossible to write here about all the topics raised by Tufte, I would like to mention several of them, which, in my opinion, would be of a special interest to the readership of *TUGboat* and the TEX community.

Tufte spent much time advocating high resolution of information. One of his villains, mentioned several times during the course, was the famous "rule of seven ± two", which prescribes no more than about seven chunky sentences on a presentation page. Tufte thinks that audience can grasp complex ideas and flowing texts. He says that the best example of a high resolution information medium is a classic printed page. This strikes a chord with us TEXnicians, brought up in the traditions of classical typography as embodied by TEX design.

Tufte brought with him a couple of rare books to demonstrate these traditions and inherent innovation: one by Galileo (*Istoria e dimostrazioni intorno alle macchie solari*, Rome, 1613) with small images embedded in the text, and a second by Oliver Byrne

(*The Elements of Euclid*, London, 1847) with 3D cutouts of stereometric bodies. For some bibliophiles the opportunity to see these books alone might well justify the cost of admission.

Another important topic of the seminar was providing the audience of a presentation multiple paths through the data. The common low resolution presentation assumes a single path followed by the audience, bullets being stepping stones across the sea of information. The high resolution presentation espoused by Tufte assumes the audience members may choose different paths through the data and reach their own conclusions, perhaps unforeseen by the presenter. The recommended way of starting the presentation with a period of silent reading actively encourages this approach.

There are two aspects to this approach. First, it seems to be surprisingly close to the Free Software philosophy. One of the fundamental purposes of Free Software is to give anyone the ability to modify and combine programs in new ways, sometimes not foreseen by the original authors. Tufte applies this principle to data. Like Free Software users, his audience is expected to be active participants rather than passive consumers fed with pre-digested pieces of information. It is not coincidental that Tufte made highly complimentary remarks about the reproducible research movement, which is an analog of Free Software in the data world.

Second, as mentioned above, this approach is inherently incompatible with misleading and manipulation. A manipulator wants the audience to reach the pre-defined conclusions, and carefully constructs the path to them, avoiding anything which might accidentally reveal the truth. The last thing a manipulator wants is to let the audience engage in an independent analysis. While reproducible research and high resolution presentation were not specifically designed to prevent data manipulation, it is an inherent feature of these approaches. Similarly, while Free Software was not specifically designed to prevent backdoors in software, backdoors are not compatible with it.

It should be said that Free Software was prominent in the recommendations spelled out in the course handouts. Besides TEX, Tufte mentioned *R*, which is used by many TEXnicians. By the way, there are several *R* packages inspired by Tufte: for example, *ggthemes* includes a Tufte theme and Tufte plots such as range frames are part of the *ggplot2* framework. (There is a detailed review of these packages by Lukasz Piwek at `motioninsocial.com/tufte`.)

Still, there was a topic of the seminar which left a mixed aftertaste for me. Namely, Tufte several times exhorted the audience to produce high quality presentations. This was done in a way that seemed to imply that the common low quality presentations are caused by the laziness of the practitioners. This is definitely not the complete case. The high quality display Tufte espouses requires a huge amount of thought. While they are immensely satisfying for the producer and very useful to the audience, they require a most valuable resource: the work of an extremely skilled professional. Therefore they are insanely expensive to make. In many corporate environments, where an incessant stream of (necessarily low quality) presentations is expected, Tufte's ideas are a losing proposition. In a sense, Tufte's exhortations in the Hyatt ballroom could be taken as similar to a motivation speech by a renowned *haute cuisine* chef to an audience of fast food cooks. Of course these cooks would be happy to work on something better than greasy burgers. However, they are not in the position to decide: the menu is determined by their managers, and ultimately by the customers. If we want to change the bleak landscape of corporate presentations, we need to change corporate culture, including the expectations of upper management and individual choices of the information producers and consumers. It seems this would also imply rather fundamental changes in our society. At any rate, shaming the overworked practitioners, in my opinion, would not help.

Nonetheless, there is a role for us TEXnicians, especially those who are in the business of writing packages. Obviously our tools cannot do the creative thinking for the presentation producer. However, there is much mundane work, which we can and should make much easier, leaving the practitioner free to be creative. Many common tools today subtly (or not so subtly) nudge the users in the direction of making poor presentations. It is possible (albeit difficult) to make tools nudging the users to make better ones.

A seminar like this one could be an inspiration and a source of ideas for this effort. Thus I would recommend it for TUG members.

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

Die TEXnische Komödie 4/2016–1/2017

Die TEXnische Komödie is the journal of DANTE e.V., the German-language TEX user group (`dante.de`). (Non-technical items are omitted.)

Die TEXnische Komödie 4/2016

Rainer-M. Fritsch, Koch- und Backrezepte sammeln mit KOMA-Script [Collecting cooking recipes with KOMA-Script]; pp. 42–49

A common problem: At some point one has collected a couple of dozen cooking recipes on paper and amended them with written notes. But when one wants to make the recipe, it does not match the present cooking routine, or the recipe is wrong.

Rainer-M. Fritsch, TextMate 2 – LATEXing unter MacOS [TextMate 2 — LATEXing under MacOS]; pp. 50–56

There is hardly a more emotional topic than the choice of the "right" editor. Some rely on Emacs or Vim, others use more TEX-centric editors like Kile, TEXmaker, TeXShop, etc.

Norbert Preining, Build- und Testing Infrastruktur für TEX Live [Build- and testing infrastructure for TEX Live]; pp. 57–61

Every year European and non-European TEX enthusiasts meet in Bachotek, Poland, for BachoTEX. The friendly atmosphere and reduced options to escape allow for intensive discussions and progress in developing new TEX features, as well as reducing the beer cellars. This year I was able to attend for the first time since my migration to Japan and used the time to discuss proposals with other team members (Mojca Miklavec und Arthur Reutenauer) for a TEX Live testing infrastructure.

