

# TUGBOAT

Volume 35, Number 3 / 2014

<b>General Delivery</b>	230	Ab epistulis / <i>Steve Peter</i>
	231	Editorial comments / <i>Barbara Beeton</i> TeX entomology; An alternative to <code>tangle</code> and <code>weave</code> ; More Lucida fonts; More from Chuck Bigelow about Lucida; Erratum: “Online Publishing via pdf2htmlEX”, <i>TUGboat</i> 34:3; Peter Flynn’s <i>Formatting Information</i> updated; Klaus Peters, 1937–2014; Other items worth a look—bibliographies; Geographical trivia: Kolophon
	232	A footnote about ‘Oh, oh, zero’ / <i>Donald Knuth</i>
	235	Twenty Questions for Donald Knuth (on the occasion of the ePublication of <i>TAOCP</i> )
<b>Letters</b>	244	A letter on the persistence of (e)books / <i>Charles Bigelow</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	245	L <sup>A</sup> T <sub>E</sub> X document class options / <i>Thomas Thurnherr</i>
	248	How to influence the position of float environments like figure and table in L <sup>A</sup> T <sub>E</sub> X? / <i>Frank Mittelbach</i>
	255	Placing a full-width insert at the bottom of two columns / <i>Barbara Beeton</i>
	256	biblatex variations / <i>Ulrike Fischer</i>
	261	Every L <sup>A</sup> T <sub>E</sub> X document brings new programming issues / <i>David Walden</i>
	269	Glistings: Lining up / <i>Peter Wilson</i>
<b>Resources</b>	274	CTAN goes multi-lingual: Additional language support for the Web portal / <i>Gerd Neugebauer</i>
<b>Fonts</b>	276	Obyknovennaya Novaya (Ordinary New Face) in METAFONT / <i>Basil Solomykov</i>
<b>Multilingual</b>	277	A simple Arabic typesetting system for mixed Latin/Arabic documents: <i>ḍād</i> / <i>Yannis Haralambous</i>
<b>Document Processing</b>		
<b>Software &amp; Tools</b>	284	Visual editing (in a specialized case): <code>prerex</code> / <i>Bob Tennent</i>
	287	l3build—A modern Lua test suite for TeX programming / <i>Frank Mittelbach, Will Robertson, L<sup>A</sup>T<sub>E</sub>X3 team</i>
	294	MetaPost path resolution isolated / <i>Taco Hoekwater</i>
<b>Macros</b>	297	Typeset MMIX programs with TeX / <i>Udo Wermuth</i>
<b>Bibliographies</b>	309	A Citation Style Language (CSL) workshop / <i>Daniel Stender</i>
<b>Hints &amp; Tricks</b>	315	The treasure chest / <i>Karl Berry</i>
<b>Book Reviews</b>	317	Book review: <i>Practical L<sup>A</sup>T<sub>E</sub>X</i> , by George Grätzer / <i>William Adams</i>
	318	Book review: <i>Apprendre à programmer en T<sub>E</sub>X</i> , by Christian Tellechea / <i>Jacques André</i>
	319	Book review: <i>The Imitation Game</i> , by Jim Ottaviani and Leland Purvis / <i>Michael Berry</i>
	320	Book review: <i>Let’s Learn L<sup>A</sup>T<sub>E</sub>X</i> , by S. Parthasarathy / <i>Nicola Talbot</i>
<b>Abstracts</b>	322	<i>Die T<sub>E</sub>Xnische Komödie</i> : Contents of issues 2–3/2014
	323	<i>Les Cahiers GUTenberg</i> : Contents of issue 57 (2012)
<b>TUG Business</b>	230	TUGboat editorial information
	323	TeX Development Fund 2013 report
	324	TUG 2013 election
	325	TUG membership form
	326	TUG institutional members
<b>Advertisements</b>	326	TeX consulting and production services
<b>News</b>	327	TUG 2015 announcement
	328	Calendar

## **T<sub>E</sub>X Users Group**

*TUGboat* (ISSN 0896-3207) is published by the T<sub>E</sub>X Users Group.

### **Memberships and Subscriptions**

2015 dues for individual members will be 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.

Membership in the T<sub>E</sub>X 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 is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site.

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is \$110 per year, including air mail delivery.

### **Institutional Membership**

Institutional membership is primarily a means of showing continuing interest in and support for both T<sub>E</sub>X and the T<sub>E</sub>X 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. The following list of trademarks which commonly appear in *TUGboat* should not be considered complete.

T<sub>E</sub>X is a trademark of American Mathematical Society. METAFONT is a trademark of Addison-Wesley Inc. PostScript is a trademark of Adobe Systems, Inc.

[printing date: October 2014]

Printed in U.S.A.

## **Board of Directors**

Donald Knuth, *Grand Wizard of T<sub>E</sub>X-arcana*<sup>‡</sup>  
Steve Peter, *President*\*  
Jim Hefferon\*, *Vice President*  
Karl Berry\*, *Treasurer*  
Susan DeMeritt\*, *Secretary*  
Barbara Beeton  
Kaja Christiansen  
Michael Doob  
Steve Grathwohl  
Taco Hoekwater  
Klaus Höppner  
Ross Moore  
Cheryl Ponchin  
Arthur Reutenauer  
Philip Taylor  
Boris Veytsman  
David Walden  
Raymond Goucher, *Founding Executive Director*<sup>†</sup>  
Hermann Zapf, *Wizard of Fonts*<sup>†</sup>

\*member of executive committee

†honorary

See <http://tug.org/board.html> for a roster of all past and present board members, and other official positions.

### **Addresses**

T<sub>E</sub>X 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**

(Internet)

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

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

Technical support for T<sub>E</sub>X users:  
[support@tug.org](mailto:support@tug.org)

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

Copyright © 2014 T<sub>E</sub>X Users Group.

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

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

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

The first book ever printed in Europe – heavy, luxurious, pungent and creaky – does not read particularly well on an iPhone.

Simon Garfield

*Just My Type: A book about fonts*

# TUGBOAT

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

EDITOR BARBARA BEETON

VOLUME 35, NUMBER 3

PORTLAND

•

OREGON

•

•

2014

U.S.A.

---

### **TUGboat editorial information**

This regular issue (Vol. 35, No. 3) is the third and last issue of the 2014 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 (<http://tug.org/store>), and online at the *TUGboat* web site, <http://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 still 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 accepted. Please submit contributions by electronic mail to [TUGboat@tug.org](mailto:TUGboat@tug.org).

The first issue for 2015 will be a regular issue, with a deadline of March 6. The second 2015 issue will be the proceedings of TUG'15 (<http://tug.org/tug2015>); the deadline for receipt of final papers is July 31. The third issue deadline is September 25.

The *TUGboat* style files, for use with plain  $\text{\TeX}$

and  $\text{\LaTeX}$ , are available from CTAN and the *TUGboat* web site. We also accept submissions using Con $\text{\TeX}$ t. Deadlines, tips for authors, and other information:

<http://tug.org/TUGboat/location.html>

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 special arrangements.

### **TUGboat editorial board**

Barbara Beeton, *Editor-in-Chief*

Karl Berry, *Production Manager*

Boris Veytsman, *Associate Editor, Book Reviews*

### **Production team**

William Adams, Barbara Beeton, Karl Berry, Kaja Christiansen, Robin Fairbairns, Robin Laakso, Steve Peter, Michael Sofka, Christina Thiele

### **TUGboat advertising**

For advertising rates and information, including consultant listings, contact the TUG office, or see:

<http://tug.org/TUGboat/advertising.html>

---

### **Ab Epistulis**

Steve Peter

As summer fades to fall here in the northern hemisphere, contemplation strikes. It seems to be a time of looking forward as we prepare for the long winter ahead, and as it is once again election season in the US. 2015 is an election year for TUG. Several director seats will be up for election, as well as the office of president. Jim Hefferon, long-time TUG board member and current vice-president, has expressed his intent to run for president, which I am happy to support. Of course, anyone interested in serving is welcome to run for a board position or president.

The previous issue of *TUGboat* contained the proceedings of the Portland conference of 2014. The innovations coming from TUG members continues to amaze me. TUG 2015 will be held in Darmstadt, Germany, July 20–22, 2015. Keep tuned to this space and the upcoming electronic newsletters as we begin planning for this exciting meeting.

*TUGboat* 34:2, which includes two notable articles by Chuck Bigelow, is now publicly available on the TUG website (at <http://tug.org/TUGboat/Contents/contents34-2.html>). Early online access to *TUGboat* issues is one of the benefits of being a TUG member. Printed copies are available in the TUG store at <http://tug.org/store/#tugboat>.

Board member Boris Veytsman continues to write prolifically. Now online at the TUG website are two new book reviews, covering *Design Museum Fifty Typefaces That Changed The World*, by John Walters, and *The Imitation Game*, by Jim Ottaviani & Leland Purvis. For these and many more reviews, see <http://tug.org/books/#reviews>.

Until next time. Happy  $\text{\TeX}$ ing!

◇ Steve Peter  
Princeton University Press  
**president** (at) **tug dot org**  
<http://tug.org/TUGboat/Pres>

---

## Editorial comments

Barbara Beeton

### **T<sub>E</sub>X** entomology

For the past many years, I have been listed on Don Knuth's T<sub>E</sub>X web page as his official collector of bugs (see the "Errata" section of <http://www-cs-faculty.stanford.edu/~uno/abcde.html>). This is about to change.

The next review is scheduled for 2020, and it's prudent for someone younger to be the bearer of this responsibility. By unanimous consent, my successor will be Karl Berry (already the bearer of many T<sub>E</sub>X-related responsibilities), [karl@freefriends.org](mailto:karl@freefriends.org); he will officially take up the butterfly net on 1 January 2015. Although in practice we will continue to share information and consult on matters involving the history of this function, as of January 1, Karl will be the person responding to inquiries and rendering decisions on whether a report is or is not a bug. As in the past, this decision will not be reached by just one person; a few "trusted experts" (trusted and approved by Don, that is) will continue to provide advice backing up responses to reports.

Bonne chance, Karl! May you find this exercise as interesting as I have.

### An alternative to tangle and weave

In addition to his presentation at TUG 2014 on a new, fully functional, T<sub>E</sub>X-language interpreter, Doug McKenna unveiled a command-line program, *literac*, that converts source code written in languages that use C-style commenting syntax into a L<sup>A</sup>T<sub>E</sub>X document. The result, if the author has been diligent, is a literate exposition of the program under consideration. Only one file and one step is involved, unlike the dual-process *tangle* and *weave*.

The slides from the talk are posted at <http://tug.org/tug2014/slides/mckenna-literac.pdf>, though sadly without the (blindingly fast) demonstration that accompanied the presentation. We look forward to articles from Doug on this and other topics in a future *TUGboat* issue.

### More Lucida fonts

The complete (albeit growing) selection of Lucida fonts has a new venue at the Lucida fonts store, <http://lucidafonts.com>. As announced on the Bigelow & Holmes site:

We have opened a store to sell downloadable Lucida Fonts. We offer 310 fonts, most of them never before released and available only from The Lucida Fonts Store.

The letter "a" is shown in all its variety at <http://bigelowandholmes.typepad.com/>. (This page, the B&H blog, also contains a remembrance of Hans Eduard Meier, Swiss lettering artist and creator of the font Syntax, who died on 15 July 2014, age 91. Other interesting items appear in the blog as well; by the time you read this, there should be an item "How and Why We Designed Lucida".)

A "Math" page at <http://lucidafonts.com/pages/lucida-math> points to the Lucida fonts page at the TUG store (<http://tug.org/store>).

### More from Chuck Bigelow about Lucida

Lucida fonts spotted "in the wild": <http://www.pinterest.com/lucidaf/lucida-on-location/>

Chuck asks, "If you run across uses of Lucida elsewhere that are photogenic enough to be legible, let me know."

He also notes that the Louvre example is out of date; the photo was taken in 1997, but by 2012, the interior signage had been switched to use other fonts (unspecified).

### Erratum: "Online Publishing via pdf2htmlEX", *TUGboat* 34:3

In their Acknowledgement on page 323, the authors, Lu Wang and Wanmin Liu, misspelled the name of Professor Masakata Kaneko.

### Peter Flynn's *Formatting Information* updated

An early version of this manual was published in *TUGboat* 23:2. It has undergone some revisions since its original appearance in 1999. The latest (HTML) version has undergone several major changes: it is now mobile-friendly, it has a new search engine, a new index, and the chapter pages (previously quite large) have been cut into files per section so that they load faster on marginal connections.

Peter says, regarding the new release,

Some things like lines of code examples won't fit happily on very small screens. I don't think anyone has a real solution to this yet.

The examples have all been reworked, and all the package links updated (and several obsolescent packages replaced by newer ones).

The PDF and eBook will follow in due course. Please email me with all corrections, suggestions, gripes, flames, etc.

Peter's contact information is available on the web site for the manual: <http://latex.silmaril.ie/formattinginformation/>. The manual is also posted on CTAN.

### Klaus Peters, 1937–2014

Klaus Peters was a mathematician who, instead of “practicing” mathematics, preferred to use his knowledge to ensure that mathematics and other scientific literature was presented in its best, most readable form for a wide audience. His publishing ventures began with the founding of Birkhäuser, Boston, in 1979, with his wife, Alice, and continued through several other publishing houses, some of which he founded (including A K Peters Ltd.), others where he worked as an editor or consultant. His expertise and friendship were greatly valued by scores of mathematicians.

His philosophy was laid out in an article in the *AMS Notices*, “Why publish mathematics?” (<http://www.ams.org/notices/200907/rtx090700819p.pdf>). It is well worth reading, as is a shorter opinion piece on the obligations of a responsible publishing house: “ $P_V$ : The value of publishing” (<http://www.ams.org/notices/201206/rtx120600741p.pdf>).

The high standards he professed are a worthy goal for any author or publisher.

### Other items worth a look — bibliographies

The web-based service <http://www.doi2bib.org/> will accept a DOI (digital object identifier) and return a BibTeX entry for use in your bibliography. A similar facility, based on author names and titles, is offered by <http://www.ams.org/mathscinet/>, but is available only to subscribers (including many academic libraries). This by way of a reminder that Nelson Beebe has amassed an amazing collection of scientific bibliographies and tools for handling them, at <http://ftp.math.utah.edu/pub/bibnet/>.

### Geographical trivia: Kolophon

Not long ago, I attended a presentation entitled “Field Dirt”, in which were reported the projects undertaken during the summer vacation by the archaeological faculty of Brown University. One of the projects covered several sites in Turkey, which were duly displayed on a map. But wait — what’s that name “Kolophon” doing there?

This city was founded by the Greeks around the turn of the first millennium B.C. as “Κολοφών”. (One of the most renowned Ionian cities until its conquest and decline in the 7th century B.C., it has been cited as a possible home or birthplace for Homer.) The origin of the name is the word κολοφών, meaning “summit”, on account of its location, built on three hills. The bibliographic “colophon” is from the same source, with the metaphorical sense of “crowning touch”, a feature nowadays too often missed.

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

### A footnote about ‘Oh, oh, zero’

Don Knuth

I can’t resist adding a few comments to Chuck Bigelow’s wonderful essay about the history of zero-versus-oh in *TUGboat* **34**:2 (2013), 168–181.

As an associate editor of ACM’s *Journal and Communications* during the 1960s, and as a prospective author, I’d been giving some thought to the fact that new concepts arising in computer science were calling for new typographic conventions. In particular, I corresponded with Myrtle Kellington, who was responsible for typography in all of ACM’s publications. (Computer scientists have Myrtle to thank for the now-universal style in which computer programs have long been typeset with a pleasant mixture of roman, bold, and italic. She introduced this style when she masterminded the publication of the Algol 60 Report, first with roman and bold only [1] and later with italic too [2].)

I applauded her work on formatted algorithms, but I wasn’t happy with the appearance of various papers about machine input and output, in cases where a monospaced font would have improved the exposition. When she learned of my concerns she wrote to me on 10 February 1966:

Whenever the vertical alignment is a requirement the printer uses the only typeface he has where this applies. It is called “Typewriter,” is machine-set, is available in four sizes, and looks like the old-fashioned typewriter style. A sample is enclosed.

Actually the sample she sent had three different fonts, and it didn’t exhibit the full character sets. Those fonts (all to be used on Monotype machines) were: Typewriter No. 74 (eight point size); Remington No. 70 (ten point and twelve point sizes); and Remington No. 17 (eleven point size), which was somewhat darker. Only No. 17 was available for machine setting; the other styles needed to be inserted by hand.

I replied on 14 February 1966 — evidently the U.S. Postal service was quite efficient in those days! — as follows:

Regarding my request for a special type-font, I believe the 8 point “Typewriter No. 74” will do very nicely (assuming there are commas, parentheses, and the usual other special characters found on a typewriter). The other styles are also adequate but not as good. If possible it would be preferable to have a more squarish capital letter O so it can’t be mistaken for zero. I don’t know how expensive it

is to make up new characters one at a time; I realize a whole new font can be very costly. The special characters will no doubt be the major problem.

As examples of printed material *using* this style, I can only say unfortunately I don't know of any, except Addison-Wesley is doing it for me in the forthcoming set of books I'm writing. I think ACM should "pioneer" in this. The best example I can give you is point out sections of the last *Communications of the ACM* issue (January '66) which would have been much improved if set in a "typewriter" style:

Page 5, right column, the "all caps" words.

Page 6, "all caps" words and displayed programs.

Page 9, Appendix C. Possibly appendix B also.

Page 30, the tables, if type set, would be candidates for fixed width, although in this case the line engravings were quite adequate as a substitute.

Page 31, the words in all caps.

Page 32 ff. The FORTRAN and COBOL program example.

Pages 36–37, the capitalized "machine response" sentences.

Page 41 ff. All-caps words except perhaps ELIZA.

In general, FORTRAN and COBOL and assembly language programs and references to symbols within such programs would look much better in a fixed-width style. Even an 8 point type would be satisfactory here for appearance sake in the midst of the normal 12 point type (it would look like "small caps"), although perhaps there would be some trouble from the monotype side in such mixing of sizes, I don't know. You already have good formats for printing ALGOL programs, and that needs no change; but these others look quite unreadable by comparison, particularly things like Appendix C on page 9.

The distinction between "oh" and "zero" is reasonably important. On page 6 I see the word "TO" which should have really been tee-zero.

I would like to continue discussing this with you by letter. Can you tell me what special characters are available, how difficult

it is to mix 8 point in with 12 point text, etc.? I will be very glad to mark up all papers that go through my desk in a special way to indicate what parts should be in this fixed-width style.

Unfortunately, Myrtle's reply (dated 17 February) shot all these ideas down:

Dear Don: I am grateful for your interest in the printing aspects . . . However, I had not realized from your earlier letter that this new or special typeface with fixed-width characters would have to be used for words run in the text — not displayed, that is. What I am trying to say is that I had assumed that use of the Typewriter typeface would be only for certain special displayed sequences or programs.

Incorporating the Typewriter, or any other typeface, for isolated words or groups of words into the customary text would lead to prohibitive costs.

Let me tell you what can be done staying within machine-set composition, which is the most economical form of typesetting in letter press composition, and perhaps we can evolve a plan that would achieve what you are after, or at least partially.

The typeface used for the text in *CACM* is Modern No. 8, referred to as #8. This is available in all sizes both in roman and italics. . . . Along with #8 we can have, for machine setting that is, one other typeface. The one we use is Bodoni boldface, called #275. . . . To summarize, we can intermix in one keyboard operation for machine setting the following typefaces and sizes:

In size 10 on 12,

#8, in roman: all caps; csc (caps and small caps); sc (all small caps); and clc (caps and lowercase).

#8, in italics: all caps, and clc.

#275, in roman: all caps, and clc.

#275, in ital: all caps, and clc.

In 8 on 10, the same range.

But the two sizes, 8 pt and 10 pt, cannot be intermixed on the same line, I am sure you know — unless of course hand work is involved.

Thus for all the examples you mentioned in your letter, one could not have the Type-writer typeface without an entire special hand operation to drop all those words in after the regular text had been set by machine in #8 and #275.

... Actually we are giving considerable attention to changing printers, originally motivated by saving money, but now by many other considerations such as automated typesetting or being prepared for a fully automated operation all the way along the line, as well as quality of the printing. With certain of these new cold type and automated composition processes, one can intersperse several typefaces and adjust the spacings almost at whim. And your request would be no problem.

I will bear all this in mind as we carry on our interviews and observe demonstrations.

(Indeed, ACM did eventually change to “cold type” printing, and it turned out to be a mistake — although I’m sure they tried valiantly to work with the available vendors during those years. Decent mathematical typesetting was becoming a lost art; and that, of course, is why I was motivated to develop  $\text{\TeX}$  some years later. A downward spiral of decreasing typographic quality in *Communications of the ACM* began with their issue of March 1971; various examples of fixed-width type can incidentally be found in that issue, all of which were poorly reproduced from line-printer output. The *Journal of the ACM* began to suffer the same fate in October 1976.)

Meanwhile, as indicated in my letter to Myrtle, I had been having much better responses from Addison-Wesley, as they were preparing to publish *The Art of Computer Programming*. Addison-Wesley had been founded by two printers who were interested in producing good textbooks; and they became the only scientific publisher with their own in-house composition facility, at least in America. All of their typesetting was done under the direction of an old-timer named Hans Wolf. At my request, Hans had figured out a clever way to adapt his Monotype keyboards and casters so that machine setting with Remington #17 could actually be mixed together with the normal roman, italic, bold, bold italic, and math symbols. (Indeed, I’m pretty sure that this had never been done before, because Hans had originally told me — as Myrtle was to do later — that such a thing would be impossible.) This mixing could be done either in 10-point type on a 12-point base, or 9-point type on an 11-point base.

Furthermore, Hans and Addison-Wesley agreed to have a special glyph cut for me, a “squarish” version of the uppercase O, compatible with the existing Remington font. The font also included a new special character like ‘ $\sqcup$ ’ to indicate a blank space.

Therefore the publication of Volume 1 of *TAOCP* in January 1968 was actually the debut of a brand-new typographic style for computer science, featuring typewriter style blended freely with ordinary text in appropriate places. The new ‘O’ didn’t quite align properly with the other letters at the baseline; but I didn’t actually notice that glitch at the time, because I was so happy to have ‘O’ instead of ‘0’.

Why did I ask Addison-Wesley for a squarish Oh? I was almost surely influenced primarily by the dot-matrix font used by keypunch machines in those days. Look, for example, at the illustration of a punched card on page 148 of the original 1968 edition of my book, or on page 152 of the current edition. Bob Bemer’s article [3, page 516] also shows it as IBM’s recommended corporate practice for distinguishing Oh from zero on keypunches, as of 1964. On typewriters, IBM recommended a wide Oh and a narrow zero, “except for the stylized fonts for OCR and MICR.”

When Volume 2 of *The Art of Computer Programming* came out in 1981, with glyphs now drawn by METAFONT, of course I retained my beloved ‘O’. And the ‘Q’ became squarish too at this time (although with a loop at the bottom instead of a crossbar).

Alas, however, Chuck’s essay demonstrates that I’m still standing alone in this respect: None of the nine monospaced typefaces in his Fig. 9 have anything like an Oh that I would want to use. (Nowhere did I see a really satisfactory Oh in Chuck’s discussion — until I came to Karl Berry’s production notes at the end, and Karl’s reference to `ZeroFontOT.otf`.) I herewith submit a humble request to have squarish O and Q available as alternates in the next edition of Lucida Console.

## References

- [1] Peter Naur et al., “Report on the Algorithmic Language ALGOL 60,” *Communications of the ACM* **3** (1960), 299–314.
- [2] Peter Naur et al., “Revised report on the Algorithmic Language ALGOL 60,” *Communications of the ACM* **6** (1963), 1–17.
- [3] R. W. Bemer, “Toward standards for handwritten zero and oh,” *Communications of the ACM* **10** (1967), 513–518.

◇ Don Knuth  
<http://www-cs-staff.stanford.edu/~uno>

---

## Twenty Questions for Donald Knuth, to celebrate the ePublication of *TAOCP*

To celebrate the publication of the eBooks of *The Art of Computer Programming (TAOCP)*, Pearson asked several computer scientists, contemporaries, colleagues, and well-wishers to pose one question each to author Donald E. Knuth. Here are his answers. (Reprinted in *TUGboat* by kind permission of Pearson, from [www.informit.com/promotions/impact-of-the-art-of-computer-programming-139881](http://www.informit.com/promotions/impact-of-the-art-of-computer-programming-139881).)

**1. Jon Bentley, researcher:** What a treat! The last time I had an opportunity like this was at the end of your data structures class at Stanford in June, 1974. On the final day, you opened the floor so that we could ask any question on any topic, barring only politics and religion. I still vividly remember one question that was asked on that day: “Among all the programs you’ve written, of which one are you most proud?”

Your answer (as I approximately recall it, four decades later) described a compiler that you wrote for a minicomputer with 1024 available bytes of memory. Your first draft was 1029 bytes long, but you eventually had it up and running and debugged at 1023 bytes. You said that you were particularly proud of cramming so much functionality into so little memory.

My query today is a slight variant on that venerable question. Of all the programs that you’ve written, what are some of which you are most proud, and why?

**Don Knuth:** I’d like to ask you the same! But that’s something like asking parents to name their favorite children.

Of course I’m proud of  $\text{\TeX}$  and  $\text{\METAFONT}$ , because they seem to have helped to change the world, and because they led to many friendships. Furthermore they’ve made these eBooks possible: I’m enormously happy that the work I did more than 30 years ago has miraculously survived many changes of technology, and that the 3,000 pages of *TAOCP* now look so great on a little tablet — even after zooming.