Rolf Niepraschk, Schreibmaschinenschriften – eine Betrachtung [Typewriter fonts — a review]; pp. 62–66

Within the last few years the supply of high-quality fonts for LATEX has increased. Partly this is due to new TEX compilers like LuaTEX and XETEX, and partly due to many new or updated fonts released under free licenses.

Die TEXnische Komödie 1/2017

Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski, GUST e-foundry font projects; pp. 13–52

[Published in *TUGboat* 37:3.]

Philipp Pilhofer, Der Satz kritischer Editionen mit LATEX und `reledmac` [Typesetting critical editions with LATEX and `reledmac`]; pp. 53–66

Preparing the final ready-to-print version of a critical edition is a complex task. This article presents a way to meet the highest requirements by using LATEX with the packages `reledmac` and `reledpar`. This approach offers many possibilities regarding a flexible workflow. The open file formats easily allow further processing, for example a digital edition.

Uwe Ziegenhagen, Wie man einen eigenen TEX Live-Mirror aufsetzt [How to setup your own TEX Live mirror server]; pp. 67–71

In this article I show how NAS (network attached storage) can be used to set up a private TEX Live mirror server.

Christine Römer, Strukturbäume mit Ti*k*Z [Structure trees with Ti*k*Z]; pp. 72–78

A supplement to my article on the `forest` package.

Herbert Voss, Im Netz gefunden [Found on the net]; pp. 79–81

On various mailing lists, web portals and newsgroups one always finds helpful tips and tricks around (LA)TEX. This time:

- The value ranges of `\escapechar`, `\newlinechar`, `\endlinechar` in various TEX engines
- Creating a reference on an equation inside `\lstinputlisting`

[Received from Herbert Voß.]

Zpravodaj 2015/3–4, 2016/1–4

Editor's note: *Zpravodaj* is the journal of CSTUG, the TEX user group oriented mainly but not entirely to the Czech and Slovak languages (`cstug.cz`).

Zpravodaj 2015/3–4

Petr Sojka, Úvodník [Editorial]; p. 97

Three papers in this issue are introduced to the kind readers.

Martin Pecina, Knížky jsou mánie jako každá jiná [Books are an obsession like any other]; pp. 98–103

The author describes his approach to prepare the typographic design of a book. He focuses on the topics of choosing a typeface and size, and the overall design. 10.5300/2015-3-4/98

Antonín Jeřábek, ISBN a online publikace [ISBN and online publications]; pp. 104–109

The paper briefly reviews the history of ISBNs, from the initial purpose of assigning ISBNs to printed publications, describing the development of this number from 8-digit through 9- and 10-digit up to the presently used 13-digit number. The new environment of the ISBN system, connected with the revival of electronic book publication — offline and online — is described, followed by the three conditions accepted within the ISBN community which an e-book must fulfill to get an ISBN. World-wide problems caused by assigning ISBNs to online publications are mentioned, too. In conclusion, the paper gives brief guidelines for Czech publishers for assigning ISBNs to online publications. 10.5300/2015-3-4/104

Luigi Scarso, Dvě užití SWIGLIB: GraphicsMagick a Ghostscript [Two applications of SWIGLIB: GraphicsMagick and Ghostscript]; pp. 110–119

[Printed in *TUGboat* 36:3.] 10.5300/2015-3-4/110

Zpravodaj 2016/1–4

Petr Sojka, Úvodník staronového předsedy [An introductory word by the once and future president]; pp. 1–6

This editorial discusses CSTUG's past and future, membership issues and shaping the organization in the Internet era, together with changes related to the publishing of Zpravodaj CSTUG.

Go forth and participate in CSTUG to make the bright future of TEX & Friends a reality! *You can!*

Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski, Fontové projekty e-písmolijny GUST [GUST e-foundry font projects]; pp. 7–46

[Printed in *TUGboat* 37:3.] 10.5300/2016-1-4/7

Luigi Scarso, Projekt SWIGLIB [The SWIGLIB project]; pp. 47–61

[Printed in *TUGboat* 36:1.] 10.5300/2016-1-4/47

Marek Pomp, Dobře dokumentované statistické výpočty [Well-documented statistical calculations]; pp. 62–77

The paper describes the usage of the literate programming paradigm in the environment of statistical computations. In particular, the package `Sweave` for writing documents with the assistance of the statistical program R is presented. 10.5300/2016-1-4/62

Vít Novotný, Sazba textu označkovaného v jazyce Markdown uvnitř TEXových dokumentů [Rendering Markdown inside TEX documents]; pp. 78–93

The article describes a new package for plain TEX derivatives that enables the direct inclusion of Markdown-formatted text into TEX documents. The author describes their motivation for the creation of the package and its inner workings. The usage of the package is explained through examples. 10.5300/2016-1-4/78

Michal Hoftich, Elektronické knihy a systém TEX4ebook [E-books and the TEX4ebook system]; pp. 94–105

This article describes the process of conversion of a TEX document to an e-book using the TEX4ebook system. Concrete examples of configuration and caveats are provided. 10.5300/2016-1-4/94

Dávid Lupták, Sadzba bibliografie podľa normy ISO 690 v systéme LATEX [Typesetting bibliographies compliant with the International Standard ISO 690 in LATEX]; pp. 106–120

The preparation of bibliographic references and citations compliant with the international standard ISO 690 is required by many institutes, not limited to Czech and Slovak academia. However, the typesetting of bibliographies conforming to the respective standard is not yet supported in the LATEX document preparation system. The biblatex-iso690 package has been revised and improved to fully meet the requirements of the international standard and thus greatly simplifies the typesetting of bibliographies for all kinds of information resources. 10.5300/2016-1-4/106

Peter Wilson, Mělo by to fungovat V – Cykly [It might work. V — loops]; pp. 121–127

This paper shows how to process strings, character by character, in LATEX. The paper also describes the LATEX macro `\@for` and shows its application for typesetting tables. 10.5300/2016-1-4/121

[Received from Michal Růžička.]

***Eutypon* 36–37, October 2016**

Eutypon is the journal of the Greek TEX Friends (`http://www.eutypon.gr`).

Linus Romer, The evolution of the Miama typeface; pp. 1–6

At the beginning of 2016, the free typeface Miama Nueva was released along with a corresponding LATEX package. This article describes how the typeface has evolved over the years. (*Article in English.*)

Giorgos Katsamakis, Eighteenth century utopias in Nikolaos Glykys' dropcaps; pp. 7–14

The dropcaps in books printed in Venice in the 18th century, at the press of Nikolaos Glykys "of Ioannina", are images of a world that never existed, but we can find some, 250 years later. (*Article in Greek with English abstract.*)

George D. Matthiopoulos, Adrian Frutiger: Reshaping the post-war æsthetics of typographical art; pp. 15–26

At a time when the photomechanical production of fonts and texts started exceeding the physical presence of metal type, Adrian Frutiger (1928–2015) was the first typographic designer to face decisively and creatively the new challenges, and it was he who actually shaped the æsthetic rules and technical implementation of phototypesetting. (*Article in Greek with English abstract.*)

Apostolos Syropoulos, The digitization of the Frederika font; pp. 27–31

The Frederika typeface was designed by the famous German designer Hermann Zapf around 1953, as the Greek complement of his own Latin Virtuosa font. Until recently, Frederika existed only in metal type. This article presents an overview of the digitization of Frederika, and some basic characteristics of the new digital font. (*Article in Greek with English abstract.*)

Pablo García-Risueño, Apostolos Syropoulos and Natàlia Vergés, On new ideograms for Physics and Chemistry; pp. 33–39

The svrsymbols package and its accompanying font provide access to new symbols (or ideograms) for use in texts in Physics and Chemistry. The rationale for the creation of the package is presented and discussed. Also, the development model employed is briefly described. (*Article in English.*)

Dimitrios Filippou, TEXniques: (a) Vignettes, dropcaps and other decorative designs; (b) PDF files for ... filing; pp. 41–44

This regular column shows (a) how to produce scalable images for book decoration, and (b) how to prepare PDF/A files with XELATEX and pdfx. (*Article in Greek.*)

Dimitrios Filippou, Book presentations; pp. 45–46

The following books are presented: (a) Jean-luc Doumont, *Trees, Maps, and Theorems*, Principiæ, Kraainem, Belgium 2009; and (b) George Grätzer, *Practical LATEX*, Springer, Cham, Switzerland 2014. (*Article in Greek.*)

[Received from Dimitrios Filippou and Apostolos Syropoulos.]

TUG financial statements for 2016

Klaus Höppner, TUG treasurer

The financial statements for 2016 have been reviewed by the TUG board but have not been audited. As a US tax-exempt organization, TUG's annual information returns are publicly available on our web site: `http://tug.org/tax-exempt`.

Revenue (income) highlights

Membership dues revenue was down about $6,000 in 2016 compared to 2015. Product sales and other income categories were close to steady, with two exceptions: The annual conference produced a small loss, while Contributions Income was up $2,000, mainly due to a generous anonymous donation. Overall, 2016 income was down 7%.

Cost of Goods Sold and Expenses highlights, and the bottom line

The TUG board felt it necessary to seek legal advice regarding the possible appeal by the suspended president, incurring extraordinary expenses of about $14,000, marked as Professional Fees.

While we saved some money on DVD production, *TUGboat* costs were up $7,000, partly due to many color pages in the 2016 conference proceedings issue.

The bottom line for 2016 was very negative: about $21,100. Even disregarding the exceptional Professional Expenses the result was a loss of $7,200.

Balance sheet highlights

TUG's end-of-year asset total is down by around $11,200 (5.5%) in 2016 compared to 2015.

Committed Funds are reserved for designated projects: LaTeX, CTAN, the TeX development fund, and others (`http://tug.org/donate`). Incoming donations are allocated accordingly and disbursed as the projects progress. TUG charges no overhead for administering these funds.

The Prepaid Member Income category is member dues that were paid in earlier years for the current year (and beyond). The 2016 portion of this liability was converted into regular Membership Dues in January of 2016. The payroll liabilities are for 2016 state and federal taxes due January 15, 2017.

Summary

We ended 2016 with 67 less members than in 2015. This was an important cause for the bottom-line loss, and the board continues to work to reverse this trend. Ideas for attracting members are always welcome!

TUG 12/31/2016 (vs. 2015) Revenue, Expense

	Dec 31, 16	Dec 31, 15
ORDINARY INCOME/EXPENSE		
Income		
Membership Dues	86,460	92,550
Product Sales	5,801	5,736
Contributions Income	10,681	8,320
Annual Conference	(699)	1,837
Interest Income	575	484
Advertising Income	315	320
Services Income	1,176	2,616
Total Income	104,309	111,863
Cost of Goods Sold		
Membership Drive		
TUGboat Prod/Mailing	(24,896)	(17,722)
Software Prod/Mailing	(2,479)	(3,200)
Postage/Delivery - Members	(1,356)	(2,147)
Lucida Sales to B&H	(2,263)	(2,195)
Member Renewal	(384)	(412)
Total COGS	(31,378)	(25,676)
Gross Profit	72,931	86,187
Expense		
Contributions made by TUG	(2,000)	(2,000)
Office Overhead	(14,934)	(15,444)
Payroll Expense	(63,167)	(63,256)
Professional Fees	(13,878)	
Interest Expense	(50)	
Total Expense	(94,029)	(80,700)
Net Ordinary Income	(21,098)	5,487
OTHER INCOME/EXPENSE		
Prior year adjust	(1)	(95)
Net Other Income	(1)	(95)
NET INCOME	(21,099)	5,392