While I was preparing for Volume 4 of *TAOCP* in the 90s, I wrote several dozen short routines using what you and I know as “literate programming.” Those little essays have been packaged into *The Stanford GraphBase* (1994), and I still enjoy using and modifying them. My favorite is the implementation of Tarjan’s beautiful algorithm for strong components, which appears on pages 512–519 of that book.

I have to admit some pride also in the implementation of IEEE floating-point arithmetic that appears

in my book *MMIXware* (1999), as well as that book’s metasimulator for MMIX, in which I explain many principles of advanced pipelined computers from the ground up.

Literate programming continues to be one of the greatest joys of my life. In fact, I find myself writing roughly two programs per week, on average, both large and small, as I draft new material for the next volumes of *TAOCP*.

**2. Dave Walden,  $\text{\TeX}$  Users Group:** Might you publish the original 3,000-page version of *TAOCP* (before the decision to change it into seven volumes), as a historical artifact of your view of the state of the art of algorithms and their analysis circa 1965? I think lots of people would like to see this.

**Don Knuth:** Scholars can look at the handwritten pages that led to Volumes 1–3 by going to the Stanford Archives, and all of the remaining pages will be deposited there eventually. I see little value in making those drafts more generally available — although some of the material about baseball that I decided not to use is pretty cool. Archives from the real pioneers of computer science, who wrote in the 40s and 50s, should be published first.

I do try to retain the youthful style of the original, in the pages that I write today, except where my first draft was embarrassingly naive or corny. I’ve also learned when to say “that” instead of “which,” thanks in part to Guy Steele’s tutelage.

**3. Charles Leiserson, MIT:** *TAOCP* shows a great love for computer science, and in particular, for algorithms and discrete mathematics. But love is not always easy. When writing this series, when did you find yourself reaching deepest into your emotional reservoir to overcome a difficult challenge to your vision?

**Don Knuth:** Again, Charles, I’d like to ask you exactly the same question!

For me, I guess, the hardest thing has always been to figure out what to cut. And I obviously haven’t been very successful at that, in spite of much rewriting.

The most difficult technical challenge was to write the metasimulator for MMIX. I needed to do that behind the scenes, in order to shape what actually appears in the books, and it was surely the toughest programming task that I’ve ever faced. Without the methodology of literate programming, I don’t think I could have finished that job successfully.

Many of the “starred” mathematical sections also stretched me pretty far. Overall, however, after working on *TAOCP* for more than fifty years, I can’t think of any aspect of the activity where the effort

of writing wasn't amply repaid by what I learned while doing it.

**4. Dennis Shasha, NYU:** How does a beautiful algorithm compare to a beautiful theorem? In other words, what would be your criteria of beauty for each?

**Don Knuth:** Beauty has many aspects, of course, and is in the eye of the beholder. Some theorems and algorithms are beautiful to me because they have many different applications; some because they do powerful things with severely limited resources; some because they involve aesthetically pleasing patterns; some because they have a poetic purity of concept.

For example, I mentioned Tarjan's algorithm for strong components. The data structures that he devised for this problem fit together in an amazingly beautiful way, so that the quantities you need to look at while exploring a directed graph are always magically at your fingertips. And his algorithm also does topological sorting as a byproduct.

It's even possible sometimes to prove a beautiful theorem by exhibiting a beautiful algorithm. Look, for instance, at Theorem 5.1.4D and/or Corollary 7H in *TAOCP*.

**5. Mark Taub, Pearson:** Does the emergence of "apps" (small, single-function, networked programs) as the dominant programming paradigm today impact your plans in any way for future material in *TAOCP*?

**Don Knuth:** People who write apps use the ideas and paradigms that are already present in the first volumes. And apps make use of ever-growing program libraries, which are intimately related to *TAOCP*. Users of those libraries ought to know something about what goes on inside.

Future volumes will probably be even more "app-likable," because I've been collecting tons of fascinating games and puzzles that illustrate programming techniques in especially instructive and appealing ways.

**6. Radia Perlman, Intel:** (1) What is not in the books that you wish you'd included? (2) If you'd been born 200 years ago, what kind of career might you imagine you'd have had?

**Don Knuth:** (1) Essentially everything that I want to include is either already in the existing volumes or planned for the future ones. Volume 4B will begin with a few dozen pages that introduce certain newfangled mathematical techniques, which I didn't know about when I wrote the corresponding parts of Volume 1. (Those pages are now viewable from my website in beta-test form, under the name "mathematical preliminaries redux.") I plan to issue similar

gap-filling "fascicles" when future volumes need to refer to recently invented material that ultimately belongs in Volume 3, say.

(2) Hey, what a fascinating question — I don't think anybody else has ever asked me that before!

If I'd been born in 1814, the truth is that I would almost certainly have had a very limited education, coupled with hardly any access to knowledge. My own male ancestors from that era were all employed as laborers, on farms that they didn't own, in what is now called northern Germany.

But I suppose you have a different question in mind. What if I had been one of the few people with a chance to get an advanced education, and who also had some flexibility to choose a career?

All my life I've wanted to be a teacher. In fact, when I was in first grade, I wanted to teach first grade; in second grade, I wanted to teach second; and so on. I ended up as a college teacher. Thus I suppose that I'd have been a teacher, if possible.

To continue this speculation, I have to explain about being a geek. Fred Gruenberger told me long ago that about 2% of all college students, in his experience, really resonated with computers in the way that he and I did. That number stuck in my mind, and over the years I was repeatedly able to confirm his empirical observations. For instance, I learned in 1977 that the University of Illinois had 11,000 grad students, of whom 220 were CS majors!

Thus I came to believe that a small percentage of the world's population has somehow acquired a peculiar way of thinking, which I happen to share, and that such people happened to discover each other's existence after computer science had acquired its name.

For simplicity, let me say that people like me are "geeks," and that geeks comprise about 2% of the world's population. I know of no explanation for the rapid rise of academic computer science departments — which went from zero to one at virtually every college and university between 1965 and 1975 — except that they provided a long-needed home where geeks could work together. Similarly, I know of no good explanation for the failure of many unsuccessful software projects that I've witnessed over the years, except for the hypothesis that they were not entrusted to geeks.

So who were the geeks of the early 19th century? Beginning a little earlier than 1814, I'd maybe like to start with Abel (1802); but he's been pretty much claimed by the mathematicians. Jacobi (1804), Hamilton (1805), Kirkman (1806), De Morgan (1806), Liouville (1809), Kummer (1810), and China's Li Shanlan (1811) are next; I'm listing "mathematicians" whose writings speak rather directly to the

geek in me. Then we get precisely to your time period, with Catalan (1814) and Sylvester (1814), Boole (1815), Weierstrass (1815), and Borchardt (1817). I would have enjoyed the company of all these people, and with luck I might have done similar things.

By the way, the first person in history whom I'd classify as "100% geek" was Alan Turing. Many of his predecessors had strong symptoms of our disease, but he was totally infected.

**7. Tony Gaddis, author:** Do you remember a specific moment when you discovered the joy of programming, and decided to make it your life's work?

**Don Knuth:** During the summer of 1957, between my freshman and sophomore years at Case Tech in Cleveland, I was allowed to spend all night with an IBM 650, and I was totally hooked.

But there was no question of viewing that as a "life's work," because I knew of nobody with such a career. Indeed, as mentioned above, my life's work was to be a teacher. I did write a compiler manual in 1958, which by chance was actually used as the textbook for one of my classes in 1959(!). Still, programming was for me primarily a hobby at first, after which it became a way to support myself while in grad school.

I saw no connection between computer programming and my intended career as a math professor until I met Bob Floyd late in 1962. I didn't foresee that computer science would ever be an academic discipline until I met George Forsythe in 1964.

**8. Robert Sedgewick, Princeton:** Don, I remember some years ago that you took the position that you weren't trying to reach everyone with your books—knowing that they would be particularly beneficial to people with a certain interest and aptitude who enjoy programming and exploring its relationship to mathematics. But lately I've been wondering about your current thoughts on this issue. It took a long time for society to realize the benefits of teaching everyone to read; now the question before us is whether everyone should learn to *program*. What do you think?

**Don Knuth:** I suppose all college professors think that their subject ought to be taught to everybody in the world. In this regard I can't help quoting from a wonderful paper that John Hammersley wrote in 1968:

Just for the fun of getting his reactions, I asked an eminent scholar of English Literature what educational benefits might lie in the study of goliardic verse, Erse curses, and runic erotica. 'A working background of goliardic verse would be more than helpful to anyone hoping to have some modest facility in his own mother tongue', he declared; and with that he warmed to his subject and to the poverties of unlettered science, so

that it was some minutes before I could steer him back to the Erse curses, about which he seemed a good deal less enthusiastic. 'Really', he said, 'that sort of thing isn't my subject at all. Of course, I applaud breadth of vocabulary; and you never know when some seemingly useless piece of knowledge may not turn out to be of cardinal practical importance. I could certainly *envisage* a situation in which they might come in very handy indeed'. 'And runic erotica?' 'Not extant'. (Was it only my fancy that heard a note of faint regret in his reply?) Certainly the higher flights of scholarship can add savour; but does the man-in-the-street have the time and the pertinacity and the intellectual digestion for them?

Programming, of course, is not just an ordinary subject. It is intrinsically empowering, and applicable to many different kinds of knowledge. And I also know that you've been having enormous successes, at Princeton and online, teaching advanced concepts of programming to students from every discipline.

But your question asks about *everybody*. I still think many years will have to go by before I would recommend that my own highly intelligent wife, son, and daughter should learn to program, much less that everybody else I know should do so.

Nick Trefethen told me a few years back that he had just visited his son's high school in Oxford, which is one of the best anywhere, and learned that not a single student knew how to program! Britain is now beginning to change that, indeed at a more rapid pace than in America. Yet such a revolution almost surely needs to take place over a generation or more. Where are the teachers going to come from?

My own experience is with the subset of college students who are sufficiently interested in programming that they expect it to become an integral part of their life. *TAOCP* is essentially for specialists. I've primarily been writing it for geeks, not for a general audience, because somebody has to write books that aren't for dummies. (By a "dummy" I mean a smart non-geek. That's a much larger market, and very important; but it's not my target audience, and general education is not my forte.)

On the other hand, believe it or not, I try to explain everything in my books by imagining a non-specialist reader. My goal is to be jargon-free whenever possible; I especially try to avoid terms from higher mathematics that tend to frighten the programmer-on-the-street. Whenever possible I try to translate results from the theoretical literature into a language that high-school students could understand.

I know that my books still aren't terribly easy to fathom, even for geeks. But I could have made them much, much harder.

**9. Barbara Steele:** What was the conversion process, and what tools did you use, to convert your print books to eBooks?

**Don Knuth:** I knew that these volumes would not work especially well as eBooks unless they were converted by experts. Fortunately I received some prize money in 2011, which could be used to pay for professional help. Therefore I was able to achieve the kind of quality that I envisioned, without delaying my work on future volumes, by letting the staff at Mathematical Sciences Publishers in Berkeley (MSP) handle all of the difficult stuff.

My principal goal was to make the books easily searchable — and that’s a much more challenging problem than it seems, if you want to do it right. Secondly, I wanted to let readers easily click on the number of any exercise or equation or illustration or table or algorithm, etc., and to jump to that exercise; also to jump readily between an exercise and its answer.

The people at MSP wrote special software that converts my source text into suitable input to other software that creates PDF files. I don’t know the details, except that they use “change files” analogous to those used in WEB and CWEB. I’ve checked the results pretty carefully, and I couldn’t be more pleased. Moreover, they’ve designed things so that it won’t be hard for me to make changes next year, as readers discover bugs in the present editions.

(My style of writing tends to maximize the number of opportunities to make mistakes, hence I would be fooling myself if I thought that the books were now perfect. Therefore it has always been important to keep future errata in mind. The production staff at Addison-Wesley has been consistently wonderful in the way they allow me to correct about fifty pages every year in each volume.)

**10. Silvio Levy, MSP:** Could you comment on the differences between the print, PDF, EPUB, etc., editions of *TAOCP*? What would you say is gained or lost with each?

**Don Knuth:** The printed versions weigh a *lot* more, but they don’t need battery power or a tether to electricity. They are always there; I don’t have to turn them on, and I can have them all open at once.

I can scribble in the margins (and elsewhere) of the print versions, and I can highlight text in different colors. Ten years from now I expect analogous features will be commonly available for eBooks.

I’m used to flipping pages and finding my way around a regular book, much more so than in an eBook; but my grandchildren might have the opposite reaction.

The great advantage of an eBook is the reader’s ability to search exhaustively. What fun it is to look for all occurrences of a random word like ‘game’, or for a random word fragment like ‘gam’ or ‘ame’, and find lots of cool material that I don’t recall having written. The search feature on these books works even better than I had a right to hope for.

The index in a printed book has the advantage of being more focused. But that index also appears in the eBook, and in the eBook you can even click in the index to get to the cited pages.

Today’s eBook readers are often inconvenient for setting bookmarks and going back to where you were a couple of minutes ago, especially after you click on an Internet link and then want to go back to reading. But that software will surely improve, and so will today’s electronic devices.

In the future I look forward to curated eBooks that have additional notes by experts — and possibly even graffiti in the style of *Concrete Mathematics* — somewhat analogous to the “director’s comments” and other extras found on the DVDs for films. One could select different subsets of these comments when reading.

**11. Peter Gordon, Addison-Wesley (retired):** If the full range of today’s eBook features and functionalities had been available when *TAOCP* was first published, would you have written those volumes very differently?

**Don Knuth:** Well, I don’t think I would have gotten very far at all. I would have had to think about doing everything in color, and with interactive figures, tables, equations, and exercises. A single person cannot use the “full range” of features that eBooks potentially have.

But by limiting myself to what can be presented well in black-and-white type, on printed pages of a fixed size, I was fortunately able to complete 3,000 pages over a period of 50 years.

**12. Udi Manber, Google:** The early volumes of *TAOCP* established computer programming as computer science. They introduced the necessary rigor. This was at the time when computers were used mostly for numerical applications. Today, most applications are related to people — social interaction, search, entertainment, and so on. Rigor is rarely used in the development of these applications. Speed is not always the most important factor, and “correctness” is rarely even defined. Do you have any advice on how to develop a new computer science that can introduce rigor to these new applications?

**Don Knuth:** The numerical computations that were somewhat central when computer science was

born are by no means gone; they continue to grow, year by year. Of course, they now represent a much smaller piece of the pie, but I don't believe in concentrating too much on the big pieces.

My work on METAFONT introduced me to applications where “correctness” cannot be defined. How do I know, for example, that my program for the letter *A* produces a correct image? I never will; and I've learned to live with that uncertainty. On the other hand, when I implemented the routines that interpret specifications and draw the associated bitmaps, there was plenty of room for rigor. The algorithms that go into font rendering are among the most interesting I've ever seen.

As a user of products from Google and Adobe and other corporations, I know that a tremendous amount of rigor goes into the manipulation of map data, transportation data, pixel data, linguistic data, metadata, and so on. Furthermore, much of that processing is done with distributed and decentralized algorithms that require more rigor than anybody ever thought of in the 60s.

So I can't say that rigor has disappeared from the computer science scene. I do wish, however, that Google's and Adobe's and Apple's programmers would learn rigorously how to keep their systems from crashing my home computers, when I'm not using Linux.

In general I agree with you that there's no decrease in the need for rigor, rather an increase in the number of kinds of rigor that are important. The fact that correctness can't be defined on the “bottom line” should not lull people into thinking that there aren't intermediate levels within every nontrivial system where correctness is crucial. Robustness and quality are compromised by every weak link.

On the other hand, I certainly don't think that everything should be mathematized, nor that everything that involves computers is properly a subdiscipline of computer science. Many parts of important software systems do not require the special talents of geeks; quite the contrary. Ideally, many disciplines collaborate, because a wide variety of orthogonal skill sets is a principal reason why life is such a joy. *Vive la difference.*

Indeed, I myself follow the path of rigor only partway: Rarely do I ever give a formal proof that any of my programs are correct, once I've constructed an informal proof that convinces me. I have no real interest, for example, in defining exactly what it would mean for  $\text{\TeX}$  to be correct, or for verifying formally that my implementation of that 550-page program is free of bugs. I know that anomalous results are possible when users try to specify pages

that are a mile wide, or constants that involve a trillion zeros, etc. I've taken care to avoid catastrophic crashes, but I don't check every addition operation for possible overflow.

There's even a fundamental gap in the foundations of my main mathematical specialty, the analysis of algorithms. Consider, for example, a computer program that sorts a list of numbers into order. Thanks to the work of Floyd, Hoare, and others, we have formal definitions of semantics, and tools by which we can verify that sorting is indeed always achieved. My job is to go beyond correctness, to an analysis of such things as the program's running time: I write down a recurrence, say, which is supposed to represent the average number of comparisons made by that program on random input data. I'm 100% sure that my recurrence correctly describes the program's performance, and all of my colleagues agree with me that the recurrence is “obviously” valid. *Yet I have no formal tools by which I can prove that my recurrence is right.* I don't really understand my reasoning processes at all! My student Lyle Ramshaw began to create suitable foundations in his thesis (1979), but the problem seems inherently difficult. Nevertheless, I don't lose any sleep over this situation.

**13. Al Aho, Columbia:** We all know that the Turing Machine is a universal model for sequential computation.

But let's consider reactive distributed systems that maintain an ongoing interaction with their environment — systems like the Internet, cloud computing, or even the human brain. Is there a universal model of computation for these kinds of systems?

**Don Knuth:** I'm not strong on logic, so *TAOCP* treads lightly on this sort of thing. The *TAOCP* model of computation, discussed on pages 4–8 of Volume 1, considers “reactive processes,” a.k.a. “computational methods,” which correspond to single processors. I've long planned to discuss recursive coroutines and other cooperative processes in Chapter 8, after I finish Chapter 7. The beautiful model of context-free parsing via semiautonomous agents, in Floyd's great survey paper of 1964, has strongly influenced my thinking in this regard.

I'd like to see extensions of the set-theoretic model of computation at the beginning of Volume 1 to the things you mention. They might well shed light on the subject.

But fully distributed processes are well beyond the scope of my books and my own ability to comprehend them. For a long time I've thought that an understanding of the way ant colonies are able to perform incredibly organized tasks might well be the

key to an understanding of human cognition. Yet the ants that invade my house continually baffle me.

**14. Guy Steele, Oracle Labs:** Don, you and I are both interested in program analysis: What can one know about an algorithm without actually executing it? Type theory and Hoare logic are two formalisms for that sort of reasoning, and you have made great contributions to using mathematical tools to analyze the execution time of algorithms. What do you think are interesting currently open problems in program analysis?

**Don Knuth:** Guy, I'm sure you aren't really *against* the idea of program execution. You and I both like to know things about programs *and* to execute them. Often the execution contradicts our supposed knowledge.

The quest for better ways to verify programs is one of the famous grand challenges of computer science. And as I said to Udi, I'm particularly rooting for better techniques that will avoid crashes.

Just now I'm writing the part of Volume 4B that discusses algorithms for satisfiability, a problem of great industrial importance. Almost nothing is known about why the heuristics in modern solvers work as well as they do, or why they fail when they do. Most of the techniques that have turned out to be important were originally introduced for the wrong reasons!

If I had my druthers, I wish people like you would put a lot of effort into a problem of which I've only recently become aware: The programmers of today's multithreaded machines need new kinds of tools that will make linked data structures much more cache-friendly. One can in many cases start up auxiliary parallel threads whose sole purpose is to anticipate the memory accesses that the main computational threads will soon be needing, and to preload such data into the cache. However, the task of setting this up is much too daunting, at present, for an ordinary programmer like me.

**15. Robert Tarjan, Princeton:** What do you see as the most promising directions for future work in algorithm design and analysis? What interesting and important open problems do you see?

**Don Knuth:** My current draft about satisfiability already mentions 25 research problems, most of which are not yet well known to the theory community. Hence many of them might well be answered before Volume 4B is ready. Open problems pop up everywhere and often. But your question is, of course, really intended to be much more general.

In general I'm looking for more focus on algorithms that work fast with respect to problems whose

size,  $n$ , is *feasible*. Most of today's literature is devoted to algorithms that are asymptotically great, but they are helpful only when  $n$  exceeds the size of the universe.

In one sense such literature makes my life easier, because I don't have to discuss those methods in *TAOCP*. I'm emphatically *not* against pure research, which significantly sharpens our abilities to deal with practical problems and which is interesting in its own right. So I sometimes play asymptotic games. But I sure wouldn't mind seeing a lot more algorithms that I could also use.

For instance, I've been reading about algorithms that decide whether or not a given graph  $G$  belongs to a certain class. Is  $G$ , say, chordal? You and others discovered some great algorithms for the chordality and minimum fillin problems, early on, and an enormous number of extremely ingenious procedures have subsequently been developed for characterizing the graphs of other classes. But I've been surprised to discover that very few of these newer algorithms have actually been implemented. They exist only on paper, and often with details only sketched.

Two years ago I needed an algorithm to decide whether  $G$  is a so-called comparability graph, and was disappointed by what had been published. I believe that all of the supposedly "most efficient" algorithms for that problem are too complicated to be trustworthy, even if I had a year to implement one of them.

Thus I think the present state of research in algorithm design misunderstands the true nature of efficiency. The literature exhibits a dangerous trend in contemporary views of what deserves to be published.

Another issue, when we come down to earth, is the efficiency of algorithms on real computers. As part of the Stanford GraphBase project I implemented four algorithms to compute minimum spanning trees of graphs, one of which was the very pretty method that you developed with Cheriton and Karp. Although I was expecting your method to be the winner, because it examines much of the data only half as often as the others, it actually came out two to three times *worse* than Kruskal's venerable method. Part of the reason was poor cache interaction, but the main cause was a large constant factor hidden by  $O$  notation.

**16. Frank Ruskey, University of Victoria:** Could you comment on the importance of working on unimportant problems? My sense is that computer science research, funding, and academic hiring is becoming more and more focused on short-term problems that have at their heart an economic motivation. Do you agree with this assessment, is it a bad

trend, and do you see a way to mitigate it?

Similarly, could you comment on the demise of the individual researcher? So many papers that I see published these days have multiple authors. Five-author papers are routine. But when I dig into the details it seems that often only one or two have contributed the fresh ideas; the others are there because they are supervisors, or financial contributors, or whatever. I'm pretty sure that Euler didn't publish any papers with five co-authors. What is the reason for this trend, how does it interfere with trying to establish a history of ideas, and what can be done to reverse it?

**Don Knuth:** I was afraid somebody was going to ask a question related to economics. I've never understood anything about that subject. I don't know why people spend money to buy things. I'm willing to believe that some economists have enough wisdom to keep the world running some of the time, but their reasons are beyond me.

I just write books. I try to tell stories that seem to be important, at least for geeks. I've never bothered to think about marketing, or about what might sell, except when my publishers ask me to answer questions as I'm doing now!

Three years ago I published *Selected Papers on Fun and Games*, a 750-page book that is entirely devoted to unimportant problems. In many ways the fact that I was able to live during a time in the history of the world when such a book could be written has given me even more satisfaction than I get when seeing the currently healthy state of *TAOCP*.

I've reached an age where I can fairly be described as a "grumpy old man," and perhaps that is why I strongly share your concern for the alarming trends that you bring up. I'm profoundly upset when people rate the quality of my work by measuring the extent to which it affects Wall Street.

Everybody seems to understand that astronomers do astronomy because astronomy is interesting. Why don't they understand that I do computer science because computer science is interesting? And that I'd do it regardless of whether or not it made money for anybody? The reason is probably that not everybody is a geek.

Regarding joint authorship, you are surely right about Euler in the 18th century. In fact I can't think of *any* two-author papers in mathematics, until Hardy and Littlewood began working together at the beginning of the 20th century.

In my own case, two of my earliest papers were joint because the other authors did the theory and I wrote computer programs to validate it. Two other papers were related to the ALGOL language, and

done together with ACM committees. In a number of others, written while I was at Caltech, I did the theory and my student co-authors wrote computer programs to validate it. There was one paper with Mike Garey, Ron Graham, and David Johnson, in which they did the theory and my role was to explain what they did. You and I wrote a joint paper in 2004, related to recursive coroutines, in which we shared equally.

The phenomenon of hyperauthorship still hasn't infected computer science as much as it has hit physics and biology, where I've read that Thomson-Reuters indexed more than 200 papers having 1,000 authors or more, in a single recent year! When I cite a paper in *TAOCP*, I like to mention all of the authors, and to give their full names in the index. That policy will become impossible if CS publication practices follow in the footsteps of those fields.

Collaborative work is exhilarating, and it's wonderful when new results are obtained that wouldn't have been discovered by individuals working alone. But as you say, authors should be authors, not hangers-on.

You mention the history of ideas. To me the method of discovery tends to be more important than the identification of the discoverers. Still, credit should be given where credit is due; conversely, credit shouldn't be given where credit isn't due.

I suppose the multiple-author anomalies are largely due to poor policies related to financial rewards. Unenlightened administrators seem to base salaries and promotions on publication counts.

What can we do? As I say, I'm incompetent to deal with economics. I've gone through life refusing to go along with a crowd, and bucking trends with which I disagree. I've often declined to have my name added to a paper. But I suppose I've had a sheltered existence; young people may be forced to bow to peer pressure.

**17. Andrew Binstock, Dr. Dobb's:** At the ACM Turing Centennial in 2012, you stated that you were becoming convinced that  $P = NP$ . Would you be kind enough to explain your current thinking on this question, how you came to it, and whether this growing conviction came as a surprise to you?

**Don Knuth:** As you say, I've come to believe that  $P = NP$ , namely that there does exist an integer  $M$  and an algorithm that will solve every  $n$ -bit problem belonging to the class  $NP$  in  $n^M$  elementary steps.