TUG 12/31/2016 (vs. 2015) Balance Sheet

	Dec 31, 16	Dec 31, 15
ASSETS		
Current Assets		
Total Checking/Savings	193,913	205,581
Accounts Receivable	715	300
Total Current Assets	194,628	205,881
LIABILITIES & EQUITY		
Current Liabilities		
Committed Funds	35,842	31,248
Administrative Services	4,017	1,528
Deferred Contributions		
Prepaid Member Income	6,850	4,085
Payroll Liabilities	1,083	1,087
Total Current Liabilities	47,792	37,948
Equity		
Unrestricted	167,934	162,543
Net Income	(21,098)	5,391
Total Equity	146,836	167,934
TOTAL LIABILITIES & EQUITY	194,628	205,882

TUG Business

TUG 2017 election

Nominations for TUG President and the Board of Directors in 2017 have been received and validated.

For President, Boris Veytsman was nominated. As there were no other nominees, he is duly elected and will serve for a two-year term.

For the Board of Directors, the following 14 individuals were nominated:

Karl Berry, Johannes Braams, Kaja Christiansen, Enrico Gregorio, Taco Hoekwater, Klaus Höppner, Frank Mittelbach, Ross Moore, Steve Peter, Arthur Reutenauer, Will Robertson, Herbert Schulz, Michael Sofka, Herbert Voß.

There were more Board nominations than open positions. Thus, an election ballot is required, in accordance with the TUG election procedures (`http://tug.org/elecproc.html`). This year, voting online is strongly recommended, through the TUG members area, `https://www.tug.org/members`. (A printable ballot is also available there, or on request from the TUG office, if you wish to vote on paper.)

Terms for both President and members of the Board of Directors will begin at the Annual Meeting.

Board members Steve Grathwohl, Jim Hefferon, and Geoffrey Poore have decided to step down at the end of this term. On behalf of the Board, I wish to thank them for their service, and for their continued participation until the Annual Meeting.

Statements for all the candidates are appended, both for President and for the Board (order determined by lot). They are also available online, along with announcements and results of previous elections.

The deadline for voting in the election is April 9, which is still some time away as this issue goes to press. The results will be announced on the TUG web site and in the electronic newsletter, as well as the next issue of *TUGboat*.

◇ Barbara Beeton
for the Elections Committee
`http://tug.org/election`

Boris Veytsman

(Candidate for TUG President.)

I was born in 1964 in Odessa, Ukraine and have a degree in Theoretical Physics. I worked for various sci-tech employers, including TEX consulting for Google and contractor work with NASA. At present my main work is in aviation safety with Harris Corporation. I teach and do research at George Mason University. I also do TEX consulting for a number of customers, including FAO UN, US Government agencies, universities and publishers worldwide. My CV is at `http://borisv.lk.net/cv/cv.html`.

I have been using TEX since 1994 and have been a TEX consultant since 2005. I have published a number of packages on CTAN and papers in TUGboat. I have been a Board member since 2010 and Vice-President since 2016. I am an Associate Editor of TUGboat and support `http://tug.org/books/`.

Over the years I have done some thinking about the future of TEX Users Group. In the old days if you wanted tapes with TEX distributions or sought help, you joined TUG. Now things have changed: you easily can download any distribution or ask questions. Thus the steady decline in our membership. While TEX is used by many people, TUG is shrinking.

If we want to remain relevant, we need to consider our role in the new user community. I think there are several areas where TUG is needed:

1. A point of contact for the community. The most important things we are doing are our conferences and TUGboat. We need to continue these and promote TUGboat for libraries. We need to promote our Web site and make it more useful for our community.
2. Coordination and steering of technical, education and outreach efforts. We have several working groups (see `http://tug.org/twg`). I am trying to jump-start Education and Accessibility work groups.
3. Help with financing TEX-related efforts. Being a tax-exempt (in the US) 501(c)(3) organization, we are in an exceptionally good position to collect funds for various projects.

We should remember that we serve not just our members, but the wide community of TEX users. We should actively promote TUG among the community and make easier the option to support TUG by volunteering or donating.

I think we need to work to be relevant in the changing world. If the TUG members trust me to lead this effort, I would be greatly honored.

Karl Berry

(Candidate for TUG Board of Directors.)

TEX biography: I served as TUG president from 2003–2011 and was a board member for two terms prior to that, and also subsequently. I am running again for a position on the board.

I co-sponsored the creation of the TEX Development Fund in 2002. I'm one of the primary system administrators and webmasters for the TUG servers, and the production manager for our journal *TUGboat*.

On the development side, I'm currently the editor of TEX Live, the largest free software TEX distribution, and thus coordinate with many other TEX projects around the world, such as CTAN, LATEX, and pdfTEX. I developed and still (co-)maintain Web2c (Unix TEX) and its basic library Kpathsea, Eplain (a macro package extending plain TEX), and other projects. I am also a co-author of *TEX for the Impatient*, an early comprehensive book on plain TEX, now freely available. I first encountered and installed TEX in 1982, as a college undergraduate.

Statement of intent: I believe TUG can best serve its members and the general TEX community by working in partnership with the other TEX user groups worldwide, and sponsoring projects and conferences that will increase interest in and use of TEX. I've been fortunate to be able to work pro bono on TUG and TEX activities the past several years, and plan to continue doing so if re-elected.

Johannes Braams

(Candidate for TUG Board of Directors.)