Some of my reasoning is admittedly naive: It's hard to believe that  $P \neq NP$  and that so many brilliant people have failed to discover why. On the other hand if you imagine a number  $M$  that's finite

but incredibly large — like say the number  $10^{\uparrow\uparrow\uparrow 3}$  discussed in my paper on “coping with finiteness” — then there’s a humongous number of possible algorithms that do  $n^M$  bitwise or addition or shift operations on  $n$  given bits, and it’s really hard to believe that all of those algorithms fail.

My main point, however, is that I don’t believe that the equality  $P = NP$  will turn out to be helpful even if it is proved, because such a proof will almost surely be nonconstructive. Although I think  $M$  probably exists, I also think human beings will never know such a value. I even suspect that nobody will even know an upper bound on  $M$ .

Mathematics is full of examples where something is proved to exist, yet the proof tells us nothing about how to find it. Knowledge of the mere *existence* of an algorithm is completely different from the knowledge of an actual algorithm.

For example, RSA cryptography relies on the fact that one party knows the factors of a number, but the other party knows only that factors exist. Another example is that the game of  $N \times N$  Hex has a winning strategy for the first player, for all  $N$ . John Nash found a beautiful and extremely simple proof of this theorem in 1952. But Wikipedia tells me that such a strategy is still unknown when  $N = 9$ , despite many attempts. I can’t believe anyone will ever know it when  $N$  is 100.

More to the point, Robertson and Seymour have proved a famous theorem in graph theory: Any class  $c$  of graphs that is closed under taking minors has a finite number of minor-minimal graphs. (A minor of a graph is any graph obtainable by deleting vertices, deleting edges, or shrinking edges to a point. A minor-minimal graph  $H$  for  $c$  is a graph whose smaller minors all belong to  $c$  although  $H$  itself doesn’t.) Therefore there exists a polynomial-time algorithm to decide whether or not a given graph belongs to  $c$ : The algorithm checks that  $G$  doesn’t contain any of  $c$ ’s minor-minimal graphs as a minor.

But we don’t know what that algorithm is, except for a few special classes  $c$ , because the set of minor-minimal graphs is often unknown. The algorithm exists, but it’s not known to be discoverable in finite time.

This consequence of Robertson and Seymour’s theorem definitely surprised me, when I learned about it while reading a paper by Lovasz. And it tipped the balance, in my mind, toward the hypothesis that  $P = NP$ .

The moral is that people should distinguish between known (or knowable) polynomial-time algorithms and arbitrary polynomial-time algorithms. People might never be able to implement a poly-

nomial-time-worst-case algorithm for satisfiability, even though  $P$  happens to equal  $NP$ .

**18. Jeffrey O. Shallit, University of Waterloo:** Decision methods, automated theorem-proving, and proof assistants have been successful in a number of different areas: the Wilf-Zeilberger method for combinatorial identities and the Robbins conjecture, to name two. What do you think theorem discovery and proof will look like in 100 years? Rather like today, or much more automated?

**Don Knuth:** Besides economics, I was also afraid that somebody would ask me about the future, because I’m a notoriously bad prophet. I’ll take a shot at your question anyway.

Assuming 100 years of sustainable civilization, I’m fairly sure that a large percentage of theorems (maybe even 38.1966%) will be discovered with computer aid, and that a nontrivial percentage (maybe 0.7297%) will have computer-verified proofs that cannot be understood by mortals.

In my Ph.D. thesis (1963), I looked at computer-generated examples of small finite projective planes, and used that data to construct infinitely many planes of a kind never before known. Ten years later, I discovered the so-called Knuth-Morris-Pratt algorithm by studying the way one of Steve Cook’s automata was able to recognize concatenated palindromes in linear time. Such investigations are fun.

A few months ago, however, I tried unsuccessfully to do a similar thing. I had a 5,000-step mechanically discovered proof that the edges of a smallish flower snark graph cannot be 3-colored, and I wanted to psych out how the machine had come up with it. Although I gave up after a couple of days, I do think it would be possible to devise new tools for the study of computer proofs in order to identify the “aha moments” therein.

In February of this year I noticed that the calculation of an Erdős-discrepancy constant — made famous by Tim Gowers’ Polymath project, in which many mathematicians collaborated via the Internet — makes an instructive benchmark for satisfiability-testing algorithms. My first attempt to compute it needed 49 hours of computer time. Two weeks later I’d cut that down to less than 2 hours, but there still were 20 million steps in the proof. I see no way at present for human beings to understand more than the first few thousand of those steps.

**19. Scott Aaronson, MIT:** Would you recommend to other scientists to abandon the use of email, as you have done?

**Don Knuth:** My own situation is unusual, because I do my best work when I’m not interrupted. I eat,

sleep, and write content, more-or-less as a recluse who spends considerable time reading archives and other people's code. As I say on my home page (<http://www-cs-faculty.stanford.edu/~uno>), most people need to keep on top of things, but my role is to get to the bottom of things.

So I don't recommend a no-email policy to people who thrive on communication. And I actually take advantage of others in this respect (either shamelessly or shamefully, I'm not sure which), by pestering them with random questions, even though I don't want anybody to pester me — except about the one topic that I happen to be zooming in on at any particular time.

I do welcome email that reports bugs in *TAOCP*, because I always try to correct them as soon as possible.

Other unsolicited messages go to the bit bucket in the sky, otherwise known as `/dev/null`.

**20. J. H. Quick, blogger:** Why is this multi-interview called “twenty questions,” when only 19 questions were asked?

**Don Knuth:** I'm stumped. No, wait — Radia asked two.

Incidentally, the eVolumes of *TAOCP* contain some 4,500 questions, and almost as many answers.

— \* —

### The panel

1. Jon Bentley, author of “Programming Pearls” in *Communications of the ACM*.
2. Dave Walden, TUG board member and coordinator of the TUG Interview Corner.
3. Charles Leiserson, MIT; theory of parallel computing and distributed computing, and the practical applications thereof.
4. Dennis Shasha, NYU; biological computing, pattern recognition, and machine learning.
5. Mark Taub, Pearson.
6. Radia Perlman, Intel; software designer and network engineer.
7. Tony Gaddis, author of computer science books.
8. Robert Sedgwick, Princeton; analysis of algorithms; one of Don Knuth's Ph.D. students.
9. Barbara Steele, contributor to *Common Lisp: The Language*.
10. Silvio Levy, co-author with Don Knuth of *The CWEB System of Literate Programming*, and with Raymond Seroul of *A Beginner's Book of T<sub>E</sub>X*; professional goal: to further the communication of mathematics.

11. Peter Gordon, Don Knuth's editor at Addison-Wesley from the early 1980s until his retirement in 2014; see his TUG interview at <http://tug.org/interviews/gordon.html>.
12. Udi Manber, a vice president of engineering at Google, responsible for search products.
13. Al Aho, Columbia University; programming languages, compilers and related algorithms, and prolific author of textbooks on the art and science of computer programming; co-author of the AWK programming language.
14. Guy Steele, designer and writer of numerous programming language specifications, including the original command set of Emacs; the first person to port T<sub>E</sub>X (from WAITS to ITS).
15. Robert Tarjan, Princeton; known for his pioneering work on graph theory algorithms and data structures; his dissertation was supervised by Don Knuth.
16. Frank Ruskey, University of Victoria; research includes algorithms for exhaustively listing discrete structures, and various combinatorial topics.
17. Andrew Binstock, Editor-in-Chief, *Dr. Dobbs's Journal*.
18. Jeffrey Outlaw Shallit, University of Waterloo; combinatorics on words, formal languages, automata theory, and algorithmic number theory; also Vice President of Electronic Frontier Canada.
19. Scott Aaronson, MIT; theory of computational complexity and quantum computing.
20. J. H. Quick, blogger.

— \* —

We conclude with another quote from

**Radia Perlman**, regarding how *TAOCP* affected her, which also nicely expresses how many of us T<sub>E</sub>X users feel about *Computers & Typesetting*:

Having the books on my bookshelf gave me a sense of security . . . that pretty much anything I'd wonder about would be explained there. Today Wikipedia serves some of that purpose. It would have been nice 20 years ago to have had a (more) portable version of Knuth so that I could know, wherever I was, that I could quickly look something up. But 20 years ago there was nothing else, so I'd have to wait until I was back at home to consult the copy in my bedroom bookshelves, or wander the halls at work to find someone who had a copy in their office. I did actually have a 2nd copy that was supposed to be at work, but it was always being “borrowed,” so I could never find my own copy at work when I needed it. ■

## A letter on the persistence of (e)books

Charles Bigelow

[Originally sent fall 2011;  
slightly edited for *TUGboat*, summer 2014.]

Dear Colleagues,

I have an original Amazon Kindle, bought around 2008. I also have a Barnes & Noble Nook, from around 2010, and an early Sony Reader a bit older than the Kindle.

My graduate student was using them for his research. He gave them back to me this past summer after he finished his thesis (*Congeniality of Reading on Digital Devices*, RIT 2011).

I set them on a shelf and didn't think about them until last week, when I got them out to show to my book design class in the fall (2011).

The Kindle was totally dead. Wouldn't light up, wouldn't charge, wouldn't run even when plugged in. The Nook wouldn't charge but would work when plugged in, until yesterday, when it wouldn't work at all. I took it to Barnes & Noble, where a staff person changed the battery, to no avail, and then called B&N tech support, who advised removing the battery, letting the thing sit for half an hour, and then recharging for four hours, to see if that would reboot the battery.

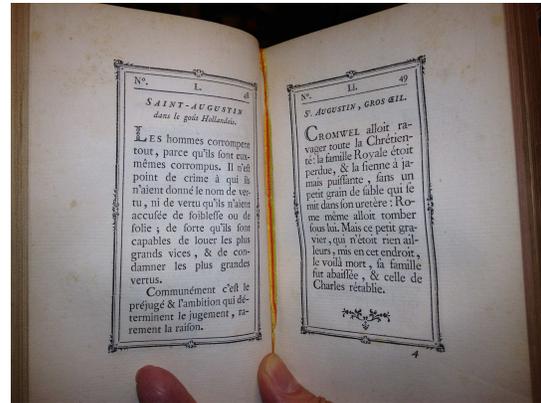
I took it home and plugged it in, with the power cord I almost left in the store, forgetting that an e-book has to have cords and connectors. The Nook connector doesn't fit my Kindle, so I have to keep track of which e-reader has which connector. But, that's not a problem now, because the Kindle is beyond recharge anyway, and so is the Nook, which never did revive.

The Sony Reader was hopeless whether or not it took a charge, because it required hooking up to a Windows PC to download books, but I use a Macintosh. And anyway, I'd misplaced the cables and connectors. Hard to keep track of those things.

At home, I went over to my bookshelf and took down a beautiful little book printed in Paris in 1764–1766 by Pierre-Simon Fournier.

It was still in excellent working condition, even though it hadn't been charged in 247 years. I opened it, and immediately it started up! I could read everything, without connectors or cables.

It still works today. The print on the pages is a rich, dark black with good contrast against a pale creamy background. The type is elegant, cut by P.-S. Fournier himself, the greatest type designer of his era. The pages have a slight texture with a good feel, making them easy to turn with a simple swipe of the fingers. It has generous margins so you can hold the book open without covering the text with your thumbs—a brilliant feature of the user interface. It has many illustrations in vivid black and white, including fold-out pages with



Fournier's *Manuel Typographique*, hand-held. Built-in bookmark tape visible along gutter. St. Augustin size (14 point modern measure, about the size of menus on Macs). Page L (50) uses St. Augustin in the Dutch style (big x-height, narrow letters, light weight); page LI (51) also St. Augustin.

landscape format images, and other fold-outs with music. Its casing is a rich, dark red leather with gold tooling and ornamentation and colorful marbling, also on the inside covers. The page edges are gilded, so the whole book glows with a faint aura even before it is opened. It is small enough to be held with one hand, but works with two hands just as well.

I won't strain credulity by pretending the book is perfect. There are two volumes, so you have to keep track of both. The search function is primitive: you must browse or skim; and if you want to capture text for later reference, you have to remember it or copy it by hand, and if you want to link to a page, you have to use a bookmark (this book comes with its own bound-in, bookmark tape movable to the page of your choice).

These reading exertions put a severe strain on my brain, to be sure. Sometimes I have to eat some chocolate to re-charge, but I have to be careful not to get chocolate on the pages, because it won't wash off. Some of the pages are slightly spotted by mold or foxing (not from chocolate but from centuries of humidity and slow chemical changes in the paper), yet the text is still readable. The type can't be resized, a problem for many of us over the age of 40, so to read the smallest size specimen, which Fournier called "Parisienne", a gem-like cutting at about 5 point body size in modern type measure, I have to use bifocals or a reading glass.

Oh, and this particular book is in French, which slows down my reading, but well, when I'm reading this book, I'm not in much of a hurry anyway.

◇ Charles Bigelow  
<http://www.lucidafonts.com>

## L<sup>A</sup>T<sub>E</sub>X document class options

Thomas Thurnherr

### Abstract

The standard document classes `article`, `report`, `book`, and `letter` accept a number of class options which allow high-level customization of a document. In this article, available options are introduced, the default for each document class is highlighted, and alternative, more flexible customizations are given.

### 1 Setting document class options

Options that differ from the default are passed to the document class through its optional argument field. Multiple options have to be separated by a comma. If contradictory options are set, the last option always overrides the previous ones. Moreover, if a non-existent option is set, L<sup>A</sup>T<sub>E</sub>X ignores it and generates a warning in the log.

```
\documentclass[option1,option2,...]{article}
```

### 2 Default options

Most default options are the same between different document classes, with a few exceptions. An overview of all the defaults is given in table 1 (below). As the `letter` class is fairly specific, several options don't apply and are therefore not implemented.

### 3 Paper size

L<sup>A</sup>T<sub>E</sub>X provides several predefined paper (page) sizes. The supported options: `a4paper`, `a5paper`, `b5paper`, `letterpaper`, `legalpaper`, and `executivepaper`.

The width and height for each of these page sizes is listed in table 2. The default depends on the T<sub>E</sub>X distribution and/or system used. It is either `a4paper` or `letterpaper`.

The `geometry` package [3] implements additional page sizes. For example, with this package, all ISO standard formats are available, including ISO A0–A6, B0–B6, and C0–C6, specified as `a0paper`–`a6paper`,

**Table 2:** Measures of predefined page formats.

<i>Option</i>	<i>width</i>	<i>height</i>
<code>a4paper</code>	210 mm	297 mm
<code>a5paper</code>	148 mm	210 mm
<code>b5paper</code>	176 mm	250 mm
<code>letterpaper</code>	8.5 in	11 in
<code>legalpaper</code>	8.5 in	14 in
<code>executivepaper</code>	7.25 in	10.5 in

and so on. To use a geometry page format, the option is passed to the package directly, rather than to the document class. Any page format set in the document class is ignored. Besides these additional predefined formats, the package allows the user to define an arbitrary page size. Here is an example:

```
% A0 size:
\usepackage[a0paper]{geometry}

% Arbitrary page size:
\usepackage[paperwidth=5cm,paperheight=5cm]
{geometry}
```

### 4 Font size

Throughout the entire document, L<sup>A</sup>T<sub>E</sub>X uses the same font size, except for headings or if the font is changed locally through a macro, such as `\small` or `\large`. 10pt is the default for all classes. Three options are available: `10pt`, `11pt`, and `12pt`. If the default font size is changed, headings and macros change accordingly. Margins are also changed according to the font size.

For a larger range, the `extsizes` package [2] provides additional classes that support font sizes between 8pt–20pt.

### 5 One or two columns

All document classes use a single column layout by default (`onecolumn`). With the `twocolumn` option,

**Table 1:** Default document class option for standard document classes.

<i>Option</i>	<code>article</code>	<code>report</code>	<code>book</code>	<code>letter</code>
Paper size (system specific)	<code>a4paper/letterpaper</code>	<code>a4paper/letterpaper</code>	<code>a4paper/letterpaper</code>	<code>a4paper/letterpaper</code>
Font size	10pt	10pt	10pt	10pt
Number of columns	<code>onecolumn</code>	<code>onecolumn</code>	<code>onecolumn</code>	<code>onecolumn</code>
Margins	<code>oneside</code>	<code>oneside</code>	<code>twoside</code>	<code>oneside</code>
Title page	<code>notitlepage</code>	<code>titlepage</code>	<code>titlepage</code>	-
Chapter start page	-	<code>openany</code>	<code>openright</code>	-
Orientation	<code>portrait</code>	<code>portrait</code>	<code>portrait</code>	<code>portrait</code>
Formula options	(center; right label)	-	-	-
Draft or final	<code>final</code>	<code>final</code>	<code>final</code>	<code>final</code>

the page is horizontally divided, a layout frequently used by scientific journals. The `\linewidth` macro flexibly adapts to the new layout and is automatically set to the width of a single column. Therefore, `\linewidth` is convenient to make optimal use of the available space, for example when adding figures. `\textwidth`, on the other hand, remains unchanged and is equal to the total width of the text area.

In two-column mode, the `figure*` environment inserts a figure that spans both columns, and similarly `table*` for a full-width table. Consequently, `\linewidth` and `\textwidth` are identical within these starred environments. An example:

```
\documentclass[twocolumn]{article}
\usepackage{graphicx}
\begin{document}

\begin{figure*}[ht]
  \includegraphics[width=\linewidth]{myFigure}
  \caption{Figure spanning two columns.}
\end{figure*}

... text of document ...
\end{document}
```

The `multicol` package [9] provides support for two or more columns. With this package, it is also possible to mix different layouts within the same document.

## 6 Margins

The options `oneside` and `twoside` affect the width of the side margins. With `oneside`, which is the default for `article`, `report`, and `letter`, the margins on both sides of every page are equally wide. With `twoside`, L<sup>A</sup>T<sub>E</sub>X distinguishes between an inner and outer margin. The outer margin is substantially wider and switches between left and right. Even pages have their outer margin on the left, odd pages on the right. Most books follow this structure and so it should not come as a surprise that the `book` class default is `twoside`.

## 7 Title page

The `titlepage` option prints the title on a separate page. This is the default for `report` and `book`. On the other hand, `article` has `notitlepage` as its default, with the main text starting directly after the title. The `letter` class doesn't implement title page commands and therefore these options are altogether unavailable.

## 8 Page orientation

All of the standard document classes produce documents in portrait orientation, by default. The option

`portrait` doesn't explicitly exist. However, there is a `landscape` option, which rotates the page by 90°, but keeps the dimensions of the text area and the margins, which is often undesired. The `geometry` package [3] provides a more convenient landscape option, where text area and margins are adapted accordingly.

```
\usepackage[landscape]{geometry}
```

The `lscap` [7] and `pdfscape` [10] packages implement the `landscape` environment, which changes the orientation locally, for one or several pages in an otherwise portrait document. In contrast to the `geometry` package, with these packages only the orientation of the text area is changed, while the margins and with them the header and footer remain in portrait mode. This environment is particularly useful for adding extra-wide figures or tables to a document. If pdf<sub>T</sub>E<sub>X</sub> is used for processing, `pdfscape` physically rotates any landscape oriented page, which makes it easier to read on screen. For example:

```
\documentclass{article}
\usepackage{pdfscape}
\begin{document}
\begin{landscape}
  % landscape oriented content
\end{landscape}
\end{document}
```

## 9 Chapter starting page

Chapters and other chapter-level headings are only available in the `report` and `book` classes. By default, a new chapter starts on the next page in `report` (`openany`), but always on an odd page in `book` (`openright`). As a consequence, in a `book` there might be a blank page between two consecutive chapters (if the previous chapter ended with an odd page number). `openany` and `openright` do not apply to `article` or `letter`.

## 10 Formula options `fleqn` and `leqno`

The `fleqn` and `leqno` options define how formulas are displayed. They are independent and so can be used together. The names are not especially self-explanatory — `fleqn` aligns formulas on the left, instead of the default centering; `leqno` prints the equation number on the left side instead of the (default) right.

For instance, consider the Cauchy-Schwartz inequality printed with the defaults: the formula is centered, with the equation number on the right.

$$|x, y|^2 \leq \langle x, x \rangle \cdot \langle y, y \rangle \quad (1)$$

With `fleqn`, the equation is left-aligned:

$$|x, y|^2 \leq \langle x, x \rangle \cdot \langle y, y \rangle \quad (2)$$

And with `leqno`, the equation number is placed left of the equation instead of right:

$$(3) \quad |x, y|^2 \leq \langle x, x \rangle \cdot \langle y, y \rangle$$

The `amsmath` package [1] provides more flexibility for equations. For example, it implements the `flalign` environment which was used here to illustrate left-alignment (equation 2).

## 11 Draft or final

All document classes have the `final` option preset. With `draft`, text or environments that reach into the margins are highlighted with a black square or bar. With that, it becomes easy to spot `Overfull \hbox` warnings in the document output.

Other packages also make use of these options and implement macros that behave differently in draft mode. For example, the `graphicx` bundle [4] replaces figures with a box that shows the file name instead of the figure. Document processing time can be drastically reduced when figures are not loaded. Two other examples: the `hyperref` [5] package removes all linking features from a document in draft mode, and `microtype` [8] disables its features altogether.

When `draft` is used for the overall document, specific packages can be still set to final mode by loading the package with the `final` option. This might sometimes be helpful to examine the package’s “final” behavior. An example:

```
\documentclass[draft]{article}
\usepackage[final]{graphicx}
```

The `ifdraft` package [6] implements commands to flexibly customize the behavior of `draft` and/or `final`. For example, in a thesis the author might like to omit the title page and content lists while he’s still working on the document. This is straightforward, using either the `\ifdraft` or `\iffinal` macro provided by the package:

```
\documentclass[draft]{report}
\usepackage{ifdraft}
\title{...}
\author{...}
\begin{document}
```

```
\ifdraft{% Draft: omit title/toc/lof/lot
}%
\maketitle
\tableofcontents\clearpage
\listoffigures\clearpage
\listoftables\clearpage
}
\end{document}
```

## References

- [1] `amsmath` — AMS mathematical facilities for L<sup>A</sup>T<sub>E</sub>X. <http://www.ctan.org/pkg/amsmath>. Accessed: 2014-09-22.
- [2] `extsizes` — extend the standard classes’ size options. <http://www.ctan.org/pkg/extsizes>. Accessed: 2014-09-30.
- [3] `geometry` — flexible and complete interface to document dimensions. <http://www.ctan.org/pkg/geometry>. Accessed: 2014-09-22.
- [4] `graphicx` — enhanced support for graphics. <http://www.ctan.org/pkg/graphicx>. Accessed: 2014-09-28.
- [5] `hyperref` — extensive support for hypertext in L<sup>A</sup>T<sub>E</sub>X. <http://www.ctan.org/pkg/hyperref>. Accessed: 2014-09-28.
- [6] `ifdraft` — detect “draft” and “final” class options. <http://www.ctan.org/pkg/ifdraft>. Accessed: 2014-09-28.
- [7] `lscap` — place selected parts of a document in landscape. <http://www.ctan.org/pkg/lscap>. Accessed: 2014-09-30.
- [8] `microtype` — subliminal refinements towards typographical perfection. <http://www.ctan.org/pkg/microtype>. Accessed: 2014-09-28.
- [9] `multicol` — intermix single and multiple columns. <http://www.ctan.org/pkg/multicol>. Accessed: 2014-09-28.
- [10] `pdflscape` — make landscape pages display as landscape. <http://www.ctan.org/pkg/pdflscape>. Accessed: 2014-09-30.

◇ Thomas Thurnherr  
 texblog (at) gmail dot com  
<http://texblog.org>

---

## How to influence the position of float environments like figure and table in L<sup>A</sup>T<sub>E</sub>X?

Frank Mittelbach

### Abstract

In 2012, a question “How to influence the float placement in L<sup>A</sup>T<sub>E</sub>X” was asked on TeX.stackexchange [3] and as there had been many earlier questions around this topic I decided to treat the topic in some depth and explain most of the mysteries that the underlying mechanism poses to people trying to use it successfully.

Once my answer appeared on the web, people asked to see this converted into an article and I foolishly replied “only if this answer ends up becoming a ‘great’ answer” (gets 100 votes). At the time of writing this article, the answer stands at 222 votes, so I had better make good on that promise.

### Contents

1	Introduction	248
2	L <sup>A</sup> T <sub>E</sub> X floats terminology	248
2.1	Float classes . . . . .	248
2.2	Float areas . . . . .	248
2.3	Float placement specifiers . . . . .	248
2.4	Float algorithm parameters . . . . .	249
2.5	Float reference point . . . . .	249
3	Basic behavioral rules of L <sup>A</sup> T <sub>E</sub> X’s float mechanism	249
3.1	The basic sequence . . . . .	249
3.2	Detailed placement rules . . . . .	250
3.3	Emptying the holding queue at the column or page boundary . . . . .	250
3.4	Parameters influencing the placement	250
4	Consequences of the algorithm	251
4.1	A float may appear in the document earlier than its location in the source	251
4.2	Double-column floats are always deferred first . . . . .	251
4.3	There is no bottom float area for double-column floats . . . . .	252
4.4	All float parameters (normally) restrict the placement possibilities . . . . .	252
4.5	“Here” just means “here if it fits” . . . . .	252
4.6	Float specifiers do not define an order of preference . . . . .	252
4.7	Relation of floats and footnotes . . . . .	252
5	Documentation of the algorithm	253
6	How to address specific issues	253

Frank Mittelbach

6.1	Ensure that floats appear “here” . . . . .	253
6.2	Provide a bottom float area for two-column floats . . . . .	253
6.3	Ensure that floats are always placed after their call-out . . . . .	253
6.4	Prevent floats on certain pages . . . . .	254
6.5	Implement float barriers . . . . .	254
6.6	Overwrite placement restrictions . . . . .	254
6.7	Final tuning advice . . . . .	254

## 1 Introduction

To answer this question one first has to understand the basic rules that govern L<sup>A</sup>T<sub>E</sub>X’s standard placement of floats. Once these are understood, adjustments can be made, for example, by modifying float parameters, or by adding certain packages that modify or extend the basic functionality.