Biography: I encountered TEX and friends sometime around 1985 when it was installed on our research VAX. It didn't take long for me to get hooked on LATEX and I started to think about and work on multilingual support, later to be known as *babel*. Besides that I have been active on quite a number of activities:

- For TEX and the TEX User Group
  - Co-founder and board member of the NTG, the Dutch-speaking TEX User group
  - As chairman of NTG I have served on the board of directors in 1994/1995 as special director for the NTG
  - designer of a couple of PhD-theses
  - Original author of the *babel* language support system
  - Member of the LATEX3 team, author of one of the chapters in the *LATEX Companion*
  - author, co-author or maintainer of a couple of publicly available LATEX packages and classes
- Professionally
  - System administrator for VAX/VMS and Unix systems
  - Manager of a team of System administrators
  - Functional administrator of one of the large administrative systems of KPN (PTT Telecom back then)
  - Project manager for various IT-projects within KPN for about 18 years
  - Consultant in the area of cyber security in Industrial Control Systems since 2015

Statement: In the last couple of years I have been coming back into the TEX-community. I would very much like to serve this wonderful community by taking up a role in the board of directors of TUG. I think TEX should and will be alive and well for many years to come, as the quality of typesetting that can be achieved by using TEX is still unsurpassed.

Kaja Christiansen

(Candidate for TUG Board of Directors.)

I was born in Warszawa, Poland and live in the city of Aarhus, Denmark. I heard about TEX for the first time in the fall of 1979. In Palo Alto at the time, I wanted to audit courses at Stanford and my top priority was lectures by Prof. Donald Knuth. That, I was told, was not possible as Prof. Knuth was on leave due to work on a text processing project... This project was TEX! Back home, it didn't take long till we had a runnable TEX system in Denmark.

I have served as a Board member since 1997, co-sponsored the creation of the TEX Development

Fund and have been the chair of TUG's Technical Council since 1999. I am also a member of the Bursary and Election committees and served as TUG vice-president from 2003–2011. I share system administrator's responsibilities for the TUG server and TUG's web site, and actively contributed to several earlier versions of TeXlive. Finally, I am a board member of the Danish TeX Users Group (DK-TUG) and served as the president of DK-TUG in 2002–2011.

Statement: TeX and friends are the only software I know of that, after 30+ years, is not only alive and well, but also the best typesetting system to produce beautiful books and papers. In my rôle as a member of the board, my special interests have been projects of immediate value to the TeX community, among them system administration, TeX Live and *TUGboat*.

Enrico Gregorio

(Candidate for TUG Board of Directors.)

Biography: I am associate professor of Algebra since 1992, currently at the University of Verona (Italy).

I'm active in TeX related matters since I started loving it, which was around 1986. I'm the author of a LaTeX programming book in Italian and a member of GuIT, the Italian TeX users group, which I also served as president and board member.

Since 1996 I deliver a LaTeX course at my Department, which is academically recognized and grants two credits to students who follow it and get their assignments approved.

I am quite active at the tex.stackexchange.com site, being the highest reputation member, with the nickname `egreg` and also in the GuIT forum as `egreg9`.

I have authored some LaTeX packages and papers for *TUGboat* and ArsTeXnica.

Personal statement: My main interest for the TeX world is in developing *good* documentation and in promoting good style in document writing and programming.

Supporting LaTeX3 development should be one of the important tasks for TUG in the next years.

Taco Hoekwater

(Candidate for TUG Board of Directors.)

TeX biography: Taco Hoekwater (born in 1969 in Netherlands) has been a ConTeXt user for two decades. He has been the first user of ConTeXt outside of PRAGMA ADE and works in tight cooperation with Hans Hagen to develop TeX engines and macros for ConTeXt ever since. He has been the president of the Dutch language-oriented TeX users group (NTG) from 2009 until 2015 and was the main editor of the NTG's magazine MAPS. He has been the maintainer of MetaPost for a number of years and a core participant in the LuaTeX development team. He organised the first international ConTeXt User Meeting in Epen, Netherlands in 2007, and various other TeX- and ConTeXt-related conferences since. He was previously on the TUG board from 2011 to 2015, and is the current president of the ConTeXt user group (CG).

Statement of intent: As a board member, I hope to be able to promote future extensions and applications of Knuth's amazing piece of software, as well as raise awareness of ConTeXt outside its current user base.

Klaus Höppner

(Candidate for TUG Board of Directors.)

Biography: I got a PhD in Physics in 1997. After several years in the control systems group of an accelerator center in Darmstadt, I've been working at an accelerator for cancer therapy in Heidelberg. My first contact to LaTeX was in 1991, using it frequently since then.

I have been preparing the CTAN snapshot on CD, distributed to the members of many user groups, from 1999 until 2002. I was the local organizer of TUG2015 and was heavily involved in the organization of several DANTE conferences and EuroTeX 2005. I've been a member of the TUG board since

2005 and was a member of the DANTE board until 2016, including terms acting as president, vice president, and treasurer.

Statement: As in the past, I want to be the voice of European users, in particular those who need characters with funny accents. TUG's last year was troublesome. The board had to make a hard decision. It's up to you to decide whether it was right or wrong. Anyway, we have to look forward, coming back to a stable and constructive cooperation. TUG has an efficient office and a well-working board, against all blames. Looking at other user groups, that's not too common.

Renovation was an often mentioned goal regarding TUG. Renovation is fine, it just has to be performed by working with the developers of TEX distributions and maintainers of TEX repositories and web sites, not by blaming them for the work they do in their spare time. I've seen many improvements in the past, regarding the installation process of TEX distributions (e.g. TEX Live) and in the user interface of CTAN. The rather plain and simple layout of TUG's web pages may be improved, of course giving the usability, especially for handicapped, the precedence over fanciness. However, the key information is there. The TEX community needs volunteers instead of hot-air merchants. And the TUG board needs constructive directors. I think the TUG conference and AGM in Toronto created the base for a new start that shouldn't be spoiled.

Frank Mittelbach

(Candidate for TUG Board of Directors.)

I came in contact with TEX in the mid-eighties and over the years TEX, LATEX and typography in general became a very important part of my life. In 1990 I took over the maintenance and further development of LATEX from Leslie Lamport and together with a small number of people (most notably David Carlisle, Chris Rowley and Rainer Schöpf at that time) we designed and implemented what became LATEX2e in 1994 — the LATEX you still essentially use today (even though it has undergone smaller modifications and improvements through by now 25 further releases).

Despite all predictions made during the last decades, TEX and LATEX are alive and kicking as proven by their (still?) strong use in various ways around the world.

Nevertheless the world has changed and is changing further and in that changing world user groups like TUG need to find their place and possibly reinvent themselves by redefining and reshaping their role. With my work on the TUG board I would like to help in that process and ensure a future for high quality typography as provided by TEX.

Ross Moore

(Candidate for TUG Board of Directors.)

I would like to offer my services to the TUG Board. Having been active in the TEX community for more than 20 years, writing code for packages, my first TUG meeting was in 1997. Previously I've been a board member, but have taken a rest these past 2 years. In January, Boris asked if I'd be willing to return; I was happy to say 'yes'.

Well, 2017 is likely to be a very important year for the PDF format, hence for TEX, and for TUG and other user groups around the world. Already on 18 January, a document "Information and Communication Technology (ICT) Final Standards and Guidelines" was published into the US Federal Register, effectively becoming law. This affects "standards for electronic and information technology developed, procured, maintained, or used by Federal agencies covered by section 508 of the Rehabilitation Act of 1973, as well as guidelines for telecommunications equipment [...]. These revisions [...] are intended to ensure that information and communication technology covered by the respective statutes is accessible to and usable by individuals with disabilities." It specifies that "authoring tools capable of exporting PDF files must conform to PDF 1.7 [...] and be capable of exporting PDF files that conform to PDF/UA-1."

While documents produced by TEX-based software are compatible with PDF 1.7, it is certainly not the case that they conform to PDF/UA, which requires producing 'Tagged PDF'. Thus if LATEX or other TEX-based software can be used within US government agencies only as part of a processing chain requiring other software to complete the final document. This is not how we normally work with TEX. Furthermore, the PDF 2.0 standard, also based upon 'Tagged PDF', may appear as early as June.

Over the past 8 years (or more), I have given talks at annual TUG meetings, demonstrating PDF

features that tagging allows, produced using an extended version of pdf-TEX. Currently I'm working on a set of LATEX macros that produce valid 'Tagged PDF' documents satisfying the WCAG 2.0 Accessibility standard. Examples, using different LATEX document classes and built with the current pdfTEX, can be found on my web page `http://maths.mq.edu.au/~ross/TaggedPDF/`. These developments need to be enhanced to become *de rigeur* for the way we use TEX and LATEX, else there will be no new TEX users in years to come. As a Director of TUG, this outlines an agenda that I'll be supporting.

Steve Peter

(Candidate for TUG Board of Directors.)

Biography: I am a linguist and publisher originally from Illinois, but now living in New Jersey. I first encountered TEX as a technical writer documenting Mathematica. Now I use TEX and friends for a majority of my publishing work, and work with several publishers customizing TEX-based publishing systems. I am especially interested in multilingual typography and finding a sane way to typeset all of those crazy symbolisms linguists create. As if that weren't bad enough, I also design typefaces. (Do I know lucrative markets, or what?)

I got involved in TUG via translations for *TUGboat*. I was on the TUG board of directors for several terms before becoming TUG president in 2011, serving two terms, after which I returned to the board.

Statement: The future of TEX and TUG lies in global communication and cooperation to promote and sustain the amazing typographic quality associated with TEX and friends. Projects such as LuaTEX show that there remains a dynamic and bright future for our preferred typesetting system. I am especially interested in having TUG support projects (technical and artistic) that will serve to bolster TEX and TUG's visibility in the world at large.

Arthur Reutenauer

(Candidate for TUG Board of Directors.)

Biography: I first joined TUG in 2005 and have been a member of the board for the past four years. My interest in TEX started during my university years when, being a student of mathematics and physics, I had to use it for typesetting reports. I was prompted to dive deeper because of my interest in languages and writing systems, and soon got passionate about it (see my entry in the TUG interview corner for excruciating details).

Statement: Today, I am active in TEX development as the maintainer of various packages and programs related to multilingual typesetting (polyglossia, hyph-utf8, XETEX), and I regularly give talks at conferences and write articles in journals. I have been on the board of several TEX users groups, founded the ConTEXt group, and have contributed to promoting TEX which in my opinion is, still today, one of the best typesetting systems available. I am truly amazed at how vibrant our user community is.

If I am reelected to the TUG board, I will do my best to represent that community in all its diversity. Our organisation has been through some hard times recently, and I want to help moving forward and foster the use of TEX and Metafont in the 21st century.

Will Robertson

(Candidate for TUG Board of Directors.)

My background is mechanical engineering, and having (finally) completed my PhD in magnetic levitation and vibration control in 2013, I am now a full-time lecturer at The University of Adelaide. My interest in LATEX was preceded by an interest in computer typesetting, and I came into LATEX in the midst of a general transition to Unicode. Early in XETEX's life I wrote some LATEX code to aid the font-loading process, and from there I became hooked and developed a number of LATEX packages, including fontspec and unicode-math. After some time I also became a member of the LATEX3 project, helping to develop the expl3 programming language.