## 2 L<sup>A</sup>T<sub>E</sub>X floats terminology

### 2.1 Float classes

Each float in L<sup>A</sup>T<sub>E</sub>X belongs to a class. By default, L<sup>A</sup>T<sub>E</sub>X knows about two classes, viz., figure and table. Further classes can be added by a document class or by packages. The class a float belongs to influences certain aspects of the float positioning, such as its default placement specification (if not overridden on the float itself).

One important property of the float placement algorithm is that L<sup>A</sup>T<sub>E</sub>X never violates the order of placement within a class of floats. E.g., if you have figure 1, table 1, figure 2 in a document, then figure 1 will always be placed before figure 2. However, table 1 (belonging to a different float class) will be placed independently and hence can appear before, after, or between the figures.

### 2.2 Float areas

L<sup>A</sup>T<sub>E</sub>X knows about two float areas within a column where it can place floats: the top area and the bottom area of the column. In two-column layout, it also knows about a top area spanning the two columns. There is no bottom area for page-wide floats in two-column mode.

In addition, L<sup>A</sup>T<sub>E</sub>X can make float columns and float pages, i.e., columns or pages which contain only floats. Finally, L<sup>A</sup>T<sub>E</sub>X can place floats in-line into the text (but only if so directed on the individual float).

### 2.3 Float placement specifiers

To direct a float to be placed into one of these areas, a float placement specifier can be provided as an optional argument to the float. If no such optional argument is given then a default placement specifier is used (which depends on the float class as mentioned

above but usually allows the float to be placed in all areas if not subject to other restrictions).

A float placement specifier can consist of the following characters in *any* order:

- ! indicates that some of the restrictions that normally apply should be ignored (discussed later)
- h indicates that the float is allowed to be placed in-line (“here”)
- t indicates that the float is allowed to go into a top area
- b indicates that the float is allowed to go into a bottom area
- p indicates that the float is allowed to go on a float page or column area

The order in which these characters are put in the optional argument does *not* influence how the algorithm tries to place the float! The precise order is discussed in section 3.2. This is one of the common misunderstandings, for instance when people think that `bt` means that the bottom area should be tried first.

However, if a letter is not present then the corresponding area will not be tried at all.

## 2.4 Float algorithm parameters

There are about 20 parameters that influence the placement. Basically they define

- how many floats can go into a certain area,
- how big a float area can become,
- how much text there has to be on a page (in other words, how much the top and bottom float areas can occupy), and
- how much space will be inserted
  - between consecutive floats in an area and
  - between the float area and the text above or below it.

## 2.5 Float reference point

A point in the document that references the float (e.g., “see figure X”) is called a “call-out” and the float body should be placed close to the (main) call-out, as its placement in the document affects the placement of the float in the output, because it determines when  $\LaTeX$  sees the float for the first time. It’s important to understand that if a float is placed in the middle of a paragraph, the reference point for the algorithm is the next line break, or page break, in the paragraph that follows the actual placement in the source.

For technical and practical reasons it is usually best to place all floats between paragraphs (i.e., after the paragraph with the call-out), even if that makes the call-out and reference point slightly disagree.

## 3 Basic behavioral rules of $\LaTeX$ ’s float mechanism

With this knowledge, we are now ready to delve into the algorithm’s behavior.

First we have to understand that all of  $\LaTeX$ ’s typesetting algorithms are designed to avoid any sort of backtracking. This means that  $\LaTeX$  reads through the document source, formats what it finds and (more or less) immediately typesets it. The reasons for this design choice were to limit complexity (which is still quite high) and also to maintain reasonable speed (remember that this is from the early eighties).

For floats, this means that the algorithm is greedy, i.e., the moment it encounters a float it will immediately try to place it and, if it succeeds, it will never change its decision. This means that it may choose a solution that could be deemed inferior in light of data received later on.

For example, if a figure is allowed to go to the top or bottom area,  $\LaTeX$  may decide to place this figure in the top area. If this figure is followed by two tables which are only allowed to go to the top, these tables may not fit anymore. A solution that could have worked in this case (but wasn’t tried) would have been to place the figure in the bottom area and the two tables in the top area.

### 3.1 The basic sequence

So here is the basic sequence the algorithm runs through:

- If a float is encountered,  $\LaTeX$  attempts to place it immediately according to its rules (detailed later);
- if this succeeds, the float is placed and that decision is never changed;
- if this does not succeed, then  $\LaTeX$  places the float into a holding queue to be reconsidered when the next page is started (but not earlier).
- Once a page has finished,  $\LaTeX$  examines this holding queue and tries to empty it as best as possible. For this it will first try to generate as many float pages as possible (in the hope of getting floats off the queue). Once this possibility is exhausted, it will next try to place the remaining floats into top and bottom areas. It looks at all the remaining floats and either places them or defers them to a later page (i.e., adding them once more to the holding queue).
- After that, it starts processing document material for this page. In the process, it may encounter further floats.
- If the end of the document has been reached or if a `\clearpage` is encountered,  $\LaTeX$  starts a

How to influence the position of float environments like figure and table in  $\LaTeX$ ?

new page, relaxes all restrictive float conditions, and outputs all floats in the holding queue by placing them on float page(s).

In two-column mode the same algorithm is used, except that it works on the level of columns, e.g., when a column has finished  $\LaTeX$  will look at the holding queue and generate float columns, etc.

### 3.2 Detailed placement rules

Whenever  $\LaTeX$  encounters a float environment in the source, it will first look at the holding queue to check if there is already a float of the same class in the queue. If that happens to be the case, no placement is allowed and the float immediately goes into the holding queue.

If not,  $\LaTeX$  looks at the float placement specifier for this float, either the explicit one in the optional argument or the default one from the float class. The default per float class is set in the document class file (e.g., `article.cls`) and very often resolves to `tbp`, but this is not guaranteed.

- If the specifier contains a `!`, the algorithm will ignore any restrictions related either to the number of floats that can be put into an area or the maximum size an area can occupy. Otherwise the restrictions defined by the parameters apply.
- As a next step it will check if `h` has been specified.
- If so, it will try to place the float right where it was encountered. If this works, i.e., if there is enough space, then it will be placed and processing of that float ends.
- If not, it will look next for `t` and if that has been specified the algorithm will try to place the float in the top area. If there is no other restriction that prevents this, then the float is placed there and float processing stops.
- If not it will finally check if `b` is present and, if so, it will try to place the float into the bottom area (again obeying any restrictions that apply if `!` wasn't given).
- If that doesn't work either or is not permitted because the specifier wasn't given, the float is added to the holding queue.
- A `p` specifier (if present) is not used during the above process. It will only be looked at when the holding queue is being emptied at the next page or column boundary.

This ends the processing when encountering a float in the document.

### 3.3 Emptying the holding queue at the column or page boundary

After a column or page has been finished,  $\LaTeX$  looks at the holding queue and attempts to empty

it out as best as possible. For this it will first try to build float pages.<sup>1</sup>

Any floats participating in a float page (or column) must have a `p` as a float specifier in its float placement specification. If not, the float cannot go on a float page and, in addition, will also prevent any further deferred float of the same class from being placed onto the float page!

If the float can go there, it will be marked for inclusion on the float page, but the processor may still abort the attempt if the float page will not get filled “enough” (depending on the parameter settings for float pages). Only at the very end of the document, or when a `\clearpage` has been issued, are these restrictions lifted, and a float will then be placed on a float page even if it has no `p` and would be the only float on that page.

Creation of float pages continues until the algorithm has no further floats to place or when it fails to produce a float page due to parameter settings. In the latter case, all floats that have not been placed so far, are then considered for inclusion in the top and bottom areas of the next page (or column).

The process there is the same as the one described above, except that

- the `h` specifier no longer has any meaning (as we are, by now, far away from the original “here”) and is therefore ignored,
- and the floats at this time are not coming from the source document but are taken one after the other from the holding queue.

Any float that couldn't be placed is then put back into the holding queue, so that when  $\LaTeX$  is ready to look at further textual input from the document the holding queue may already contain floats. A consequence of this is that a float encountered in the document may immediately get deferred just because an earlier float of the same float class is already on hold.

### 3.4 Parameters influencing the placement

There are four counters that control how many floats can go into areas:

`totalnumber` (default 3) is the maximum number of floats on a text column; it is not used for float pages;

`topnumber` (default 2) is the maximum number of floats in the top area;

`bottomnumber` (default 1) is the maximum number of floats in the bottom area;

---

<sup>1</sup> In two-column mode  $\LaTeX$  will build float columns (when finishing a column) and also attempt to generate float pages when finishing a page. In the remainder of the article “float page” will denote either depending on the context.

`dbltopnumber` (default 2) is the maximum number of full-width floats in two-column mode going above the text columns.

The size of the areas are controlled through parameters (to be changed with `\renewcommand`) that define the maximum (or minimum) size of the area, expressed as a fraction of the page height:

`\topfraction` (default 0.7) maximum size of the top area

`\bottomfraction` (default 0.3) maximum size of the bottom area

`\dbltopfraction` (default 0.7) maximum size of the top area for double-column floats

`\textfraction` (default 0.2) minimum size of the text area, i.e., the area that must not be occupied by floats

The space that separates floats within an area, as well as between float areas and text areas, is defined through the following parameters (all of which are rubber lengths, i.e., can contain some stretch or shrink components). Their defaults depend on the document font size and change when class options like 11pt or 12pt are used. We show only the 10pt defaults:

`\floatsep` (default 12pt plus 2pt minus 2pt) the separation between floats in top or bottom areas

`\dblfloatsep` (default 12pt plus 2pt minus 2pt) the separation between double-column floats on two-column pages

`\textfloatsep` (default 20pt plus 2pt minus 4pt) the separation between top or bottom float area and the text area

`\dbltextfloatsep` (default 20pt plus 2pt minus 4pt) the analog of `\textfloatsep` for two-column floats

For in-line floats (that have been placed “here”) the separation to the surrounding text is controlled by

`\intertextsep` (default 12pt plus 2pt minus 2pt)

In the case of float pages or float columns (i.e., a page or a column of a page containing only floats) parameters like `\topfraction` etc. do not apply. Instead the creation of them is controlled through

`\floatpagefraction` (default 0.5) minimum part of the page (or column) that needs to be occupied by floats to be allowed to form a float page (or column).

## 4 Consequences of the algorithm

### 4.1 A float may appear in the document earlier than its location in the source

The placement of the float environment in the source determines the earliest point where it can appear in

the final document. It may move visually backward to some degree as it may be placed in the top area on the current page; see section 6.3 on how to change this. It can, however, not end up on an earlier page than the surrounding text due to the fact that  $\text{\LaTeX}$  does no backtracking and the earlier pages have already been typeset.

Thus normally a float is placed in the source near its first call-out (i.e., text like “see figure 5”) because this will ensure that the float appears either on the same page as this text or on a later page. However, in some situations you may want to place a float on the preceding page (if that page is still visible from the call-out). This is possible only by moving the float to an earlier position in the source.

### 4.2 Double-column floats are always deferred first

When  $\text{\LaTeX}$  encounters a page-wide float environment (indicated by a `*` at the end of the environment name, e.g., `figure*`) in two-column mode, it immediately moves that float to the deferred queue. The reason for this behavior again lies in the “greedy” behavior of its algorithm: if  $\text{\LaTeX}$  is currently assembling the second column of that page, the first column has already been assembled and stored away; recall that because  $\text{\LaTeX}$  does not backtrack there is no way to fit the float on the current page. To keep the algorithm simple, it does the same even if working on the first column (where it could in theory do better even without backtracking).

Thus, in order to place such a float onto the current page, one has to manually move it to an earlier place in the source — before the start of the current page. If this is done, obviously any further change in the document could make this adjustment obsolete; hence, such adjustments are best done (if at all) only at the very last stage of document production — when all material has been written and the focus is on fine-tuning the visual appearance.

Also note that the base algorithm has a bug<sup>2</sup> in this area: it maintains two independent holding queues: one for single-column and one for double-column floats. As a result the float order is not necessarily preserved and floats may get typeset out of sequence. If this happens one either has to manually move the double-column float to an earlier (or later) place in the document or load the `fixltx2e` package that implements a correction for this issue.

<sup>2</sup> As this is the documented behavior in the  $\text{\LaTeX}$  manual [1] it is perhaps more correctly called an undesired feature than a bug.

### 4.3 There is no bottom float area for double-column floats

This isn't so much a consequence of the algorithm but rather a fact about its implementation. For double-column floats the only possible placements offered are the top area or a float page. Thus if somebody adds an `h` or a `b` float placement specifier to such a float it simply gets ignored. As a special important case `{figure*}[b]` implies that this float will not get typeset at all until either a `\clearpage` is encountered or the end of the document is reached.

### 4.4 All float parameters (normally) restrict the placement possibilities

This may be obvious but it is worth repeating: any float parameter defines a restriction on L<sup>A</sup>T<sub>E</sub>X's ability to place the floats. How much of a restriction depends on the setting: there is always a way to set a parameter in such a way that it does not affect the placement at all. Unfortunately, in doing so one invites rather poor-looking placements.

By default L<sup>A</sup>T<sub>E</sub>X has settings that are fairly liberal. For example, for a float page to be accepted the float(s) must occupy at least half of the available page. Expressed differently, this means that such a page is allowed to be half empty (which is certainly not the best possible placement in most cases).

What often happens is that users try to improve such settings and then get surprised when suddenly all floats pile up at the end of the document. To stay with this example: if one changes the parameter `\floatpagefraction` to require, say, 0.8 of the float page, a float that occupies about 0.75 of the page will not be allowed to form a float page on its own. Thus, if there isn't another float that could be added and actually fits in the remaining space, the float will get deferred and with it all other floats of the same class. But, even worse, this specific float is too big to go into the next top area as well because there the default maximum permissible area is 0.7 (from `\topfraction`). As a result all your floats stay deferred until the next `\clearpage`.

For this reason it is best not to meddle with the parameters while writing a document or at least not to do so in a way that makes it more difficult for the algorithm to place a float close to its call-out. For proof-reading it is far more important to have a figure next to the place it is referenced than to avoid half-empty pages. Possibilities for fine-tuning an otherwise finished document are discussed below.

Another conclusion to draw here is that there are dependencies between some of the float parameters; it is important to take these dependencies into account when changing their values.

### 4.5 “Here” just means “here if it fits”

... and often it doesn't fit. This is somewhat surprising for many people, but the way the algorithm has been designed the `h` specifier is not an unconditional command. If an unconditional command is needed, extension packages such as the `float` package offer `H` as an alternative specifier that really means “here” (and starts a new page first if necessary).

### 4.6 Float specifiers do not define an order of preference

As mentioned above, the algorithm tries to place floats into available float areas in a well-defined order that is hard-wired into the algorithm: “here”, “top”, “bottom” and —on page boundaries— first “page” and only if that is no longer possible, “top” followed by “bottom” for the next page.

Thus specifying `[bt]` does not mean try bottom first and only then top. It simply means allow this float to go into top or bottom area (but not onto a float page) just like `[tb]` would.

### 4.7 Relation of floats and footnotes

This is not exactly a consequence of the algorithm but one of its implementation: Whenever L<sup>A</sup>T<sub>E</sub>X tries to decide on a placement for a float (or a `\marginpar`!) it has to trigger the output routine to do this. And as part of this process all footnotes on the page are removed from their current place in the galley and are collected together in the `\footins` box as part of T<sub>E</sub>X's preparation for page production.

But after placing the float (or deferring it) L<sup>A</sup>T<sub>E</sub>X then returns the page material to the galley, and because of T<sub>E</sub>X's output routine behavior the galley has now changed: all the footnotes have been taken out from their original places. So L<sup>A</sup>T<sub>E</sub>X has to put the footnotes back, but it can only place them in a single place (not knowing the origin anymore). What it does is reinsert the footnotes (the footnote text to be precise) at the end of the galley. There are some good reasons for doing this, one of which is that L<sup>A</sup>T<sub>E</sub>X expects that all of the returned material still fits on the current page.

However, if for some reason a page break is finally taken at an earlier point than the footnotes will show up on the wrong page or column. This is a fairly unlikely scenario and L<sup>A</sup>T<sub>E</sub>X works hard at making it a near-impossibility, but if it happens check if there is a float near the chosen page break and either move the float or guide the algorithm by using explicit page breaks. An example of this behavior can be found in another question on [TeX.stackexchange](http://TeX.stackexchange.com) [4]. In fact the particular case discussed in the question is worth highlighting: Do *not* place a float directly

after a heading, unless it is a heading that always starts a page. The reason is that headings normally form very large objects (as a heading prevents a page break directly after it). However placing a float in the middle of this means that the output routine gets triggered before L<sup>A</sup>T<sub>E</sub>X makes its decision where to break and any footnotes get moved into the wrong place.

## 5 Documentation of the algorithm

As requested, here is some information on existing documentation. The algorithm and its implementation are documented in the file `l1output.dtx` as part of the L<sup>A</sup>T<sub>E</sub>X kernel source. This can be typeset standalone or as part of the whole kernel (i.e., by typesetting `source2e.tex` — ignore the checksum error if it is still there,<sup>3</sup> sorry).

This documentation is an interesting historical artifact. Parts of it show semi-formatted pseudo-code which dates back to L<sup>A</sup>T<sub>E</sub>X 2.09; in other words it is from the original documentation by Leslie Lamport. The actual code is documented using doc style and in parts is more or less properly documented (from scratch) and dates back to 1994 or thereabouts when Chris Rowley and myself adjusted and extended the original algorithm for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> (the current version). It also fairly openly documents the various issues with the algorithm and/or its implementation — in many cases we didn’t dare to alter it because of the many dependencies and, of course, because of the danger to screw up too many existing documents that implicitly rely on the current behavior for good or ill.<sup>4</sup> Near the end you’ll find a list of comments compiled on the algorithm back then, but there are also comments, questions, and tasks (?:-) sprinkled throughout the documentation of the code.

One interesting aspect of this file (that I forgot all about) is that it contains all the code necessary to trace the behavior of the algorithm in real life. It is fairly raw and detailed output and probably for that reason I didn’t make this publicly available back then. But even in its current form it does give some interesting insight into the behavior of the algorithm and how certain decisions come about.

Thus while writing this article I had second thoughts and now the most recent distribution of L<sup>A</sup>T<sub>E</sub>X (May 2014)<sup>5</sup> offers the package `fltrace` that you

<sup>3</sup> But this also means you are running an older release of L<sup>A</sup>T<sub>E</sub>X.

<sup>4</sup> This is, for example, the reason that the correction of the issue discussed in section 4.2 was placed into the `fixltx2e` package and not made part of the kernel algorithm.

<sup>5</sup> If you have an earlier version of L<sup>A</sup>T<sub>E</sub>X installed, you can still extract this code yourself, by writing a short installation file `fltrace.ins` with the following content:

can load to trace some strange float placement decisions, or simply to understand the algorithm a bit better. It offers the commands `\tracefloats` and `\tracefloatsoff` to start or stop tracing the algorithm and `\tracefloatvals` to display the current values of various float parameters that are discussed in this article.

As the package is identical to the kernel code with tracing added, it may or may not work if you load any other package that manipulates that part of the kernel code. In such a case your best bet is to load `fltrace` first.

## 6 How to address specific issues

In the final section we discuss a few strategies to circumvent or resolve common issues. It is by no means comprehensive and you may find further information in other publications, e.g., *The L<sup>A</sup>T<sub>E</sub>X Companion* [2] that devotes a whole chapter to the topic of floats.

### 6.1 Ensure that floats appear “here”

Sometimes it is necessary to ensure that floats appear in-line at certain points in the document text even if that results in some partially empty pages. As discussed above the `h` specifier doesn’t provide this functionality but there are extensions that do, such as the `float` package which offers an `H` specifier for this purpose.

An alternative is the `\captionof` command from the `caption` package that generates a normal float caption (including its entry in the list of figures or tables, etc.) but without the need for a surrounding float environment.

### 6.2 Provide a bottom float area for two-column floats

As discussed above, the standard algorithm doesn’t support double-column floats at the bottom of pages. This missing functionality is added, except for the first page<sup>6</sup>, if you load the `stfloats` package.

### 6.3 Ensure that floats are always placed after their call-out

By default the L<sup>A</sup>T<sub>E</sub>X float algorithm allows for floats to move before their call-out as long as float and call-out are on the same page; more precisely, it allows floats to appear in the top area of the column in which the float has been encountered.

```
\input docstrip
\generateFile{fltrace.sty}{t}{%
  \from{l1output.dtx}{fltrace,trace}}
\endbatchfile
```

and run this through L<sup>A</sup>T<sub>E</sub>X.

<sup>6</sup> See [5] in this issue to manually lift even this restriction.

This practice offers a better chance that the float is visible from the call-out position and doesn't end up on a later page. For some journals, however, this is too liberal and they require that floats are strictly placed after their call-out, i.e., that in the call-out column only the bottom area forms a valid placement option. To accommodate this requirement, this strategy is implemented by the `flafter` package.

This may work well if your document has only a few floats. For documents with lots of floats, placement obviously becomes much more difficult, and you may find that all your floats appear together at the end of the document or chapter, or you may receive a “Too many unprocessed floats” error.

#### 6.4 Prevent floats on certain pages

Sometimes it is helpful to prevent floats from appearing on a certain page, for example, to prevent a float in a new section from moving into the top area on the current page without disallowing a placement in the top area of a later page. For this type of fine-tuning  $\LaTeX$  offers the command `\suppressfloats[placement]`. The optional argument can be either `t` or `b` and prevents any further placement into the respective area(s) on the current page. Without an argument, all remaining floats on the current page are deferred.

#### 6.5 Implement float barriers

Standard  $\LaTeX$  already implements a float barrier called `\clearpage`. Floats on either side will never appear on the other. It works by outputting all deferred floats, if necessary by generating float pages, and then starting a new page. While this is suitable to keep floats within one chapter (as chapters typically start on a new page) there are cases where one would wish for a less intrusive barrier, i.e., one that works without forcing a new page or is partially porous.

This functionality is offered by the `placeins` package, which implements a `\FloatBarrier` command that doesn't introduce a page break. Through package options you can alter the behavior to allow for floats to migrate from one side to the other as long as they still appear on the same page.

#### 6.6 Overwrite placement restrictions

If a given float is (slightly) too large to fit into a certain area or if an area already contains the maximum number of floats but you nevertheless want to force the current float into this place then adding `!` to the optional argument of the float is a good choice. It results in ignoring all restrictions implemented through parameters for this particular float, so that it will

always be placed unless there are already deferred floats with the same float class or the allowed areas get bigger than the available space when adding the float.

As the order of attempts is still the same (first top then bottom), you may have to use `![b]` to force a float into the bottom area as `![tb]` would normally already succeed in placing it into the top area. The downside is of course that if the float doesn't fit, it will only appear in the bottom area of a following page. Thus any later text change may create havoc on your placement decisions.

#### 6.7 Final tuning advice

There are many ways to fine-tune the behavior of the float placement algorithm; most of them have been discussed in this article. However, there is one more “tuning” possibility and in fact the biggest of all: changes in your document text.

Therefore, as final advice: do not start manipulating parameters or change placement specifiers or move floats within your document until after you have fully written your text and your document is close to completion. It is a waste of effort and it may even result in inferior placements as your initially provided restrictions may no longer be adequate after a text change.

#### References

- [1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1994. Reprinted with corrections in 1996.
- [2] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion. Tools and Techniques for Computer Typesetting*. Addison-Wesley, Reading, MA, USA, second edition, 2004. Also available as an eBook, see <http://www.latex-project.org/site-news.html#2013-11-02>.
- [3] Marco Daniel. How to influence the position of float environments like figure or table in  $\LaTeX$ ?, 2012. <http://tex.stackexchange.com/q/39020>.
- [4] Martin Hermann. “thanks” note (footnote) placed below right column even though there is enough space on the left, 2012. <http://tex.stackexchange.com/q/43294>.
- [5] Barbara Beeton. Placing a full-width insert at the bottom of two columns. *TUGboat*, 35(3):255–255, 2014. <http://tug.org/TUGboat/35-3/tb111beet-banner.pdf>.

◇ Frank Mittelbach  
 $\LaTeX$ 3 Project  
<http://www.latex-project.org>

## Placing a full-width insert at the bottom of two columns

Barbara Beeton

There's one location on a two-column page where a full-width `\begin{figure*}` can't be placed under ordinary circumstances—the bottom. The package `stfloats` lifts that restriction—except for the first page. The purpose of the present exercise is to demonstrate that this can in fact be done, using only basic L<sup>A</sup>T<sub>E</sub>X tools.

Why might one want (or need) to do this? Consider a project for which an interim report is best expressed as a table or diagram, with very little prose, but the required report format specifies two columns. The impact of the data is lost if the illustration must be deferred to another page, while the first page is nearly empty. In fact, the meat of the report could even be lost, when the impatient recipient fails to turn the page over.

At the 2010 TUG annual meeting, Frank Mittelbach presented a talk entitled “Exhuming coffins from the last century” that dealt with the problems of positioning boxes on a page. The talk didn't make it into print in *TUGboat*, but Kaveh Bazargan was there with his recording equipment, and produced a video that can be viewed at [river-valley.zeeba.tv/exhuming-coffins-from-the-last-century/](http://river-valley.zeeba.tv/exhuming-coffins-from-the-last-century/). The techniques proposed there won't solve this problem any time soon, but they show promise for the future.

At TUG 2014 in Portland, Boris Veytsman gave a talk<sup>1</sup> on composing a book in which the illustrations were more important—and occupied more space—than the text, and indeed, there were pages with

two columns of text at the top and a single wide illustration at the bottom. However, the nature of the material allowed all pages to be divided into four quadrants which could be managed individually or as horizontal or vertical pairs. That doesn't help in solving the more general problem.

So what can be done today? A L<sup>A</sup>T<sub>E</sub>X-flavored kludge that will produce a one-page document with two columns at the top and a full-width insertion at the bottom is shown in fig. 1.<sup>2</sup> Of course, this also works for a longer document, but for this demonstration, one *TUGboat* page is sufficient. It is also evident that the method works with footnotes (and other such insertions), and that cross-references work normally.

The figure is given as an override single-column [b] figure, in the first column. The page must contain enough text to continue into the second column. Once there is enough text, the trick is to issue a *negative* `\enlargethispage` command that will leave the bottom part of the second column blank, allowing the full-width figure to overflow into the empty area. (On a two-column page, `\enlargethispage` is equivalent to the (nonexistent) `\enlargethiscolumn`.)

Of course, this is entirely manual, and requires intervention and iteration, preferably after the text is final. Tweaking of L<sup>A</sup>T<sub>E</sub>X's float parameters, such as `\bottomfraction`, is likely. Captions may require still more effort. Nevertheless, there are situations in which it makes possible a desirable effect that cannot otherwise be accomplished. Enjoy!

◇ Barbara Beeton  
<http://tug.org/TUGboat>  
 tugboat (at) tug dot org

<sup>1</sup> “An output routine for an illustrated book: Making the *FAO Statistical Yearbook*”, *TUGboat* 35:2, pages 202–204.

<sup>2</sup> This technique was presented in [TeX.stackexchange.com/q/107270](http://tex.stackexchange.com/q/107270).

```
\documentclass{ltugboat}
\title{Placing a full-width insert at the bottom of two columns} \author{Barbara Beeton}
\begin{document}
\maketitle
There's one location on a two-column page where a full-width ...
\begin{figure}[b]\setlength{\hfuzz}{1.1\columnwidth}
\begin{minipage}{\textwidth}
\ttfamily ... code for the insertion ...
\end{minipage}
\end{figure}
At the 2010 \tug\ annual meeting, Frank Mittelbach presented ...
\enlargethispage{-16.5\baselineskip}
Of course, this is entirely manual, and requires intervention ...
\end{document}
```

Figure 1: A full-width figure at the bottom of the first page!

---

**biblatex variations**

Ulrike Fischer

**Abstract**

I show three small examples of using the `biblatex` package for more than printing bibliographies: we will redefine bibliography drivers, define new cite commands and declare new entry types to create qrcodes, insert PDF files and manage addresses.

**Remark**

In April I gave a talk at the DANTE e.V. meeting about `biblatex` variations and later wrote an article for the proceedings in *Die T<sub>E</sub>Xnische Komödie*. This is more or less a translation of that article. I didn't adapt the examples, so they are still in German.

***ad hoc* small-scale databases**

I have always been interested in the handling of small databases. I wrote my first articles in *DTK* ([1, 2]) about a mail merging system that I developed to handle around 50 addresses. And later on I regularly had to find ways to automatically process small numbers of data records without too much fuss.

For such small scale databases, the `bib` is an interesting option — at least on the input side. Looking at it without the prejudice “that is something for bibliographies only” — Listing 1 shows such a typical `bib` file, that I will use in this article — one can see that it has several features making it suitable for *ad hoc* small scale databases:

- One can mix different datatypes in one file.
- One can easily create new datatypes.
- One can easily add new fields.
- Fields without values can be (should be) omitted. This saves space.
- The records can be created, changed and read with any editor, but there are also good GUIs (e.g. JabRef), so the database can be edited by people who don't know L<sup>A</sup>T<sub>E</sub>X.
- With `@string` one can define variables.
- It is possible to define relations between records, e.g. with `crossref`.

So, it is easy to create a small database but ... how should one process and output the records? Before `biblatex` this was not usually feasible. Creating a suitable `bst` file was a difficult and time-consuming task. But *with* `biblatex` this has completely changed. Now it is possible, with very little effort, to output the records in various ways. The following examples are meant to demonstrate this. They will show various methods one can use. The

**Listing 1:** The example bib file `vortrag.bib`

```

1 @termin{dante2014,
2   title={Biblatex-Variationen},
3   date  ={2014-04-11},
4   time  ={15.15},
5   location={Heidelberg}}
6
7 @online{dante,
8   title={Internetseite dante e.V.},
9   url   ={http://www.dante.de}}
10
11 @online{heidelberg,
12   title={Stadt Heidelberg},
13   url   ={http://www.heidelberg.de}}
14
15 @adresse{max,
16   name   ={Muster, Max},
17   strasse={Im Versuchsweg 10},
18   ort    ={Testgelände},
19   plz    ={X01234},
20   gender  ={sm}}
21
22 @adresse{eva,
23   name   ={Muster, Eva},
24   strasse={Im Versuchsweg 10},
25   ort    ={Testgelände},
26   plz    ={X01234},
27   gender  ={sf}}
28
29 @article{input1,
30   author={Fischer, Ulrike},
31   title  ={Erster Text},
32   journal={Beispiele},
33   date   ={2012-04-08},
34   url    ={inputtext1.pdf}}
35
36 @article{input2,
37   author={Fischer, Ulrike},
38   title  ={Zweiter Text},
39   journal={Beispiele},
40   date   ={2013-02-07},
41   url    ={inputtext2.pdf}}
42
43 @book{gambol,
44   author={Gambolputty de von Ausfern-
45     ↪schplenden-schlitter-crasscrenbon,
46     ↪Johann},
47   title={Titel},
48   year={1970}}
49
50 @book{dante2007,
51   author  = {Dante Alighieri},
52   title   = {Die Göttliche Kommödie},
53   gender  = {sm},
54   location = {Stuttgart},
55   year    = {2007},
56   translator = {Hermann Gmelin}}

```

**Listing 2:** The creation of QR-codes

```

1 % Compile with XeLaTeX
2 \documentclass{article}
3 \usepackage[margin=0.05in,textwidth=1.9in,
  ↪textheight=1.9in,paperwidth=2in,
  ↪paperheight=2in]{geometry}
4 \usepackage{xcolor}
5 \usepackage{pst-barcode}
6 \usepackage{fontspec}
7 \usepackage{biblatex}
8 \addbibresource{vortrag.bib}
9
10 \defbibenvironment{qrcode}{\centering}{}{}
11
12 \DeclareBibliographyDriver{online}{%
13   \begin{minipage}[c][1.9in]{1.9in}
14     \centering
15     \printtext{\thefield{entrykey}}\ [2ex]
16     \printfield{title}\ [2ex]
17     \printfield{url}
18   \end{minipage}
19   \newpage
20   \begin{pspicture}(1.9in,1.9in)
21     \label{\thefield{entrykey}}%
22     \psbarcode[linecolor=red]{\thefield{url}
  ↪}{width=1.9 height=1.9}{qrcode}%
23   \end{pspicture}%
24   \newpage}
25
26 \begin{document}
27 \nocite{*}
28 \printbibliography[env=qrcode,type=online,
  ↪heading=none]
29 \end{document}

```

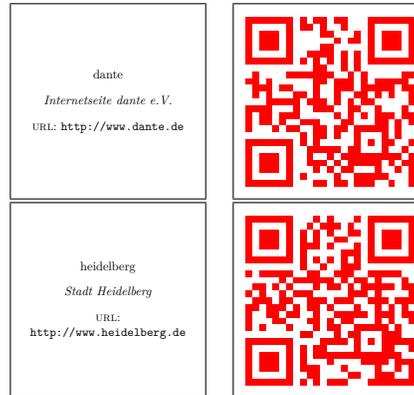
examples are kept as simple as possible. In larger databases one would likely need to add some security precautions, such as tests for empty fields.

## 1 Example 1: QR-codes

In this example, we first create a PDF file which contains the QR-code of the `url` field of every `@online` entry in a `bib` file. The QR-codes can then be inserted with `\includegraphics` in another document. As methodology, the redefinition of a *bibliography driver* is shown.

### 1.1 The creation of the QR-codes

Herbert Voß has shown how QR-codes can be created generally in a *DTK* article [3]. That method uses the package `pst-barcode`, meaning that one needs a  $\TeX$  compiler which can handle PostScript; I usually use `xelatex`.

**Figure 1:** The PDF output pages with QR-codes

Listing 2 shows how one can create QR-codes from the `url` field of a `bib` file. The actual document body is very short (lines 26–29): all entries are cited with `\nocite{*}` and then the bibliography is printed with `\printbibliography`, which is given three options: `heading=none` suppresses the heading of the bibliography; `type=online` only outputs entries of type `@online`; and `env=qrcode` ensures that the bibliography is not a rather complicated list but the simple environment defined in line 10.

The start of the listing (lines 3–8) is basic: the page size is set to  $1.9\text{ in} \times 1.9\text{ in}$ , needed packages are loaded and the name of the `bib` file is declared.

Line 10 defines a simple `qrcode` environment, as the standard list environment would only insert unwanted spaces.

The core of the approach is in lines 12–24. They define how a `@online` entry is formatted in the bibliography. The code creates two pages for each entry.

The first page shows some information about the following QR-codes. This page is not required; I produce it only because it looks nice. The code uses the `biblatex` commands `\printtext`, `\printfield` and `\thefield` to output fields from the `bib` record.

Lines 20–24 create the second page with the QR-code. A useful addition is the label, specified in line 21: an external document is then able to find the page with the QR-code of a specific `bib` key (the “entrykey”). In line 22 the QR-code is created. The `url` is inserted with `\thefield{url}`.

Compiling with `xelatex-biber-xelatex` results in a PDF file with four pages, shown in figure 1.

### 1.2 Using the QR-codes

Listing 3 shows how one can insert the QR-codes in other documents. The code uses the package `refcount` to convert a label to a number which can be used with the option `page` of `\includegraphics`.

Listing 3: Inserting the QR-codes

```

1 \documentclass[parskip=half-]{scrartcl}
2 \usepackage{graphicx,refcount,xr}
3 \externaldocument[qrcode-]
4   {qrcodes-bib-erzeugen}
5 \begin{document}
6 \section*{QR-Codes laden}
7 \includegraphics[page=
8   \getpagerefnumber{qrcode-dante}]
9   {qrcodes-bib-erzeugen-dtk}
10 \quad
11 \includegraphics[page=
12   \getpagerefnumber{qrcode-heidelberg}]
13   {qrcodes-bib-erzeugen-dtk}
14 \end{document}

```

It also uses the package `xr` to access the labels of the document with the QR-codes. The code must be able to find the PDF and `aux` files of the external document with the QR-codes.

## 2 Example 2: Inserting PDF attachments

The second example inserts a PDF file in a document and writes information about this file to the table of contents. The method used is the definition of a new cite command. The example also shows that things do not always work as smoothly as one would wish.

Listing 4 shows the core idea: In lines 12–27 a new cite command with the name `\citeanlageX` is defined. Defining a cite command is a bit more complicated than defining a driver, as a cite command can have *lists* of entries as an argument.

In the so-called *loopcode* argument (lines 14–26) a new page is started and the counter for the attachment is advanced (line 14). Then in lines 15–22 an entry for the `toc` file is written. The content of this entry is the word “Anlage” followed by the number and a `\fullcite`.

In lines 23–26 the file from the `url` field is included with `\includepdf`. The existence of the file is checked first with `\IfFileExists`.

`\citeanlageX` does everything that is needed, but it has a flaw: it can lead to unwanted empty pages. The problem is that every `biblatex` cite command internally executes `\leavevmode`, and so can start a page. Together with the `\clearpage` there is then a page too many.

So I had to dig around a bit in the code to find the source of the `\leavevmode`. Happily it is easy to deactivate it locally. This is done in lines 29–36.

Listing 4: Inserting PDF attachments

```

1 \documentclass[parskip=half-,toc=flat]{
  ↪scrartcl}
2 \usepackage[utf8]{inputenc}
3 \usepackage[T1]{fontenc}
4 \usepackage[ngerman]{babel}
5 \usepackage[autostyle]{csquotes}
6 \usepackage{pdfpages}
7 \usepackage[style=authoryear]{biblatex}
8 \addbibresource{vortrag.bib}
9
10 \newcounter{anlage}
11
12 \DeclareCiteCommand{\citeanlageX}
13   {}
14   {\clearpage\refstepcounter{anlage}%
15     \addtocontents{toc}
16     {\protect\contentsline
17       {section}
18       {\protect\numberline{Anlage~\
19 ↪theanlage}%
20       \protect\fullcite{\thefield{
21 ↪entrykey}}}%
22     }%
23     {\thepage}
24     {}}%
25   \IfFileExists{\thefield{url}}
26   {\includepdf[pages=-,fitpaper]{\thefield{
27 ↪url}}}
28   {\par\textbf{Datei zu \thefield{entrykey
29 ↪} wurde nicht gefunden!}%
30     \clearpage}}
31   {}{}
32
33 \makeatletter
34 \newcommand\citeanlage[1]{%
35   \begingroup
36     \let\blx@leavevmode\relax
37     \let\blx@leavevmode@cite\relax
38     \citeanlageX{#1}%
39   \endgroup}
40 \makeatother
41
42 \begin{document}
43 \tableofcontents
44
45 \citeanlage{input1} \citeanlage{input2}
46 \end{document}

```

### Inhaltsverzeichnis

Anlage 1 Ulrike Fischer (2012). „Erster Text“. In: <i>Beispiele</i> . URL: inputtext1.pdf	2
Anlage 2 Ulrike Fischer (2013). „Zweiter Text“. In: <i>Beispiele</i> . URL: inputtext2.pdf	4

Figure 2: The table of contents created by Listing 4

Listing 5: The datamodel file `ufischer.dbx`

```

1 \DeclareDatamodelEntrytypes{adresse}
2
3 \DeclareDatamodelFields[type=list,
4   ↪datatype=name]
5   {name}
6
7 \DeclareDatamodelFields[type=field,
8   ↪datatype=literal]
9   {strasse,ort,plz}
10
11 \DeclareDatamodelEntryfields[adresse]{%
12   name,strasse,ort,plz,gender}

```

### 3 Example 3: Managing addresses

The third example is a bit more complicated. It shows how to manage addresses in a `bib` file. The new method shown this time is how the *datamodel* can be extended.

New entrytypes and fields should be declared in a `dbx` file, as shown in Listing 5. In line 1 a new entry type `adresse` is added and in lines 3–7 new fields for this type. One should also declare which fields can be used by an entry type. A new entry type can use known fields, such as the field `gender` on the last line.

Listing 6 shows how to use the new entry type. First, the new datamodel is loaded in line 7 with `datamodel=ufischer`.

Lines 10–14 declare a new bibliography driver `adresse` for the distribution list; it creates a simple, comma-separated list of the data.

Line 16 declares a *name format* for the greeting in the letter. It prints only the last name from a name.

In lines 18–22 a cite command for the greeting is defined: `\citeanrede`. It uses the `gender` field to decide if “Frau” or “Herr” should be printed before the name.

In lines 24–30 another cite command is defined: `\citeadresse`. This command is meant for the address window and prints the data line-by-line.<sup>1</sup>

Lines 32–43 show how the various commands can be used. The output can be seen in figure 3.

### 4 Finally: How to control the content of the bibliography?

When one starts to “misuse” `bib` entries for things other than standard citations, one quickly finds the need to prevent such entries from finding their way

<sup>1</sup> Many of the comment signs in the code are not needed but they do no harm, either.

Listing 6: How to use the type `@adresse`

```

1 \documentclass[parskip=half-,toc=flat,
2   ↪fontsize=9pt,DIV = 9,
3     paper=a5,pagesize,headings=
4     ↪normal]{scrartcl}
5 \usepackage[utf8]{inputenc}
6 \usepackage[T1]{fontenc}
7 \usepackage[ngerman]{babel}
8 \usepackage[autostyle]{csquotes}
9 \usepackage[datamodel=ufischer,defernumbers
10  ↪]{biblatex}
11 \addbibresource{vortrag.bib}
12
13 \DeclareBibliographyDriver{adresse}{%
14   \printnames{name}\setunit{\addcomma\
15   ↪addspace}%
16   \printfield{strasse}\setunit{\addcomma\
17   ↪addspace}%
18   \printfield{plz}\setunit{\addspace}\
19   ↪printfield{ort}%
20   \usebibmacro{finentry}}
21
22 \DeclareNameFormat[adresse]{anrede}{#1}
23
24 \DeclareCiteCommand{\citeanrede}{}{%
25   \iffieldequalstr{gender}{sm}
26   {\printtext{Herr}}{\printtext{Frau}}%
27   \setunit{\addspace}\printnames[anrede]{
28   ↪name}}
29   {}{}
30
31 \DeclareCiteCommand{\citeadresse}{}{%
32   \printtext{\par\noindent}%
33   \iffieldequalstr{gender}{sm}{\printtext{
34   ↪Herr}}{\printtext{Frau}}%
35   \setunit{\}\printnames{name}%
36   \setunit{\}\printfield{strasse}%
37   \setunit{\}\printfield{plz}\setunit{\
38   ↪addspace}\printfield{ort}}
39   {}{}
40
41 \begin{document}
42 \citeadresse{max}
43 \citeadresse{eva}
44
45 \bigskip
46 Lieber \citeanrede{max}, liebe \citeanrede{
47   ↪eva},
48
49 schaut euch doch mal \cite{dante} an und
50   ↪lest \cite{input1}
51
52 \printbibliography[type=adresse,title=
53   ↪Verteilerliste]
54 \printbibliography[notttype=adresse,
55   ↪resetnumbers]
56 \end{document}

```

```

Herrn
Max Muster
Im Versuchsweg 10
X01234 Testgelände

Frau
Eva Muster
Im Versuchsweg 10
X01234 Testgelände

Lieber Herr Muster, liebe Frau Muster,
schaut euch doch mal [2] an und lest [1]

Verteilerliste

[1] Max Muster, Im Versuchsweg 10, X01234 Testgelände.
[2] Eva Muster, Im Versuchsweg 10, X01234 Testgelände.

Literatur

[1] Ulrike Fischer. „Erster Text“. In: Beispiele (8. Apr. 2012). URL:
inputtext1.pdf.
[2] Internetseite dante e.V. URL: http://www.dante.de.

```

Figure 3: The output of Listing 6

Listing 7: Unwanted cite commands

```

1 ... Namen \citeauthor{gambol} und
2 \citeauthor{dante2007} ist nicht leicht ...
3 Die göttliche Komödie ... \cite{dante2007}

```

into the “normal” bibliography. The previous examples have already shown some possibilities: `bibtex` has excellent filter features. E.g., with `nottype` one can exclude a given type from a bibliography.

But this doesn’t help when we need to avoid specific cite commands leading to an entry in the bibliography. Listing 7 demonstrates the problem. It “mis-” uses the standard `\citeauthor` to facilitate the writing of a complicated name (here, from a Monty Python sketch). Neither `\citeauthor` command should trigger entries in the bibliography. But `dante2007` is cited normally later on. So it is not possible to exclude (e.g., with the keyword `skipbib`) both entries completely from the bibliography.

Listing 8 gives one possible solution to this dilemma: we introduce a new category `inbib` (line 1) and a new boolean variable `citeinbib` (lines 2–3). Using `\AtEveryCitekey`, a cited entry is added to the category if the variable is true (lines 4–6). If a cite command should not create an entry in the bibliography, `citeinbib` is set locally to false (lines 9–10), so the entry is not added to the category. Finally, `\printbibliography` can then filter using the category (line 13).

Listing 8: One solution for cite commands which should not lead to bibliography entries

```

1 \DeclareBibliographyCategory{inbib}
2 \newboolean{citeinbib}
3 \booltrue{citeinbib}
4 \AtEveryCitekey{%
5     \ifbool{citeinbib}{%
6         \addtocategory{inbib}{\thefield{
7             ↪entrykey}}{}}
8 \begin{document}
9 Die Schreibweise der Namen {\boolfalse{
10     ↪citeinbib}\citeauthor{gambol} und
11 \citeauthor{dante2007}} ist nicht leicht zu
12     ↪ merken.
13 Die göttliche Komödie ... \cite{dante2007}
14 \printbibliography[category=inbib]
15 \end{document}

```

Die Schreibweise der Namen Gambolputty de von Ausfern-schplenden -schlitter -crasscrenbon und Alighieri ist nicht leicht zu merken.

Die göttliche Komödie ... [1]

#### Literatur

- [1] Dante Alighieri. *Die Göttliche Komödie*. Übers. von Hermann Gmelin. Stuttgart, 2007.

Figure 4: The output of Listing 8

## 5 Summary

I hope the three examples have shown that `bibtex` offers much more than a way to print bibliographies.

## References

- [1] Ulrike Fischer. Eine Schnittstelle zwischen Datenbanken und L<sup>A</sup>T<sub>E</sub>X. *Die T<sub>E</sub>Xnische Komödie*, 4/98:28–33, Dec. 1998.
- [2] Ulrike Fischer. Serienbriefe. *Die T<sub>E</sub>Xnische Komödie*, 2/99:38–44, May 1999.
- [3] Herbert Voß. QR-Codes im Rand ausgeben. *Die T<sub>E</sub>Xnische Komödie*, 4/13:34–37, Nov. 2013.

◇ Ulrike Fischer  
Bismarckstr. 91  
41061 Mönchengladbach  
fischer (at) troubleshooting-tex dot de

## Every L<sup>A</sup>T<sub>E</sub>X document brings new programming issues

David Walden

*Background: For several years I wrote a column, called Travels in T<sub>E</sub>Xland, for The PracT<sub>E</sub>X Journal about the way I used L<sup>A</sup>T<sub>E</sub>X. Although I am not a L<sup>A</sup>T<sub>E</sub>X expert, I was once a full-time computer programmer and am not afraid to bash around trying to find some (perhaps ad hoc) way to accomplish some typesetting goal. After I stopped writing the column, I still sometimes wrote up something I had figured out for the purpose of reflecting on what was done. The present article pulls together three such reflections.*

### 1 Introduction

There is seldom a time I am not composing a document drafted in L<sup>A</sup>T<sub>E</sub>X. Each document brings its own style and efficiency issues and, thus, each time I seem to have to solve some new little L<sup>A</sup>T<sub>E</sub>X programming problem.

A natural question might be, “Why not find an existing package that does what you need rather than coding your own thing?” Of course, sometimes what I need may well be provided by an existing package. However, mostly I am too lazy to search out an existing package if I can’t immediately find one that meets my needs; or I find one but it doesn’t install and work without difficulty and without much study. I’d rather code my own little thing than struggle with package installation issues or inter-package interface issues. I’d rather do my own little thing (even if it is less efficient than what exists) to avoid having to understand complex documentation.<sup>1</sup>

L<sup>A</sup>T<sub>E</sub>X is swell because it is programmable, such that I can create little “tools” that help me do what I want to do. Also, like any experienced programmer, I collect these little solutions for reuse in future documents by copying rather than new thinking.

In this note I give three examples of such little programming problems and the (perhaps quick-and-dirty) solutions at which I arrived:

- Issues and ways for typesetting ellipses
- Blank verso sides without using the book class’s `twoside` option
- Flexible layout of a photo album

All three examples here derive from my work on which I have written before—self- or private publishing.<sup>2</sup>

### 2 Typesetting ellipses with L<sup>A</sup>T<sub>E</sub>X

There are many situations and approaches for using ellipses in L<sup>A</sup>T<sub>E</sub>X. After sketching some of the myriad

situations and a few of the approaches, I describe what I do.

#### 2.1 Diversity of situations

I am mainly concerned with the use of ellipses in American English, non-mathematical writing. In this context, ellipses seem to have two purposes: (1) indicating where something has been left out of quoted text; (2) to indicate a pause or something never stated—an unfinished thought or an implicit thought (“and so on”).

Here are three examples (in which I have not tried to perfect the typesetting of the ellipses):

1. “Four score and seven years ago our fathers brought forth . . . a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.”  
The words “on this continent” have been left out.
2. My wife may think that I am fussy about little things . . .  
The ellipsis here might indicate that I have a lot more to say about this subject that will go unspoken.
3. “Leave me alone . . . I’m too tired to talk about it,” he said.

If the quoted words are in dialog, the ellipsis might indicate a pause in the speech.

Ellipses are complicated for at least three reasons:<sup>3</sup> (1) they can have many different uses; (2) there are different sets of conventions for how to indicate the elision, for example, always using three dots or sometimes using three and sometimes using four dots depending on context within sentences; (3) there are different approaches for typesetting ellipses.

Here are some of the possible contexts for use of ellipses:

- at the end of a sentence
- within a sentence
- at the beginning of a line
- at the end of a line
- within a line
- with, or without, other punctuation before, or after, the ellipsis

Lots of combinations of these and other situations also can happen.

#### 2.2 Different conventions

On pages 82–83 of his *The Elements of Typographic Style*,<sup>4</sup> Robert Bringhurst gives a sketch covering some of the conventions for using ellipses. In the following quotation, the instances of ellipses in the first paragraph and the very last instance are part of Bringhurst’s text (and are typeset as specified). The

rest of the ellipses are by me to indicate my elisions from the Bringhurst quote.

Most digital fonts now include, among other things, a prefabricated *ellipsis* (row of three baseline dots). Many typographers nevertheless prefer to make their own. Some prefer to see the three dots flush ... with a normal word space before and after. Others prefer ... to add *thin* spaces between the dots. Thick spaces (M/3) are prescribed by the *Chicago Manual of Style*, ... In most cases the Chicago ellipsis is much too wide.

Flush set ellipses work well with some fonts and faces but not with all... At small text sizes ... it is generally best to add space ... between the dots. Extra space may also look best in the midst of light, open letterforms, ..., and less space in the company of a dark font, ..., or when setting in bold face...