Although my day-to-day programming contributions have diminished in recent years as my personal and professional responsibilities have increased, my passion and appreciation for TEX in all its forms only increases with the years. Personally, I see TEX as the only typesetting system worth using for most forms of document production, although I'm happy for people to hold their own opinions on the matter. I teach LATEX to our entire mechanical engineering honours project student cohort, and every year I

have overwhelmingly positive feedback on both its style and utility over what they're used to.

As a member of the TUG Board of Directors, I would seek greater recognition for TEX from the world's universities and technical companies. Recognition by way of TUG membership, of course, to allow the TUG organisation to flourish. In addition, I would seek initiatives for younger users and developers to engage with TUG, with an emphasis on open discussion with the number of companies that now sell access to their online TEX editors and document previewers.

I believe that my immersion in the current technical landscape of TEX and friends gives me a view of the TEX world that is both broad and deep. I would not sit on the TUG Board as a LATEX3 developer, but rather as one who has a great wish to see success across the whole TEX world, from ConTEXt in its technical mastery, to eplain in its hackability, to LATEX in its rich ecosystem. TEX has a bright future, and it would be an honour to contribute to that future as a member of the TUG Board.

Herbert Schulz

(Candidate for TUG Board of Directors.)

I've been using TEX since the mid-1980s; well, except for several years in the mid- to late-1990s. During that time period I received a great deal of help from others as I tried to better understand the subtleties of using TEX, and later LATEX. After retiring in 2001 I decided to give back to the community and have helped put together MacTEX as well as MacTEXtras as well as help others on multiple (both Mac and non-Mac centric) on-line forums and email lists.

I would like to become a member of the TUG Board to further help develop on-line help through the TUG Web Pages by gathering links to as many on-line and up-to-date TEX, LATEX and Distribution help locations along with descriptions of what they provide. I believe this would be one way to help increase TUG membership and speed up the learning curve for new users of TEX.

Michael Sofka

(Candidate for TUG Board of Directors.)

Biography: I have been a TEX user for nearly 30 years, and a member of TUG for 28 years, including a term on the Board (2001–2005). My use of TEX started as a programmer for a full service typesetting company writing macro packages for books, device drivers for various laser typesetters. I continued using TEX and LATEX in my work and personal life after switching to IT in Academia. During this time TEX and LATEX have changed tremendously, and I am continually amazed by the creativity of the TEX community. It is no exaggeration to say my continued interest and membership in TUG derives from the delight of discovering new tools and techniques for traditional and electronic publishing.

Statement: TEX and TUG have come a long way from the days of half-inch tape distribution, CWEB, and the Berkeley Pascal compiler. The typesetting engines available have expanded and grown, and become far easier to install. The ways in which people find information about TEX has also changed. When I joined TUG I was desperate for quality information. Through *TUGboat* articles, and many readings of *The TEXbook*, I learned how to program and write using TEX. Today, the World Wide Web, through sites such as StackExchange, has become the first source of information. Most people who use TEX and LATEX might not even be aware of the TEX Users Group, and the role it plays in funding and promoting the programs and packages they depend upon. Going forward, TUG will grow and adapt to this new way of sharing information, and explore new methods of attracting TEX users, whether it be through Web pages, sponsoring development, annual meetings, or local conferences. I hope you will support me in being a part of that process.

Herbert Voß

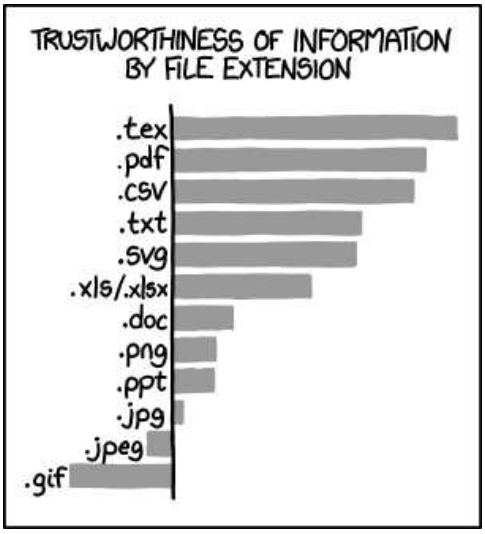
(Candidate for TUG Board of Directors.)

I am retired from my job as a teacher for physics, mathematics and computer science at a so-called Gymnasium (higher education school), led by the Jesuit order. For more than ten years I am also a lecturer at the Free University of Berlin. I started with one course per year in scientific writing with LaTeX and have now eight courses per year.

I like reading and writing and created together with the publisher Lehmanns Media the so-called DANTE Edition, (La)TeX books from members for members of DANTE, the German speaking TeX user group. For DANTE's 25th anniversary I created a new edition of Victor Eijkhout's book *TeX by Topic* which was also published in the DANTE series.

I am very interested in creating an ebook from a LaTeX source, which is, of course, still not really easy. And my other major interest is "LaTeX for beginners": how can one minimize the problems of the installation of a TeX distribution and the troubleshooting when running (La)TeX on source documents.

Statement: I still believe that TeX has a future. With LuaTeX we have a great potential to keep TeX running and I think that TUG can play a major role to make this public to the rest of the world.



Comic by Randall Munroe (`https://xkcd.com`), licensed under CC BY-NC.

TEX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at `tug.org/consultants.html`. If you'd like to be listed, please see there.

Aicart Martinez, Mercè
Tarragona 102 4^o 2^a
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) ono.com`
Web: `http://www.edilatex.com`
We provide, at reasonable low cost, LaTeX or TeX page layout and typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990. For more information visit our web site.