In English ..., when the ellipsis occurs at the end of a sentence, a fourth dot, the period, is added and the space beginning the ellipsis disappears... When the ellipsis combines with a comma, exclamation mark or question mark, the same typographic principle applies. Otherwise a word space is required fore and aft.

However, the Bringhurst summary leaves out other common conventions. For instance, pages 292–296 of my copy of *Chicago Manual of Style*<sup>5</sup> starts by giving two main conventions for the form of ellipses: (1) always only three dots, or (2) four dots at the end of sentences (or three dots and another punctuation mark, as described by Bringhurst) and three dots elsewhere. The latter is the manual’s preferred convention. Ellipses in block quotes also provide additional circumstances beyond those mentioned in Bringhurst’s sketch.

There are a number of useful on-line discussions of the use of ellipses, for example, in the Wikipedia<sup>6</sup> and in Doc Scribe’s Guide to research styles, where you can look up the approaches recommended in five well-known style guides (AMA, APA, ASA, Chicago, and MLA).<sup>7</sup>

### 2.3 Standard tools versus hand crafting

It seems to me that merely using `\dots` or `\ldots` in L<sup>A</sup>T<sub>E</sub>X is often not enough to address some of the potential needs mentioned in the previous sections. (See also section 2.5.)

Also, naive use of `\dots` apparently has a problem that Peter Heslin’s ellipsis style<sup>8</sup> works on fixing. As Heslin says,

There is a problem in the way L<sup>A</sup>T<sub>E</sub>X handles ellipses: it always puts a tiny bit more space after `\dots` in text mode than before it, which often results in the ellipsis being off-center when set between two other things.

It is worth reading the documentation of Heslin’s package, which also describes some of the issues relating to using ellipses. The package also allows one to specify the Chicago or MLA style and to specify the spacing between dots (e.g., in terms of an em) in an ellipsis.

Another package is `lips.sty`,<sup>9</sup> which Heslin suggests using if one wants the full Chicago style.

With so many possibilities and needs, it is not surprising that, as Bringhurst says, “Many typographers nevertheless prefer to make their own.” It is hard for me to imagine a package with sufficient capabilities and options for everyone. (But maybe I’m wrong.)

### 2.4 My approach

My approach has been to define a few macros to handle common situations for using ellipses in *the writing I do*. These also implement my own preferences, such as for inter-dot spacing and spacing before and after an ellipsis. I have used a couple of versions of these macros.

#### Version 1 of ellipses

The following definitions were sufficient for the *Breakthrough Management* book ([walden-family.com/breakthrough](http://walden-family.com/breakthrough)) which I co-authored and typeset. I used the Minion typeface for this book. The comments in the following code provide the relevant explanations.

```
% dots for main text
\def\bigdotsspace{3pt}

% three dots
% I like the same size space on each side
% of the ellipsis as between its dots.
\def\mydots{\hbox{\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}}}}

% period and three dots = four altogether
% and the same size space after a period
% and before 3 dots
\def\fmydots{\hskip0pt}%
\hbox{.\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}}}
```

```

% dots with only beginning space,
% no following space.
% I use this with an ellipsis and
% following comma, etc.
\def\mydotsnfs{\hbox{\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}%
.\hspace{\bigdotsspace}%
.}}

% dots for block quote text,
% which is smaller than main text
\def\smalldotsspace{2pt}

% small dots without end spaces
\def\minsmalldots{\hbox{%
.\hspace{\smalldotsspace}%
.\hspace{\smalldotsspace}%
.}}

% small dots with end spaces
\def\smydots{\hbox{\hspace{\smalldotsspace}%
\minsmalldots\hspace{\smalldotsspace}}}

% period + three small dots = four altogether
\def\fsmydots{\hskip0pt}%
\hbox{\hspace{.3pt}.\hspace{\smalldotsspace}%
\minsmalldots\hspace{\smalldotsspace}}}

```

## Version 2 of ellipses

I used the following set of definitions with the book I compiled and typeset about the technology history of the company Bolt Beranek and Newman.<sup>10</sup> This book uses the Lucida Bright typeface. For this second book, I had learned about doing things in terms which varied with font size, e.g., among main, footnote, and block quote text. My decisions in the following definitions are only about what looks good to me, not about the conventions of a particular style manual. (The comments in the following code provide the relevant explanations.)

```

% with an end-of-sentence ellipsis I use
% a following thin, not word, space
\def\fourdots{\hbox{\hspace{.33em}%
.\hspace{.33em}.\hspace{.33em}.\,}}

% sometimes I don't even want the
% trailing thin space, e.g., at end of line
\def\fourdotstightright{\hbox{\hspace{.33em}%
.\hspace{.33em}.\hspace{.33em}.}}

% for a non-end-of-sentence ellipsis
\def\threedots{\hbox{\,.\hspace{.33em}%
.\hspace{.33em}.\,}}

% for beginning of line, e.g., block quote
\def\threedotstightleft{\hbox{.\hspace{.33em}%
.\hspace{.33em}.\,}}

```

```

% for end of line, e.g., in a block quote
\def\threedotstightright{\hbox{\,.\hspace{.33em}%
.\hspace{.33em}.}}

% tried this but didn't use it
%\def\sencespace{\unskip\spacefactor=3000
% \space\ignorespaces}

% also tried this but didn't use it
%\def\fourdots{\unskip\kern\fontdimen3\font
% .\kern.1667em\ldots\sencespace{}}

% also tried this but didn't use it
%\def\threedots{\unskip\ldots\unskip{}}

% for use in a footnote; I originally thought
% I might need a different definition, but
% then I was happy with the main text ratios
\def\fnfourdots{\fourdots{}}

% ditto
\def\fnthreedots{\threedots{}}

```

## Ellipses summary

It is easy to see how my set of definitions could be adapted to using word or sentence spaces before and after an ellipsis while using some other appropriate inter-dot spacing, or to adapt the definitions to other traditional or personal conventions. For instance,

```

% sentence space after four dots:
\def\fourdots{\hbox{.\hspace{.33em}%
.\hspace{.33em}.\hspace{.33em}. }

```

I am also sure that there are better approaches than mine for handling a variety of ellipsis situations in L<sup>A</sup>T<sub>E</sub>X, or at least better ways to do what I am doing (perhaps automatically detecting whether an ellipsis is at the beginning or end of a line and thus eliminating the need for those definitions).

## 2.5 Ellipses: appendix

By the way, in the file `latex.ltx`, I found the following definitions, presented without comment.

```

\DeclareTextCommandDefault{\textellipsis}{%
.\kern\fontdimen3\font
.\kern\fontdimen3\font
.\kern\fontdimen3\font}

\DeclareRobustCommand{\dots}{%
\ifmmode\mathellipsis\else\textellipsis\fi}

\let\ldots\dots

```

### 3 Blank verso sides without using the book class twoside option

Several years ago I was involved in creating a small book (approximately 100 pages<sup>11</sup>), and a year later I did the L<sup>A</sup>T<sub>E</sub>Xing of a small pamphlet (approximately 60 pages<sup>12</sup>). In both cases, the document needed to look like a book, but using all the built-in capabilities of the book class wasn't necessary. Therefore, I drafted my own class file (which, in the first case, Karl Berry significantly improved) and loaded that on top of the standard book class, e.g.,

```
\documentclass{book}
\usepackage{ctssbook}
```

Naturally, the added style file included a macro, `\beginnewchapter`, which reset the various counters (such as footnote and figure numbers), formatted the chapter title, changed the running headings, and put the chapter title in the table of contents using the command

```
\addcontentsline{toc}{chapter}
  {\protect\fmittocnumber{\thechapter}#1}
```

where `#1` is the chapter title passed to the macro via the macro call (and `\fmittocnumber` is a macro that formats a right-justified chapter number in a properly sized field).

Because the command `\chapter` is never given in the L<sup>A</sup>T<sub>E</sub>X for these two books, the two-side and one-side capabilities of the book style aren't available. This is not a problem. For a document that will be printed and needs to start a new chapter on a recto side, it is easy enough (in the last stages of typesetting) to, first, perfect the page breaks: for this I typically use calls to macros such as

```
\newcommand{\Lpushlines}[1]
  {\enlargethispage{-#1\baselineskip}}
\newcommand{\Lpulllines}[1]
  {\enlargethispage{#1\baselineskip}}
```

And then, second, to go through the root file of the document and add a macro call (after the commands to input the content of chapter, frontmatter and backmatter files) to create the necessary blank verso sides where needed. This end-of-chapter macro definition is something like

```
\def\EOC{%
  \newpage\null\thispagestyle{empty}\newpage
}
```

and the resulting root file looked like this:

```
\documentclass{book}
\usepackage{ctssbook}
\begin{document}
\frontmatter
```

```
  \include{title-pages}
  \include{preface}
\mainmatter
  \include{history}
  \EOC
  \include{toms-webpage-r1}
  \EOC
  \include{uses-r}
  \include{views}
  \EOC
  \include{other}
\backmatter
  \include{biblio}
  \EOC
  \input{colophon}
  \EOC
\end{document}
```

In the pamphlet shown in this example, some chapters already end on verso sides and calls to `\EOC` are not needed. Also, the command `\tableofcontents` is included in the `title-pages.tex` file; and, since the table of contents in this case is only one page long, it also includes a call of `\EOC`.

As I was finishing this pamphlet, I needed PDFs both for sending to the printer and for posting on the web. For the printer, there needed to be two PDFs: one for the color cover (i.e., a single file of the back cover, spine, and front cover), and one for the grayscale interior of the pamphlet including blank pages at the end of chapters as needed to start chapters on recto sides. For the web, I needed a single PDF with the front and back covers at the beginning and end of the interior pages, and I decided I wanted to leave out the blank verso sides from the interior, but keep the same page numbers as in the print version.

Thus, I created a macro to conditionally add the covers to the interior and augmented the `\EOC` macro to add blank verso sides only when needed for the print version.

```
\def\Forweb{0} %0 = print
%\def\Forweb{1} %1 = web

\RequirePackage[final]{pdfpages}
\def\Covers#1{%
  \ifodd\Forweb
    % #1 is cover filenames
    \includepdf[pages=1-1]{#1.pdf}%
  \fi}

\RequirePackage{ifthen,changepage}
% if for web, increment page counter
% if for print, output blank page
\def\EOC{\newpage\checkoddpaper
  \ifthenelse{\boolean{oddpaper}}%
    {} {\ifodd\Forweb\stepcounter{page}%
```

```
\else\null\thispagestyle{empty}%
\newpage\fi}}
```

Thus, I added `\Covers` macro calls bracketing the rest of the document as follows:

```
\begin{document}
\Covers{front-cover}
...
\Covers{back-cover}
\end{document}
```

I also then include a call to the revised `\EOC` macro after including *each* chapter, frontmatter, and backmatter file (`title-pages.tex` already contained a call to `\EOC`).

#### 4 Flexible layout of a photo album

This past year I decided to print a dozen or so copies of an album of old photos to distribute to family members. The photos had been pulled from several photo albums of a deceased parent that were broken up and various photos of individuals sent to the individual or a family member of the individual. However, some photos needed to go to more than one person; hence, I wanted to create an album of these remaining photos which could be distributed to multiple family members.

The photos came in a variety of sizes ranging from 8×10 inches to smaller than 3.5×5 inches, many of the sizes being non-standard for today's typical digital printing businesses that serve amateur photographers (these businesses tend to assume 8×10, 5×7, 4×6, and 3.5×5). The photos also came in a variety of conditions from quite good to quite bad (faded or otherwise discolored, or never high quality in the first place). I scanned all of these photos at either 600 pixels per inch or 300 ppi, cropped off the borders on the digital version, and then did lots of Photoshopping to bring as much quality back to the images as I could manage. Because I thought it might be useful, I put the images into six separate directories for 8×10 (the few instances of this only had a vertical orientation), 5×7 (the instances were also all vertically oriented), 4×6 tall orientation, 4×6 wide orientation, 3.5×5 tall orientation, and 3.5×5 wide orientation.

I decided that I wanted the photos in the album I was creating to be the exact size of the originals in inches on the printed page. Consequently, I decided the album's trim size when perfect bound would be 10×11.5 inches. The next step was to figure out how to lay out the photos on printed pages.<sup>13</sup>

##### 4.1 Photo album: First effort

The first macro I wrote, `\image`, took two arguments: an image `directory/filename` and a draft caption

(the file name without its directory), and displayed the image with the caption beneath it at the current location. Then I wrote three other macros:

1. `\oneperpage`, which called `\image` once and centered the specified image and its caption on a page;
2. `\sidebyside`, which called `\image` twice and placed the two images side by side, centered on a page;
3. `\overunder`, which called `\image` twice and placed the two images (and their captions) on the page centered horizontally and spaced out equally in the vertical direction.

I wrote a Perl program to generate calls (in alphabetical order by file name) to the three page-layout macros for all the image files in each of the six directories, with the 8×10 and 5×7 images being placed alone on pages, the 4×6 and 3.5×5 tall images placed side by side on a page, and the 4×6 and 3.5×5 wide images placed in an over-under position on the page. With a little manual text editing, the macro definitions and the Perl-generated calls to the page-layout macros became the  $\LaTeX$  program to generate a first draft album of all of the images.

However, there was a problem. My intention was to print the images at the actual size of the scanned photographs, counting on `\includegraphics` to read the metadata in the image file to specify the print size. This worked well for most of the images. But for some of images in the 3.5×5 (tall) directory, the images printed at much too big a size. Rather than sort out the reason, I chose the brute force course of temporarily changing the definition of `\image` so `\includegraphics` used a width of 3.5 inches in calls by the `\sidebyside` macro.

With the printout of this first draft in hand, I could begin to improve the captions and think seriously about layout issues. With actual captions written, many were wider than their image which didn't look good on horizontal images and which were completely broken on side-by-side images. So I redefined `\image`, as follows, to measure the width of the image and make the caption be that width:

```
\def\image#1#2{
\centerline{\includegraphics{#1}}
\smallskip

\settowidth\imagewidth{\includegraphics{#1}}
\begin{minipage}[b]{\imagewidth}
\centering
\large#2
\end{minipage}
}
```

## 4.2 Photo album: Next approach

With the new (to me) concept of measuring the image width and adjusting the caption width accordingly, I redid the page layout macros.

The `\overunder` macro could use the new definition of `\image` directly as shown in the following definition and example call:

```
\def\overunder#1#2{
  \clearpage \vspace*{\fill}
  #1\vfill
  #2\vfill
  \clearpage
}
```

called like:

```
\overunder
  {\image{filename1}}{\caption1}}
  {\image{filename2}}{\caption2}}
```

I redid the `\oneperpage` and `\sidebyside` macros to use the `\imagewidth` approach without calling the `image` macro, i.e.,

```
\def\oneperpage#1#2{
  \clearpage \vspace*{\fill}
  \centering
  \includegraphics{#1}

  \medskip
  \settoheight\imagewidth{\includegraphics{#1}}%
  \begin{minipage}[b]{\imagewidth}
    \centering
    \large#2
  \end{minipage}
  \vfill
  \clearpage
}
```

```
\def\sidebyside#1#2#3#4{
  \clearpage \vspace*{\fill}
  \centerline{\includegraphics{#1}}%
  \quad\includegraphics{#3}

  \settoheight\imagewidth{\includegraphics{#1}}%
  \quad\includegraphics{#3}
  \smallskip
  \begin{minipage}[b]{\imagewidth}
    \centering
    Left: #2\Right: #4
  \end{minipage}
  %\centerline{Left: #2; right: #4}
  \vfill
  \clearpage
}
```

Notice that by this time I had created inconsistency in how I included images: sometimes

```
\macrocall
  {filename1}{caption1}
  {filename2}{caption2}
```

and sometimes

```
\macrocall
  {\macrocall{filename1}{caption1}}
  {\macrocall{filename2}{caption2}}
```

## 4.3 Photo album: Final approach

At this point, I concluded I needed to do several things differently:

1. try top justifying side-by-side images rather than bottom justifying them, which is what happened without doing anything special;
2. fix the page layout macros so they called images in a consistent way;
3. make it easy to move around the calls of image-caption pairs in the L<sup>A</sup>T<sub>E</sub>X source file.

Regarding the first point above, top justification of side-by-side images, I looked at or tried four different methods, three from a question and answer on [tex.stackexchange.com](http://tex.stackexchange.com)<sup>14</sup> and one I made up myself. One of the [tex.stackexchange.com](http://tex.stackexchange.com) suggestions didn't seem quite relevant and I couldn't manage to install and use the suggested packages in the other two suggestions there.

The method I tried to develop myself was to measure the height of the images, find the difference in heights as a positive number, and insert vertical space of that difference under the shorter of the images; unfortunately, I couldn't get the units of the various parts of these calculations to match well enough to make the method work. After several hours of trying things spread over a couple of days, the bottom-justified approach began to look better and better, and I gave up trying for top justification.

It came to me that dealing with the third point above (moving around image-caption pairs), would naturally address the second point (consistent calling sequences).

Regarding the third point above, it seemed to me that the best approach was to separate specifying images and their captions from the page layout macros, that is, to not have the page layout macros call the image-caption specification macros. My idea was to allow something like the following:

```
\specifyimage
\specifyimage
\layoutpagewithsidebysideimages
```

```
\specifyimage
\layoutpagewithsingleimage
```

```
\specifyimage
\specifyimage
\layoutpagewithoverunderimages
```

Then I could simply drag the `\specifyimage` macro calls around to the places I wanted them to be in my  $\LaTeX$  source file for the album, although I would still have to be aware of what size images could fit within the bounds of a page.

For the `\specifyimage` macro I developed the following macro (Karl Berry pointed out the  $\TeX$  language construct to me):

```
\newcounter{savedphotocount} % define counter
\setcounter{savedphotocount}{0}% clear counter

\def\savephoto#1#2{%
  \stepcounter{savedphotocount}%
  \ifcase\value{savedphotocount}
    \errmessage{case zero should never happen}%
  \or \gdef\photoa{#1}\gdef\captiona{#2}% case 1
  \or \gdef\photob{#1}\gdef\captionb{#2}% case 2
  \or \gdef\photoc{#1}\gdef\captionc{#2}% case 3
  \or \gdef\photod{#1}\gdef\captiond{#2}% case 4
  \else \errmessage{more photos than expected!}
  \fi}
```

This macro saves up to four images in well-known places from which the page layout macros can use them; obviously, this macro could have been extended to save more images between instances of zeroing `\savedphotocount`, which was done at the end of each page layout macro.

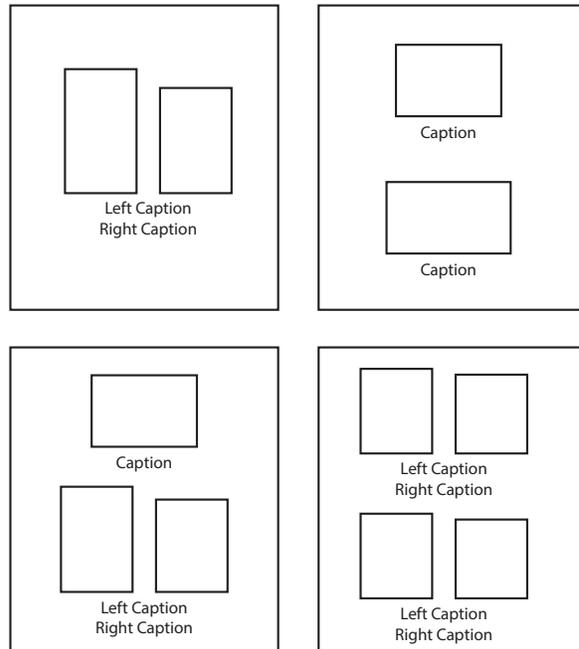
Below is an example definition of one of the page layout macros that used `\savephoto`.

```
\def\oneperpage{
  \clearpage \vspace*{\fill}
  \centering
  \includegraphics{\photoa}

  \medskip
  \settowidth\imagewidth
  {\includegraphics{\photoa}}%
  \begin{minipage}[b]{\imagewidth}
    \centering
    \large\captiona
  \end{minipage}
  \vfill
  \clearpage
  \setcounter{savedphotocount}{0}
}
```

Other page layout macros I needed to define for the album were:

`\overunder` two images centered horizontally, with equal top, between, and below spacing (see the top right example in Fig. 1).



**Figure 1:** Some examples of page layouts; example images from the actual album are available at [walden-family.com/texland/photo-album.pdf](http://walden-family.com/texland/photo-album.pdf)

`\sidebyside` two images side by side, with the pair centered horizontally, bottom justified, with equal top and bottom spacing (top left example).

`\oneovertwo` three images with the top one over the bottom pair, with equal top, between, and bottom spacing among the two rows of images, and the image and image-pair centered horizontally (bottom left example).

`\twooverone` reverse of the above.

`\twoovertwo` a side-by-side pair over another side-by-side pair with equal top, between, and bottom spacing among the rows, and the rows centered horizontally (bottom right example).

This may have not been the optimal approach in terms of requiring the writing a bunch of different page layout macros, but it was very useful in terms of flexibly moving images around within the  $\LaTeX$  file and experimenting with image ordering and page layouts until a satisfactory overall album layout was determined. If I was going to do lots of such albums, I might have wanted to include more calculations in the macros and let  $\LaTeX$  figure out how to lay out pages, but I didn't need that for this one case (but I do have a good starting point if I ever want to create something more automatic).

#### 4.4 Incorrectly sized JPGs

Through all of the above, I had to maintain some special versions of the macro for the 3.5×5 inch images with the tall orientation that `\includegraphics` was not displaying at the correct size in the PDF. This mostly resulting in these images being bigger than real size and thus with less pixels per inch than the desirable 300 pixel minimum.

Thus, I embarked on trying to figure out why `\includegraphics` wasn't correctly reading (or processing) the image size metadata from the image files. I could see no reason why not, and I asked on [tex.stackexchange.com](http://tex.stackexchange.com)<sup>15</sup> if someone knew how `includegraphics` read and processed JPG image metadata. The list tried to help but had no definitive and workable answers.

Eventually, I converted the problem JPGs to be PNGs, and then `\includegraphics` correctly sized the images. I don't know why this worked for PNGs and not for JPGs (the problem JPGs may have been originally scanned at 600 pixels per inch rather than 300 pixels per inch as was done for most of the images); however, now I have a work-around that I will try immediately if I see this problem again sometime.

#### 4.5 Finishing the album

With the work-around found for the problem noted in the prior subsection, I could now do a final compilation of the PDF for delivery to the print shop. There, although almost all the images were grayscale originally (with only a few color images), I had the print shop print them all in color which made the old brownish grayscale images look better than they would have using a black-and-white based grayscale.

I made the cover (front, spine, and back) of the photo album with Adobe Illustrator rather than  $\LaTeX$ . It might have been easier to do the whole layout with a graphical-user-interface (GUI) document layout tool. Then again it might have been more work with a GUI to match the caption widths to the images and to drag whole photo-caption pairs around (rather than dragging calls to `\savephoto` around in my text editor). Who knows? I have  $\LaTeX$ , and I don't have the alternative tool.

#### 5 Reflection

In this paper I have discussed three examples of little programming problems that I was led to by my work, and by the fact I was using  $\TeX$  and had been a programmer by trade.  $\TeX$  suits me particularly well as I dislike learning the ins-and-outs of user interfaces (especially user interfaces that change with product updates); and I often want something a little

different than is built into the user interface of a less programmable tool (or maybe what I want is built into a part of the tool I have not bothered to learn about). With  $\LaTeX$  I can get a surprising amount done with the relatively modest amount of  $\LaTeX$  programming I know (and packages that “just work” without much study), and over time I have built quite a library of ad hoc  $\LaTeX$  tools. The library is not very organized. I am careful to keep the sources for all my  $\LaTeX$ -based documents, and when I need a tool I try to remember for which document I developed that tool, and then I go and copy it.

#### Notes

<sup>1</sup> Perhaps I am a bad guy, but I lack motivation for developing my little tools into packages that might help others (although I certainly appreciate that others develop packages that help me).

<sup>2</sup> Self-publishing: Experiences and opinions, [tug.org/TUGboat/tb30-2/tb95walden.pdf](http://tug.org/TUGboat/tb30-2/tb95walden.pdf)

<sup>3</sup> While in this note I only discuss non-math use in American English, it provides enough variety of situations and problems to perhaps suggest things to think about when using and typesetting ellipses in another language.

<sup>4</sup> Version 3.1, Hartley & Marks, Publishers, Vancouver, BC, 2005

<sup>5</sup> I'm looking at the 13th edition and not a later edition.

<sup>6</sup> [en.wikipedia.org/wiki/Ellipsis](http://en.wikipedia.org/wiki/Ellipsis)

<sup>7</sup> [www.docstyles.com](http://www.docstyles.com)

<sup>8</sup> [ctan.org/pkg/ellipsis](http://ctan.org/pkg/ellipsis)

<sup>9</sup> [ctan.org/pkg/lips](http://ctan.org/pkg/lips)

<sup>10</sup> *A Culture of Innovation: Insider Accounts of Computing and Life at BBN*, David Walden and Raymond Nickerson, editors, Waterside Publishing, 2011, [walden-family.com/bbn/bbn-print2.pdf](http://walden-family.com/bbn/bbn-print2.pdf)

<sup>11</sup> Karl Berry and David Walden, editors,  *$\TeX$ 's*

<sup>25</sup> *Anniversary: A Commemorative Collection*,  $\TeX$  Users Group, Portland, OR, 2010, [tug.org/store/tug10](http://tug.org/store/tug10)

<sup>12</sup> David Walden and Tom Van Vleck, editors, *Compatible Time-Sharing System (1961–1973): Fiftieth Anniversary Commemorative Overview*, IEEE Computer Society, Washington, DC, 2011, [walden-family.com/ieee/ctss.pdf](http://walden-family.com/ieee/ctss.pdf)

<sup>13</sup> The reader may be interested in Boris Veytsman's approach for a somewhat similar project: “An output routine for an illustrated book”, *TUGboat* **35**:2, pp. 202–204, [tug.org/TUGboat/tb35-2/tb110veytsman.pdf](http://tug.org/TUGboat/tb35-2/tb110veytsman.pdf).