Dangerous Curve
+1 213-617-8483
Email: `typesetting (at) dangerouscurve.org`
We are your macro specialists for TeX or LaTeX fine typography specs beyond those of the average LaTeX macro package. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical TeX and LaTeX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a TeX book.

de Bari, Onofrio and Dominici, Massimiliano
Email: `info (at) typotexnica.it`
Web: `http://www.typotexnica.it`
Our skills: layout of books, journals, articles; creation of LaTeX classes and packages; graphic design; conversion between different formats of documents.

We offer our services (related to publishing in Mathematics, Physics and Humanities) for documents in Italian, English, or French. Let us know the work plan and details; we will find a customized solution. Please check our website and/or send us email for further details.

Latchman, David
2005 Eye St. Suite #4 Bakersfield, CA 93301
+1 518-951-8786
Email: `david.latchman (at) texnical-designs.com`
Web: `http://www.texnical-designs.com`
LaTeX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized packages to meet your needs. Call or email to discuss your project or visit my website for further details.

Peter, Steve
+1 732 306-6309
Email: `speter (at) mac.com`
Specializing in foreign language, multilingual, linguistic, and technical typesetting using most flavors of TeX, I have typeset books for Pragmatic Programmers, Oxford University Press, Routledge, and Kluwer, among others, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. In addition, I've helped publishers write, maintain, and streamline TeX-based publishing systems. I have an MA in Linguistics from Harvard University and live in the New York metro area.

Sofka, Michael
8 Providence St.
Albany, NY 12203
+1 518 331-3457
Email: `michael.sofka (at) gmail.com`
Skilled, personalized TeX and LaTeX consulting and programming services.

I offer over 25 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in TeX and LaTeX: Automated document conversion; Programming in Perl, C, C++ and other languages; Writing and customizing macro packages in TeX or LaTeX; Generating custom output in PDF, HTML and XML; Data format conversion; Databases.

If you have a specialized TeX or LaTeX need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

Veytsman, Boris
46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`
TeX and LaTeX consulting, training and seminars. Integration with databases, automated document preparation, custom LaTeX packages, conversions and much more. I have about two decades of experience in TeX and three decades of experience in teaching & training. I have authored several packages on CTAN, Perl packages on CPAN, R packages on CRAN, published papers in TeX related journals, and conducted several workshops on TeX and related subjects.

Webley, Jonathan
2/4 31 St Andrews St
Glasgow, G1 5PB, UK
07914344479
Email: `jonathan.webley (at) gmail.com`
I'm a proofreader, copy-editor, and LaTeX typesetter. I specialize in math, physics, and IT. However, I'm comfortable with most other science, engineering and technical material and I'm willing to undertake most LaTeX work. I'm good with equations and tricky tables, and converting a Word document to LaTeX. I've done hundreds of papers for journals over the years. Samples of work can be supplied on request.

Calendar

2017

March 26 — **TUG 2017**, deadline for presentation proposals. tug.org/tug2017

Mar 30 – 31 — Centre for Printing History & Culture, "From Craft to Technology and Back Again: print's progress in the twentieth century", National Print Museum, Dublin, Ireland. http://www.cphc.org.uk/events

Apr 9 — **TUG election**, deadline for receipt of ballots. tug.org/election/2017

TUG 2017: TUG @ BachoTEX 2017 Bachotek, Poland.

Apr 29 – May 3 — The 38th annual meeting of the TEX Users Group, jointly with the 25th meeting of GUST and GUST's 25th birthday. "Premises, predilections, predictions". tug.org/tug2017 www.gust.org.pl/bachotex/2017-en

---

May 12 — *TUGboat* **38**:2 (proceedings issue), submission deadline.

May 21 – 26 — 16th Annual Book History Workshop, Texas A & M University, College Station, Texas. cushing.library.tamu.edu/programs/bookhistoryworkshop

May 25 – 27 — TYPO Berlin 2017, "Wanderlust", Berlin, Germany. typotalks.com/berlin

Jun 5 – 16 — Mills College Summer Institute for Book and Print Technologies, Oakland, California. millsbookartsummer.org

Jun 9 – 12 — SHARP 2017, "Technologies of the Book". Society for the History of Authorship, Reading & Publishing. Victoria, BC, Canada. www.sharpweb.org/main

Jun 26 – 29 — Book history workshop, Institut d'histoire du livre, Lyon, France. ihl.enssib.fr

Jul 5 – 7 — The Fifteenth International Conference on New Directions in the Humanities (formerly Books, Publishing, and Libraries), "New Directions of the Humanities in the Knowledge Society", Imperial College, London, UK. thehumanities.com/2017-conference

Jul 30 – Aug 3 — SIGGRAPH 2017, "At the ♡ of Computer Graphics & Interactive Techniques", Los Angeles, California. s2017.siggraph.org

Aug 8 – 11 — Digital Humanities 2017, Alliance of Digital Humanities Organizations, "Access/Accès", McGill University, Montréal, Canada. dh2017.adho.org

Aug 23 – 27 — TypeCon 2017, Boston, Massachusetts. typecon.com

Sep 1 — *TUGboat* **38**:3 (regular issue), submission deadline.

Sep 11 – 17 — 11th International ConTEXt Meeting, "ConTEXt Gardening", Maibacher Schweiz, Germany. meeting.contextgarden.net/2017

Sep 23 — DANTE 2017 Herbsttagung and 57th meeting, VHS, Mönchengladbach, Germany. www.dante.de/events.html

---

2018

Mar 2 — *TUGboat* **39**:1 (regular issue), submission deadline.

Status as of 15 March 2017

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568. e-mail: office@tug.org). For events sponsored by other organizations, please use the contact address provided.

User group meeting announcements are posted at lists.tug.org/tex-meetings. Interested users can subscribe and/or post to the list, and are encouraged to do so.

Other calendars of typographic interest are linked from tug.org/calendar.html.

TUGBOAT Volume 38 (2017), No. 1