<sup>14</sup> [tex.stackexchange.com/questions/101858/make-two-figures-aligned-at-top](http://tex.stackexchange.com/questions/101858/make-two-figures-aligned-at-top)

<sup>15</sup> [tex.stackexchange.com/questions/171906/how-does-includegraphics-from-the-graphicsx-get-the-size-of-a-jpg-image](http://tex.stackexchange.com/questions/171906/how-does-includegraphics-from-the-graphicsx-get-the-size-of-a-jpg-image)

◇ David Walden  
[walden-family.com](http://walden-family.com)

## Glisterings: Lining up

Peter Wilson

The lines are fallen unto me in pleasant places; yea, I have a goodly heritage.

*The Bible, Psalm 16, v. 6*

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. This installment presents some items about lining things up.

### 1 Ruling off

Pujo wrote that he wanted to create a box with a line at the top and bottom but found that the `fancybox` package [10] only supplied boxes with all four sides enclosed. Peter Flynn [5] responded with the following, based on `fancybox`:

```
\documentclass{article}
\usepackage{fancybox,lipsum}
\newenvironment{ruledbox}{%
  \begin{Sbox}
  \begin{minipage}{\columnwidth}}{%
  \end{minipage}\end{Sbox}%
  \centering\medskip
  \vbox{\hrule height1pt
  \par\medskip
  \TheSbox
  \medskip\hrule height1pt}\par\medskip}
\begin{document}
\lipsum[1]

\begin{ruledbox}
\lipsum[2]
\end{ruledbox}
\lipsum[1]
\end{document}
```

Sometime later I wondered if a box was really needed, wouldn't just drawing a couple of rules do as well? I came up with the `ruled` environment which let you change the width of the ruled contents.

```
\newdimen\narrowsize
\newenvironment{ruled}[1][0pt]{%
  \par
  \narrowsize\hsize
  \advance\leftskip#1\advance\rightskip#1
  \advance\narrowsize-2\leftskip
  \noindent%
  \rule{\narrowsize}{3pt}\par
}{%
  \par\noindent
  \rule{\narrowsize}{1pt}
  \par}
```

The optional length argument to the environment is the distance the left and right margins should be increased, thus temporarily reducing the apparent width of the textblock. The next paragraph is set within

```
\begin{ruled}[1pc] ... \end{ruled}
```

The `ruled` environment produces a result that might be a little too fancy for your taste, in which case change the thickness of the rules.

On the other hand, a box will not break across a page boundary which may be an advantage, but on the whole I think not.

### 2 Marginal rules

David Arnold posed the following problem on `ctt`.

*I'd like to adjust my example environment in the code below so that each example is bracketed between two horizontal rules. The first rule should be placed above the example, align with the inner edge of the text and flow to the outer edge of the text, add a couple of spaces in the outer margin, typeset 'You Try It!', then continue to flow to within 1cm of the page edge.*

*Similarly, for the rule at the bottom of the example, I'd like to start it at the inner edge of the text, flow into the outer margin, then typeset the square that is flush right within 1cm of the paper edge.*

I'm not showing David's code here. Instead, the below is effectively the code that I responded with [9]. Drawing the rules across the textblock is no problem. Also typesetting in the margins can be catered for by using the `\rlap` and `\llap` macros, thus avoiding L<sup>A</sup>T<sub>E</sub>X getting huffy about overlong lines. The only tedious part of the code is calculating the length of the two rules in the margin area. Hopefully the comments in the code explain sufficiently what is done in this regard.

```
\documentclass[twoside]{report}
\usepackage{lipsum}
\usepackage{amssymb}

\newcounter{example}[section]
\renewcommand{\theexample}{\arabic{example}}

% insert lengths
\newdimen\uwidth
\newcommand*{\Utryit}{%
  \space\space You Try It!\space}
\settowidth{\uwidth}{\Utryit}
\newdimen\sqwidth
\newcommand*{\Usq}{\Large$\square$}
\settowidth{\sqwidth}{\Usq}
```

```

%% rule length in the oddpage margins =
%% paperwidth - textwidth - 1cm - 1in
%% - oddmargin - insert
% odd page rule lengths
\newdimen\uxtra % Utryit
\newdimen\sqxtra % Square
\uxtra=\paperwidth
\advance\uxtra-\textwidth
\advance\uxtra-1cm
\advance\uxtra-1in
\advance\uxtra-\oddsidemargin
\sqxtra=\uxtra
\advance\uxtra-\uwidth
\advance\sqxtra\sqwidth

%% rule length in the evenpage margins =
%% 1in + evenmargin - 1cm - insert
\newdimen\uxtrav % Utryit
\newdimen\sqxtrav % Square
\uxtrav=\evensidemargin
\advance\uxtrav 1in
\advance\uxtrav-1cm
\sqxtrav=\uxtrav
\advance\uxtrav-\uwidth
\advance\sqxtrav-\sqwidth

\makeatletter
\newenvironment{example}{%
\medskip\refstepcounter{example}%
\ifodd\c@page% odd page
  \noindent\rule{\hsize}{3pt}%
  \rlap{\Utryit\rule{\uxtra}{3pt}}
}else
  \noindent\llap{\rule{\uxtrav}{3pt}\Utryit}%
  \rule{\hsize}{3pt}
\fi
\par\noindent\textbf{Example \theexample.}}%
{%
\ifodd\c@page
  \par\noindent\rule{\hsize}{1pt}%
  \rlap{\rule{\sqxtra}{1pt}}
  \Usq}
}else
  \par\noindent\llap{\Usq\rule{\sqxtrav}{1pt}}%
  \rule{\hsize}{1pt}
\fi
  \par\medskip}
\makeatother

\begin{document}
\lipsum[1]
\begin{example}
\marginpar{Simplify: $33+28$}
\lipsum[2]
\end{example}
\lipsum[1]
\end{document}

```

The code just shown is intended for use in single column documents, and as *TUGboat* uses two columns it will not work here (account must be taken of which column the example is in). Extending it to cater for two columns is left as an exercise.

### 3 Preventing an awkward page break

Szabolcs Horvát requested help on *ctt*:

*I would like to have an environment that starts and ends with a horizontal line (`\hrule`), with text in smaller type in between. The text may run several pages long. How can it be prevented that the page be broken right after the first `\hrule` or right before the last one?*

As so often happens Donald Arseneau came up with an answer [1].

*The answer to the question is easy: insert `\par` and `\nobreak` and `\@nobreaktrue`.*

*The tricky problem is getting the spacing right! `\hrule` causes normal `\baselineskip` to be omitted, but `\rule` takes a full baseline which leaves too much whitespace. Try this:*

```

\makeatletter
\newenvironment{aside}{%
  \list{}{\leftmargin 5ex
    \rightmargin\leftmargin}
  \vtop{\hrule width\columnwidth}%
  \nobreak\@nobreaktrue
  \vspace{0.5ex}%
  \item\relax\small}
}{%
  \par\nobreak\@nobreaktrue
  \advance\baselineskip -0.7ex
  \vtop{\hrule width\columnwidth}%
  \endlist}
\makeatother

```

I tried the `aside` environment and it worked even better than requested as it kept a rule and at least two lines of text together.

I haven't tried to combine the `aside` and `ruled` environments which I leave as an interesting exercise.

From a slightly different viewpoint, Nick Urbanik posted to *ctt* that he wanted to keep a list of items always on a single page [8]. In particular, to keep a question and its suggested answers together, where there was a list of questions each with its list of answers. There were some six respondents to Nick's request for help but the discussion for some reason veered from the `enumitem` package to the `titlesec` package that had no relevance to the initial posting. Donald Arseneau again provided a simple solution to the original problem [3], resulting in questions and answers for a possible accountant's interview being coded like:

```

\textbf{Accountancy test}
\begin{questions}
\Qitem What is  $2+2$ ?
  \begin{enumerate}
  \item 3
  \item 4
  \item Whatever you want it to be.
  \end{enumerate}
\Qitem What is the essence of double-entry
  bookkeeping?
  \begin{enumerate}
  \item Each transaction recorded twice,
    in the credit and debit ledgers.
  \item Two sets of books, the real ones and
    the ones for the tax inspectors.
  \item Don't know.
  \end{enumerate}
\Qitem What ...
\end{questions}

```

When processed this will result in a question and all its potential answers being kept together on a page and, depending on their length, there may be several sets of questions and answers on a page.

#### Accountancy test

1. What is  $2 + 2$ ?
  - (a) 3
  - (b) 4
  - (c) Whatever you want it to be.
2. What is the essence of double-entry bookkeeping?
  - (a) Each transaction recorded twice, in both the credit and debit ledgers.
  - (b) Two sets of books, the real ones and the ones for the tax inspectors.
  - (c) Don't know.
3. What ...

Donald's method to make this happen is:

```

\newcommand*\Qitem{\pagebreak[0]\item}
\newenvironment{questions}%
{\enumerate\samepage}%
{\endenumerate}

```

which says that the `questions` environment should be all on one page except that a pagebreak is allowed just before a question's `\Qitem`.

#### 4 Not at a page break

Sometimes it may be desirable to have a divisional marker of some kind disappear at a page break. I have forgotten the details but someone once had a supplement (with a title such as 'Notes') at the end of each chapter in the document and wanted to have

a rule before the supplement unless the supplement started a new page.

---

A  $\TeX$  *leader* is not a permissible breakpoint and may vanish at a page break and so provides a potential means of meeting such a requirement. Just before this paragraph I specified:

```

\newskip\rulebreakskip
\rulebreakskip=\baselineskip
\newcommand*\filler{\hbox to \hsize{%
  \hss \rule{0.7\hsize}{1pt} \hss}\vskip 1pt}
\newcommand*\rulebreak{%
  \vskip\rulebreakskip
  \cleaders\filler
  \vskip\rulebreakskip}
\rulebreak

```

which resulted in either a centered rule or, if at the bottom of the column, nothing.

\* \* \*

Just before this paragraph I specified:

```

\renewcommand*\filler{%
  \hbox to \hsize{\hss * * * \hss}}
\rulebreak

```

which resulted in either three centered asterisks or, if at the bottom of the column, nothing.

You can put different elements in the `\filler` box, such as an `\asterisk` or a moustachio but you might have to adjust the value of `\rulebreakskip` for the best optical effect.

#### 5 Line backing

'talazem' presented `ctt` with a problem that has never been completely solved in  $\LaTeX$  — namely typesetting to a grid.  $\TeX$  was not designed with this in mind. Slightly edited, his presentation was:

*I am typesetting a book in Memoir and want to ensure that the lines register well to avoid shine through. The book is mainly in English with a font size 10/12.5. However there are some paragraphs that are causing alignment problems.*

*There are some paragraphs that have to be set to a 0.8 ratio of the primary face with a 0.5 ratio of line spacing. There are others in a non-English typeface where the font is about 1.4 times bigger than the Roman font for the English text.*

*Paragraphs of this kind throw off the alignment of text lines on adjacent pages, and causing shine through on the recto and verso sides of a page.*

The basic requirement here is that these irregular paragraphs should take up a space that is an integral number of the normal `\baselineskip`.

The one potential solution provided came from an exchange of views between Donald Arseneau and Dan Luecking [4], as follows, where the environment will occupy an integral number of the normal lines.

```
\makeatletter
\ifundefined{@tempdimc}{\newdimen@tempdimc}{}
\newenvironment{gridblock}{\par
  \setbox@tempboxa\top\bgroup
}{\par\egroup
% measurement of top
  \@tempdima=\ht@tempboxa
  \@tempdimc=\dp@tempboxa
  \ifdim@tempdima>\ht\strutbox
    \advance@tempdimc@tempdima
    \@tempdima=\ht\strutbox
% \@tempdima is the top height.
    \advance@tempdimc-@tempdima
  \fi
% measurement of bottom
  \setbox@tempboxa\vbox{\unvbox@tempboxa}%
  \ifdim\dp@tempboxa>\dp\strutbox
    \@tempdimb=\dp\strutbox
  \else
    \@tempdimb=\dp@tempboxa
  \fi
% \@tempdimb is the bottom depth.
  \advance@tempdimc-@tempdimb
% \@tempdimc is distance between the top
% and bottom baselines.
% The excess, @tempcnta, is the number
% of baselines.
  \@tempcnta=@tempdimc
  \divide@tempcnta\baselineskip
  \advance@tempdimc -@tempcnta\baselineskip
  \ifdim@tempdimc >2\vfuzz
    \advance@tempdimc-\baselineskip \fi
  \divide@tempdimc\tw@
  \vbox to@tempdima{}%
  \nobreak \nointerlineskip
  \kern-@tempdima \kern-@tempdimc \nobreak
\box@tempboxa
\nobreak \nointerlineskip
  \kern-@tempdimb \kern-@tempdimc \nobreak
\hbox{\vrule
  height \z@ width \z@ depth @tempdimb}}
\makeatother
```

The `gridblock` environment doesn't cater for footnotes, floats, or really anything other than plain text. It certainly does *not* handle page breaks.

This is \tiny text in the gridblock environment. I'm not sure how well the effect will be demonstrated as the adjacent column may, or may not, be evenly spaced vertically.

Did that work out? Are the lines in this paragraph aligned with those on the adjacent columns, or pages? If not it may be because the adjacent columns are not set on a grid. Incidentally, the relatively recent package `grid` may be of interest, though it is not a complete solution either.

Peter Wilson

## 6 Linespacing

Pander wrote [7]:

*I have some questions on line spacing (leading) that should respect font size. It mainly concerns non-uniform line spacing that doesn't reserve space for ascenders and descenders and line spacing that is too big or too small for small and large font sizes.*

*Please see the following TeX [code] for the exact questions. I know this is tricky in TeX, but have to ask any way.*

```
\noindent
{\tiny aeou\aeou\}%too much leading
{\normalsize aeou\aeou\}
{\Huge aeou\aeou\}%not enough leading
{\tiny gpqy\gpqy\}%too much leading
{\normalsize gpqy\gpqy\}
{\Huge gpqy\gpqy\}%no space for descenders
{\tiny bdfhkl\bdfhkl\}%too much leading
{\normalsize bdfhkl\bdfhkl\}
{\Huge bdfhkl\bdfhkl\}%no space for ascenders
{\tiny gpqybdfhkl\gpqybdfhkl\}%too much leading
{\normalsize gpqybdfhkl\gpqybdfhkl\}
{\Huge gpqybdfhkl\gpqybdfhkl\}
```

The result of processing this is shown in the left side of Figure 1. Pander also noted a similar problem when using different fonts in a `tabular`.

There were several respondents all of whom noted that Pander's example consisted of a single paragraph within which the several font size changes were closed within groups. Further, that  $\TeX$  takes the font size in effect at the end of a paragraph as applying throughout the paragraph, and hence that the leading is constant.

Donald Arseneau [2] replied with:

*Set \baselineskip=0pt or some small value  
Set \lineskip=\lineskiplimit= desired space*

```
\baselineskip=8pt
\lineskip=4pt
\lineskiplimit=\lineskip
```

*Then be aware that font-change commands reset \baselineskip, so that font changes that span the end of a paragraph will go back to some larger \baselineskip.*

*In tabular put \strut in with all your variant fonts.*

Roughly speaking, the normal spacing between the baselines of text is `\baselineskip` but if the 'top' of a line is closer than `\lineskiplimit` to the bottom of the previous line then the spacing will be increased so that the top to bottom space is `\lineskip` [6, Ch. 12]. The results of applying Donald's settings are shown at the right of Figure 1.



**Figure 1:** Different font sizes in a paragraph: (left) Pander's problem; (right) following Donald Arseneau

## References

- [1] Donald Arseneau. Re: Preventing page breaks at certain positions. Post to `comp.text.tex` newsgroup, 17 November 2009.
- [2] Donald Arseneau. Re: Line spacing respecting space for ascenders / descenders and fontsize. Post to `comp.text.tex` newsgroup, 11 April 2011.
- [3] Donald Arseneau. Re: List items always on the same page. Post to `comp.text.tex` newsgroup, 15 March 2011.
- [4] Donald Arseneau and Dan Luecking. Re: vertical height of boxes by multiple of `baselineskip`. Post to `comp.text.tex` newsgroup, 9–10 December 2009.
- [5] Peter Flynn. Re: Fancybox alternatives. Post to `comp.text.tex` newsgroup, 11 September 2009.
- [6] Donald E. Knuth. *The T<sub>E</sub>Xbook*. Addison-Wesley, 1984. ISBN 0-201-13448-9.
- [7] Pander. Line spacing respecting space for ascenders / descenders and fontsize. Post to `comp.text.tex` newsgroup, 11 April 2011.
- [8] Nick Urbanik. List items always on the same page. Post to `comp.text.tex` newsgroup, 13 March 2011.
- [9] Peter Wilson. Re: Marginpar in memoir. Post to `comp.text.tex` newsgroup, 27 December 2009.
- [10] Timothy Van Zandt. `fancybox.sty`: Box tips and tricks for L<sup>A</sup>T<sub>E</sub>X, September 2000. <http://ctan.org/pkg/fancybox>.
  - ◇ Peter Wilson
  - 12 Sovereign Close
  - Kenilworth, CV8 1SQ
  - UK
  - herries dot press (at)
  - earthlink dot net

## CTAN goes multi-lingual: Additional language support for the Web portal

Gerd Neugebauer

### Abstract

$\text{\TeX}$  is used for many languages: support in the  $\text{\TeX}$  engines and macro packages is abundant, but the CTAN portal has been available in English only. Now we are conducting an experiment to provide the Web presentation in German as well.

### 1 Introduction

Modern Web frameworks are equipped with means for internationalizing the presentation. We are currently attempting to make use of these means for the CTAN portal, <https://www.ctan.org>.

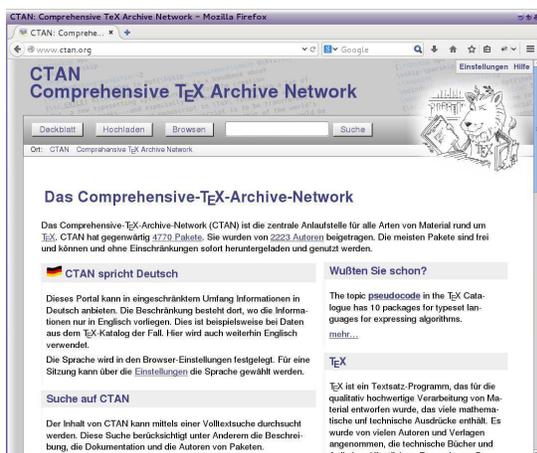


Figure 1: The cover page in German

Whenever you visit the portal, the language is automatically selected for you. This magic happens in a negotiation between the browser and the server. In the browser you can configure your preferred languages (cf. figures 2–4).



Figure 2: Language configuration in Firefox

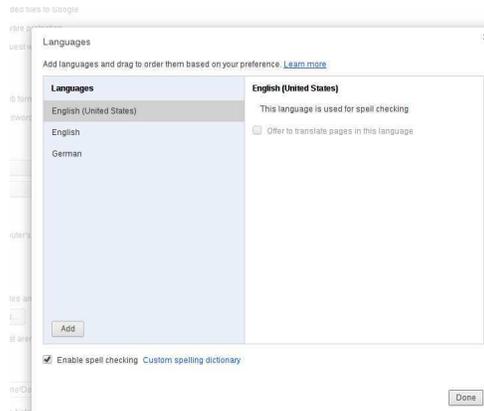


Figure 3: Language configuration in Chrome

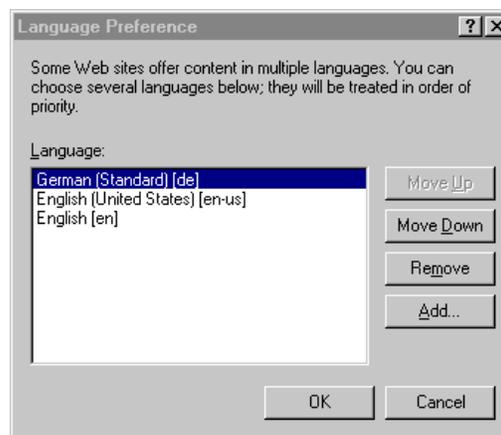


Figure 4: Language configuration in IE

When you request a page, the browser passes these language preferences to the server, where the list is compared with the list of supported languages. The best fit — or the default fallback — is then chosen. Currently, the CTAN portal supports English and German. Thus it is usually sufficient for you to navigate to the CTAN portal to see your preferred language from these alternatives.

You can also explicitly choose the language on the settings page by clicking on the appropriate flag (figure 5). This selection is valid only for the current session. When you return later, this setting is forgotten.

### 2 Features

Internationalization includes the localized presentation of several types of content:

- The page frame which includes header and footer as well as the static parts of the dynamic pages
- The page content for the “static” pages
- The dynamic content from the  $\text{\TeX}$  archive

portal (skinning) to fit to your personal taste. This feature can be used on the CTAN portal and without being logged in.

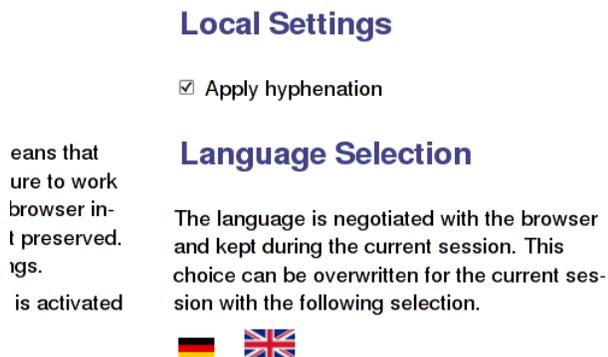


Figure 5: The language selection on the settings page

- The dynamic content from the Catalogue
- The dynamic content from the search index

For the current internationalization experiment, only the frame and some of the static pages are offered in German, as a starting point. It should primarily help to make it more comfortable for visitors in their first steps.

Some pages have intentionally not been translated. For instance, the upload page (<https://www.ctan.org/upload>) is provided in English only since the communication language with the CTAN team is English. Thus, an uploader should not be encouraged to try another language since this might not be understood on our side.

### 3 Limitations

The major limitations are derived from missing data in the back-end. The files in the  $\TeX$  archive are provided by the numerous authors. The language is whatever the author provides — for instance in the package documentation.

The  $\TeX$  catalogue is mainly in English. It is partially prepared to carry texts in different languages. For instance the descriptions are to a small degree already present in languages other than English. The CTAN team is presently not able to provide translations, due to limited resources and skills.

If no appropriate localized text is available the English version is used. This can be seen in figure 1. Here the text for the topic teaser is in English while the other parts of the page are in German.

Another limitation applies to the search. The search is currently language-agnostic. It should in the future be based on the language settings, in that entries with the proper language should be ranked higher than non-matching entries. Maybe entries

for languages not understood by the user should be suppressed completely. But this already belongs into the next section on “visions”.

### 4 Visions

In the future we could consider to develop the internationalization of the CTAN portal in several directions. First, of course there are more languages to be supported. Right now the language-specific texts are contained in 31 files. These files are either mapping files which map keys to language-specific texts, or complete pages which are present in separate incarnations.

To support a new language is initially a matter of providing these 31 files and adding a configuration option for the new language. However, this isn’t enough. The portal is not static. The content changes over time. Thus we would need a commitment that we have volunteers for the new supported language to guarantee continuity.

For the support of even more languages the existing administration interface could be extended. Then maintenance of the different languages could be performed via a Web interface by different persons.

The other side has already been mentioned. The  $\TeX$  catalogue has to be augmented with new language-specific texts. The same applies here. The support has to be guaranteed for future changes. Thus volunteers for a long-term engagement would be required.

We must also reconsider the processes in the CTAN team. They are currently not designed for parallel processing of a change for a single package. These processes work well for the small team which is currently active. The more languages that are supported, the more people have to work on one change. Unfortunately this seems to be beyond the capabilities of the CTAN team right now.

### 5 Epilogue

I hope that the experiment with the German language for the CTAN portal succeeds. Feedback in any direction is welcome. For discussions, please consider using the mailing list [ctanweb@dante.de](mailto:ctanweb@dante.de). You can subscribe via <https://lists.dante.de/mailman/listinfo/ctanweb>.

Keep on  $\TeX$ ing — in many languages.

◇ Gerd Neugebauer  
 Im Lerchelsbühl 5  
 64521 Groß-Gerau (Germany)  
[gene@gerd-neugebauer.de](mailto:gene@gerd-neugebauer.de)  
[www.gerd-neugebauer.de](http://www.gerd-neugebauer.de)

## Obyknovennaya Novaya (Ordinary New Face) in METAFONT

Basil Solomykov

The Obyknovennaya Novaya (Ordinary New Face) typeface was widely used in the USSR for scientific and technical publications, as well as textbooks. My “Obyknovennaya Novaya” is a revival of that typeface, and though it is not the first one, I believe it is the most complete. The Obyknovennaya Novaya family currently includes regular, bold, italic, bold italic, slanted and small capitals shapes. Obyknovennaya Novaya is free software, available under the terms of the LPPL. The story of the METAFONT version of this font follows . . .

In the beginning of 2008 I was a student and needed to choose the theme for my qualifying work. My scientific supervisor was Vladimir Lidovski, and he advised to learn METAFONT and make the font Obyknovennaya Novaya in METAFONT, to expand the variety of available Cyrillic fonts in T<sub>E</sub>X. I began to read Donald Knuth’s *The METAFONTbook* and make my first steps in drawing and font making. After approximately a month, I made the first letter — at that moment it was a big success to me. Over the next month I learned about main parameters, such as stems, curves, bars and others. The “army” of my letters was growing, and it inspired me to continue my work. By the time the number of letters reached 50, I had learned about some new typographic features.

I read Knuth’s Volume E of *Computers & Typesetting*, which contains precise definitions of about 500 letters, numerals, and other symbols of the Computer Modern Typefaces, all described with METAFONT. I realized that my letters had different parameters, each letter was described in its own file, for example `LetterA.mf`, without any unification. So I decided to combine them by making one file for all letters of one size, and began a large amount of work to restructure my font.

By the end of May 2008 I had 66 Cyrillic letters (33 capitals and 33 small), 52 Latin letters, numerals and some punctuation marks, all in one shape (regular), at several point sizes: 7, 10, 12, 17 pt. I met with Alexander Shen, who supported me and observed new directions to improve my typeface. I successfully graduated from the university and then it was time to decide what to do next with my project. At that time the development of the Obyknovennaya Novaya typeface became supported by the TUG development fund, so I began the long journey of making a high-quality font.

There were big plans for the future to make italic, bold, bold italic and slanted shapes, work with rounding errors, kerning and others. In spite of my no longer officially being his student, Vladimir Lidovski continued testing my font, made suggestions and gave advice. It was partly this support that kept me from throwing everything down. During this time I also met Alexander Tarbeev, who gave me some valuable advice regarding the overall design and relationship between the different parameters of the typeface.

By the spring of 2011 I had made regular, bold, italic, bold italic, slanted and small capital shapes. Some work to optimize font rendering was done, and new kerning pairs had been added.

At that point, I stopped working with the font until 2014, and began to learn about other font formats, such as TrueType, OpenType and PostScript Type 1.

I read a lot about font smoothing and rendering on digital devices. Beat Stamm’s site (<http://www.rastertragedy.com>) was especially useful for me; it gives a detailed description how fonts are rendered on the screen. I am trying to understand how to apply this method in Metafont, and hope to implement it in the future.

In general the work on this font has been a fruitful experience for me. It is my first contribution to CTAN and the L<sup>A</sup>T<sub>E</sub>X community. I hope the font will be useful and I am glad to be able to help people. I also hope that the publication of my font will provide more feedback and suggestions from experienced T<sub>E</sub>X users on how to improve it.

◇ Basil Solomykov  
bs44550 (at) gmail dot com  
<http://ctan.org/pkg/obnov>

Normal shape, (10pt): АБВГД...ЭЮЯ абвгд...ьбэюя  
0123456789 ABCD...WXYZ abcd...wxyz ?@&\*№

Bold shape, (10pt): АБВГД...ЭЮЯ абвгд...ьбэюя  
0123456789 ABCD...WXYZ abcd...wxyz ?@&\*№

Italic shape, (10pt): АБВГД...ЭЮЯ абвгд...ьбэюя  
0123456789 ABCD...WXYZ abcd...wxyz ?@&\*№

Bold italic, (10pt): АБВГД...ЭЮЯ абвгд...ьбэюя  
123456789 ABCD...XYZ abcd...wxyz ?@&\*№

Slanted shape, (10pt): АБВГД...ЭЮЯ абвгд...ьбэюя  
0123456789 ABCD...WXYZ abcd...wxyz ?@&\*№

Small Capitals, (10pt): АБВГД...ЭЮЯ АБВГД...ЬБЭЮЯ  
0123456789 ABCD...WXYZ ABCD...WXYZ ?@&\*№

## A simple Arabic typesetting system for mixed Latin/Arabic documents: *dād*

Yannis Haralambous

### Abstract

We describe *د* (*dād*), a package allowing simple typesetting in Arabic script, intended for mixed Latin-Arabic script usage, in situations where heavy-duty solutions are discouraged. The *د* system operates with both Unicode and transliterated input, allowing the user to choose the most appropriate approach.

### 1 Introduction

As with many  $\text{\TeX}$  projects, this one was started to fulfill an immediate need: the author was writing a paper for an Arabic language Natural Language Processing conference [7] and hence was in need of a straightforward way to introduce Arabic text into his document. In this context, “straightforward” can be subdivided into the following five requirements:

1. it should be compatible both with the IEEE  $\text{\LaTeX}$  style [19] (required by the conference) and with the Write $\text{\LaTeX}$  platform [3], of which the author is an enthusiastic user;
2. it should allow user-friendly and robust input of Arabic text, including when placed inside  $\text{\TeX}$  command arguments;
3. it should typeset in an optimal way all combinations of letters and diacritics that may appear in scholarly text;
4. it should provide some extra features: being able to easily change the letter form, as well as to colorize specific letters without breaking contextual analysis;
5. the font should be easily readable in a context of mixed Latin-Arabic script.

In the following discussion we will see why existing systems did not fulfill the requirements, and how the author solved the problem.

#### 1.1 Existing systems and their pitfalls

As we live in the 21st century, an obvious choice for typesetting Arabic in  $\text{\TeX}$  is  $\text{Xe}\text{\TeX}$  [12], a  $\text{\TeX}$  avatar (in)famous for typesetting in non-Latin scripts by taking advantage of operating system resources.

**Requirement 1**  $\text{Xe}\text{\TeX}$  indeed is provided on the Write $\text{\LaTeX}$  platform. But the  $\text{\LaTeX}$  overhead for typesetting Arabic in  $\text{Xe}\text{\TeX}$  is quite heavy (packages `fontspec`, `xunicode`, `arabxetex`, etc.) and hence, not surprisingly,  $\text{Xe}\text{\TeX}$  is incompatible with the IEEE style.

**Requirement 2**  $\text{Xe}\text{\TeX}$  is unable to use a transliteration system and hence requires the Arabic text

to be input in Arabic script and *only* in Arabic script. Using a transliteration to input Arabic may seem terribly old-fashioned to the reader, but there are cases where it is the best solution. One of these cases is the context of this paper: a mixture of Latin script, Arabic script, and  $\text{\TeX}$  commands.

Indeed, at the GUI level, there are — at least — two drawbacks in combining Arabic and Latin script in the same paragraph:

1. the use of the cursor and of left and right arrow keys is very cumbersome: when you select a location with the cursor you don’t know whether you are in right-to-left or left-to-right mode and hence you don’t know in which direction to advance, or how to select a given character string;
2. the situation is made even worse by the fact that some punctuation marks (period, exclamation mark, dashes, parentheses, braces, brackets, etc.) are common to the two scripts and hence the — quite sophisticated — bidirectional algorithm is used to determine whether a punctuation mark is to be placed on the left or on the right of an Arabic word. The bidirectional algorithm (or ‘bidi’ for the insiders) is both a blessing and a curse. It is a blessing because it puts some order in the rendering of mixed right-to-left and left-to-right texts (cf. [6, p. 133–146]) — but this works well only if RLE and LRE are used to indicate the embedding level. Otherwise, in everyday use, bidi is a curse. Fig. 1 shows how the  $\text{\TeX}$  code for writing the word `كتب` with the middle letter colorized in red is displayed by various programs under various operating systems; not a single one of them really makes sense.

**Requirement 4** Sophisticated OpenType fonts [6, §D.9.4] handle relatively well the many letter + diacritic combinations, and most systems (including  $\text{Xe}\text{\TeX}$ ) can colorize word parts without breaking contextual analysis through the use of the zero-width joiner character [6, p. 104]. But not a single system is able to colorize single letters inside  $\text{\textcircled{y}}$ , since this ligature has always been considered as a single glyph by font designers.

**Requirement 5** The best way to match Latin and Arabic script is to choose an Arabic font with relatively small differences in height between letters. A quite common choice is the font *Geezah* by Diwan Software Ltd (developed for Apple WorldScript in the early nineties, and still included in Mac OS X, through today). *Geezah* is a nice font but its diacritics are placed rather suboptimally, and modifying their positions requires a high amount of competence in fiddling around with OpenType features.



ligatures between characters. In this case, the default letter form is the isolated one. The presence of a second letter changes the form of the first from isolated to initial, and the one of the second from isolated to final. A third letter will turn the second letter into middle form and the third letter into final (provided, of course, the second letter is quadriform), and so on. More details are given in § 4.

## 2 The name

The next question was what to name the package.

Thanks to the Internet, search engines, social media, and the like, people are becoming more and more aware of other languages and writing systems. Why not give this package an Arabic name, be it a single letter?

The author has chosen the letter ض, called *dād*, because Arabic is traditionally called the “language of the *dād*”, since this sound was historically considered as being unique to Arabic.

The reader is probably wondering how to pronounce this letter, technically a “voiced velarized alveolar stop” [18, p. 16]. Here is how [20, p. 10] describes its pronunciation:

Pronounce the regular sound ‘d’ and you will find that the tip of your tongue will touch in the region of the upper front teeth/gum. Now pronounce the sound again and at the same time depress the *middle* of the tongue. This has the effect of creating a larger space between the tongue and the roof of the mouth and gives the sound produced a distinctive ‘hollow’ characteristic, which also affects the surrounding vowels. It is difficult to find a parallel in English, but the difference between ‘Sam’ and ‘psalm’ (standard English pronunciation) gives a clue. Tense the tongue muscles in pronouncing ‘psalm’ and you are nearly there. Now pronounce the a-vowel of ‘psalm’ before and after ‘d’, saying ‘aḍa’, keeping the tongue tense, and that’s as near as we can get to describing it in print.

## 3 How to use ض

The package provides three PostScript Type 1 fonts (plain, bold and typewriter), “real” fonts (regular TFM) and large virtual fonts (OVF and OFM files). There are also rudimentary FD and STY files, a MAP file, Perl scripts for conversion to (and from) UTF-8, the Perl script which builds the font and finally adjustment files, in case the user wants to change kerning and diacritic placement.

It requires LuaTeX for change of direction and OVF/OFM compliance.

To typeset in Arabic, one need only load the `dad` package and use the macro `\arab`, which is a `\long` macro: its argument may have multiple paragraphs.

Arabic text can be input in transliteration as described in Table 1 or in UTF-8. To obtain, for example, الكِتَاب one would write `\arab{AlkitAb}` or `\arab{الكِتَاب}`. By writing `\arabtt{AlkitAb}` one obtains the typewriter version الكِتَاب (which is less appealing, but fits quite nicely with the Computer Modern Typewriter font).

### 3.1 Rationale of the transliteration

Here are the rules of the proposed transliteration (see Table 1):

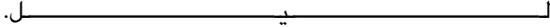
1. pharyngeal ح = H, emphatic ص = S, ض = D, ط = T, ظ = Z and velar غ = R are *uppercased* — do not confuse them with glottal ه = h, non-emphatic س = s, د = d, ت = t, ز = z, and alveolar ر = r;
2. long vowels (ا = A, و = U, ي = Y) and *ʿalif maqṣūra* (ى = I) are also *uppercased*;
3. some consonants are modified by adding a character h (ذ = dh, ث = th, ش = sh);
4. the stand-alone *hamza* is obtained by a vertical bar | and letter *ʿayn* by a grave accent (which, in legacy TeX produces an inverted curly apostrophe, which is sometimes used to transliterate this letter);
5. to avoid confusion between pairs of letters and letters obtained by digraphs, one has to use a dash to separate characters: compare سه = s-h and ش = sh, or ته = t-h and ث = th;
6. more generally, the dash plays the rôle of *zero-width joiner*:<sup>1</sup> when writing ب = -b, the letter *bāʿ* will be in final form; ٣ = b- and ٣ = -b- will produce initial and middle letters, provided of course the letter is quadriform (as is letter *bāʿ* in this example). This is very useful when describing grammar rules, to signify that a letter (or letter group) is an affix;
7. the dash can also be used to reestablish contextual forms when combined with TeX commands, for example, to colorize letters as in Fig. 1. There is only one special case: when we want to colorize a letter of an isolated ligature ٣, we add a digit 4 in front of the dash. For the final ligature ٣ it will be a digit 5. Example: to colorize the *lāms* of لآلآ, write

```
\arab{t-\textcolor{red}{-15-}-A5%
\textcolor{red}{14-}-A4}
```

<sup>1</sup> Except for the case of letter ذ = dh which is bifurcated and hence is not connected with the following letter. By writing د = d-h one obtains letters *dāl* and *hāʿ*, but the *hāʿ* is not in medial form, as it would be in any other case when preceded by a dash.

Table 1: Transliteration of ض system

ء		آ	'A	أ	'a	ؤ	'u	إ	'i	ئ	'I
ا	A	ب	b	ة	t*	ت	t	ث	th	ج	j
ح	H	خ	x	د	d	ذ	dh	ر	r	ز	z
س	s	ش	sh	ص	S	ض	D	ط	T	ظ	Z
ع	'	غ	R	ف	f	ق	q	ك	k	ل	l
م	m	ن	n	ه	h	و	U	ى	I	ي	Y
آ	A*	ا	o	ا	a	ا	i	ا	u	ا	aN
ا	iN	ا	uN	ا	+	ا	+a	ا	+i	ا	+u
ا	+aN	ا	+iN	ا	+uN	ا	a*	ا	+a*	الله	LLh
پ	p	گ	g	چ	C	ژ	J	ه	e	ف	v
ب	'b	ن	'n	ف	'f	ق	'q	ا	a**	ا	+a**

8. finally, there is yet another use of the dash: when doubled, it produces a kashida stroke: compare ليل = 1Y1 and ليل = 1--Y--1. There is also a `\kesh` command for extensible kashida (equivalent to a `\hrulefill` using the default rule thickness font dimension `\fontdimen8`): `1--\kesh--Y--\kesh--1`. produces: 
9. some digraphs start with an apostrophe: the hamza-carriers  $\dot{a}$  = 'a,  $\dot{i}$  = 'i,  $\dot{u}$  = 'u,  $\dot{I}$  = 'I,  $\dot{A}$  = 'A and also undotted letters  $\dot{b}$  = 'b,  $\dot{n}$  = 'n,  $\dot{f}$  = 'f and  $\dot{q}$  = 'q;
10. other digraphs end with one or more asterisks: the most frequent one is the  $\dot{t}$  *marbuṭa*  $\dot{t}$  = t\* (which can be used also in initial and medial forms, and then becomes a regular  $\dot{t}$ ). The asterisk is also used for the *waṣla* (which is only placed on the *ʿalif*)  $\dot{a}$  = A\* as well as for the vertical *fatha* (as in  $\dot{h}$  = ha\*dhA) and the *madda*. The latter is normally used only on the *ʿalif* ( $\dot{a}$  = 'A) but can be found also in the notorious *muqattaʿat* in the Koran, as in عَسَقَ (Koran 42:2) or كَيْبَعَصَ (Koran 19:1) — sometimes it is even combined with a *šadda* (as in الْمَصْرَ, Koran 7:1 and [21, p. 111] for the *šadda*);
11. a special transcription is provided for the ligature  $\text{الله}$  = LLh used for the *اسم الجلالة* “noun of majesty”, which is the name of God  $\text{الله}$ : in

this case — and in this case only — an uppercase L is used. The reason is that we wish to avoid ambiguity with other uses of the trigram *lām-lām-hāʿ*, for example  $\text{يُضَلِّلُهُ}$  (Koran 6:39) where we encounter letters  $\text{الله}$  but not with the meaning “God”. In contrast to other systems, the  $\text{الله}$  ligature is available also in final form (for  $\text{فَالله}$  which occurs six times in the Koran, for example Koran 6:149), and it is possible to add diacritics to its first glyph (as in  $\text{وَالله}$ , Koran 2:115 or  $\text{لله}$ , Koran 2:165).

### 3.2 Unicode input

Input can be transliterated or provided directly in Unicode Arabic: `\arab{YAnis}` or `\arab{يانيس}` or even `\arab{يانيس}` or `\arab{YAnis}` will produce the same result: يانيس.

All cells of Table 1 can be obtained by the corresponding Unicode characters (mostly via a single character, except for *šadda* + vowel combinations which require two characters). There is a special case, though: the  $\text{الله}$  ligature (see next section).

For the convenience of the user who wants to write kashida (so that Arabic input is not disrupted) we have defined a command (in Arabic characters)  $\text{ا\kesh}$  (ا\kesh are the first two letters of *tatwīl* = *tatwīl*, the Arabic name of kashida) which is exactly equivalent to `\kesh` and has to be placed between Unicode U+0640 ARABIC TATWEEL characters.

**Weak.** Weak verbs are those with one or more weak letters (و or ي) as radicals. There are four sub-classes:

- **Assimilated.** Assimilated verbs have *initial* و or (much more rarely) ي, and two sound radicals or middle ء and a sound final radical. Typical doubled roots are وصل, يبس.
- **Hollow.** Hollow verbs have *middle* و or ي and two sound radicals or initial ء and a sound final radical. Typical hollow roots are قول, صير.
- **Defective.** Defective verbs have *final* و or ي and two sound radicals. Typical roots are رمي, رجو.
- **Doubly weak.** Doubly weak verbs have two weak radicals, و, ي or ء. Typical doubly weak roots are سوء, رأي, أتي, سوى, ولي.

In the middle of a verse *hamzah* is merged with the final vowel of the preceding word, e.g., كَهَلَقَ، الْإِنْسَانَ، He created man. Note that the قَ or كَهَلَقَ has joined the لَ or الْإِنْسَانَ and while the *hamzah* sign has disappeared from the text, *'alif* is retained but it is not pronounced.

#### Practice text 11

1. Man says — قَالَ الْإِنْسَانُ
2. Does man think? — أَيْحَسِبُ الْإِنْسَانُ
3. He said: How long hast thou tarried? — قَالَ كَمْ لَبِثْتَ
4. The truth is out — حَصْحَصَ الْحَقُّ
5. He created man from dry clay — كَهَلَقْنَا الْإِنْسَانَ مِنْ صَلْصَالٍ

```

\documentclass{article}
\usepackage{dad}
\begin{document}

\textbf{Weak}. Weak verbs are those with one
or more weak letters (\arab{U} or \arab{Y})
as radicals. There are four sub-classes:
\begin{itemize}
\item\textbf{Assimilated.} Assimilated verbs
have \emph{initial} \arab{U} or (much more
rarely) \arab{Y}, and two sound radicals or
middle \arab{I} and a sound final radical.
Typical doubled roots are \arab{Ybs},
\arab{USl}.

\item\textbf{Hollow.} Hollow verbs have
\emph{middle} \arab{U} or \arab{Y} and two
sound radicals or initial \arab{I} and a
sound final radical. Typical hollow roots
are \arab{qUl}, \arab{SYr}.

\item\textbf{Defective.} Defective verbs have
\emph{final} \arab{U} or \arab{Y} and two
sound radicals. Typical roots are \arab{rjU},
\arab{rMY}.

\item\textbf{Doubly weak.} Doubly weak verbs
have two weak radicals, \arab{U}, \arab{Y}
or \arab{I}. Typical doubly weak roots
are \arab{ULY}, \arab{sUI}, \arab{'atY},
\arab{r'aY}, \arab{sU}.

\end{itemize}

In the middle of a verse \emph{hamzah} is
merged with the final vowel of the preceding
word, e.g., \arab{khalafa Alo'iinosaAna},
He created man. Note that the \arab{qa}
or \arab{khalafa} has joined the \arab{lo}
or \arab{'aalo'iinosaAna} and while the
\emph{hamzah} sign has disappeared from the
text, \emph{'alif} is retained but it is not
pronounced.\[6pt]
\textbf{Practice text 11}\[6pt]
1. Man says --- \arab{qaAla Alo'iinosaAnu}\[6pt]
2. Does man think? --- \arab{'aaYaHosabu
Alo'iinosaAnu}\[6pt]
3. He said: How long hast thou tarried? ---
\arab{qaAla kamo labithota}\[6pt]
4. The truth is out --- \arab{HaSoHaSa
AloHaq+u}\[6pt]
5. He created man from dry clay ---
\arab{khalafanaA Alo'iinosaAna mino
SaloSaAliN}\[6pt]

\end{document}

```

Figure 2: Sample L<sup>A</sup>T<sub>E</sub>X document using ض ([16, p. 5] and [21, p. 54–55])

### 3.2.1 The الله ligature and Unicode

The الله ligature is traditionally used for writing the name of God: الله. It can be found in religious texts, but also in expressions (for example, إن شاء الله which means “hopefully” appears even in the French language as *inchallah* and in Portuguese as *oxalá*) and in the very common surname عبد الله Abdullah.

The problem with this ligature is that it contains a rather rare diacritic (a *šadda* combined with a vertical *fatha* — the latter is available in the Apple Arabic keyboard layout but not the Microsoft one) and, as a convenience, most standard fonts will replace the character string *lām-lām-hāʾ* (which would normally look like الله) by the complete ligature الله; in other words: the font not only changes the glyphs but, at the same time, also adds the diacritics. This behavior is barely legitimate: a ligature (as in ‘fi’ or ‘y’) is normally limited to a change of glyphs, and should not add new characters (in this case, characters U+0651 ARABIC SHADDA and U+0671 ARABIC LETTER SUPERSCRIPT ALEF) since this means that what is rendered no longer corresponds to the underlying Unicode character string.

Nevertheless, for the user’s convenience, we have adopted that behavior also in ض, but only in the case of Unicode input. Therefore when the user types Unicode *lām-lām-hāʾ* (the first *lām* must not be preceded by a quadriform letter), the system will produce the الله ligature.

This method will not work if a diacritic is inserted between the two *lāms*, or if the first *lām* follows a quadriform letter and hence will be medial. For that case, we have defined a macro الله/ (the macro name is in Arabic script so that right-to-left direction is not disrupted) which takes an argument: the vowel between the two *lāms*. Hence, to obtain الله the user can choose between one of the following:

ف/الله/

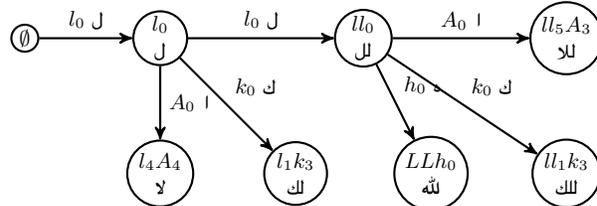
faLiLhi

The dotted circle, used to show the combining nature of short vowels and other diacritics, can be obtained by the macros `\arabdottedcircle` or دائرة/ with the macro name in Arabic script.

## 4 TeXnicities

The ض font is a *tour de force* of smart ligature use: for example, to obtain  $k_1 t_2 b_3$  (كتب) out of `ktb = k_0 t_0 b_0` (or “0643”062A”0628) input, one needs the following smart ligatures:

$k_0 \text{ LIG/ } t_0 \rightarrow k_1 t_0$   
 $k_1 \text{ /LIG } t_0 \rightarrow k_1 t_3$   
 $t_3 \text{ LIG/ } b_0 \rightarrow t_2 b_0$   
 $t_2 \text{ /LIG } b_0 \rightarrow t_2 b_3$



**Figure 3:** Finite state automaton starting with an isolated *lām* (*ʿalif* | stands for the set of letter  $\mathcal{A} = \{ \bar{a}, \bar{i}, \bar{j}, \bar{l}, \bar{r} \}$ ; ك stands for any Arabic letter besides *ʿ* and set  $\mathcal{A}$ ).

as well as the following four for Unicode input:

"0643 LIG/ "062A  $\rightarrow k_1 t_0$   
 $k_1 \text{ /LIG } "062A \rightarrow k_1 t_3$   
 $t_3 \text{ LIG/ } "0628 \rightarrow t_2 b_0$   
 $t_2 \text{ /LIG } "0628 \rightarrow t_2 b_3$

The first ligature of each group leaves *t*/"062A unchanged (isolated) and turns *k*/"0643 into initial form. Then the second ligature takes  $k_1 t_0/k_1$ "062A, leaves *k* unchanged (initial) and turns *t*/"062A into final form. But, because of the following *b*/"0628, the third ligature will turn *t* into medial form, leaving *b*/"0628 unchanged. And, finally, the fourth ligature will leave *t* unchanged and turn *b*/"0628 into final form. It is noteworthy that *t* changes form thrice: from isolated (default) it turns into final and then into medial form.

All basic Arabic glyphs are placed into the first 8-bit table. Then one 8-bit table (except for table "06xx which is used for Unicode input) is added for every letter + diacritic(s) combination, so that we have, in total, 20 tables. The complete font contains 3,514 virtual glyphs, 403,913 ligatures (321,935 of which are smart) and 7,810 kerns.

The most challenging letter is *lām*: the font contains 3 initial *lāms*, 4 medial ones as well as 3 “fake” ligatures *lām-lām* (“fake” in the sense that they are only needed because of TeX’s approach of building ligature stepwise and hence needing intermediate steps for all ligatures of length three and more: to obtain the *lām-lām-hāʾ* ligature (see §3.2.1) one needs an intermediate *lām-lām*, even though this pair of letters does not take any special form. In Fig. 3 the reader can see the finite state automaton starting with an isolated *lām*.

The virtual OVP font is built from the metrics of the PostScript Type 1 font by a Perl script. This script also reads configuration files specifying all kern pairs as well as all horizontal and vertical adjustments of diacritics. By this method, every letter has its diacritics placed at optimal positions. To compile the OVP file produced by the script into

OVF, it is mandatory to use tool *wovp2ouf* of version higher than “1.13 (build 34787)”, which will be included in T<sub>E</sub>X Live 2015.

Names of PostScript glyphs are standard,<sup>2</sup> so that copy-paste from a PDF file results in almost perfect Unicode strings.

#### 4.1 Conversion to and from UTF-8

As a tool for users, we provide two Perl scripts allowing conversion from UTF-8 to our transliteration scheme and back. These scripts can be applied selectively using, for example, the feature of many advanced text editors of applying text filters to selected text areas.

## 5 Conclusion

There was a period (in the early days of non-Latin-alphabet T<sub>E</sub>X [4, 5, 14]) where transliteration of input text was the only available method. Then, when Unicode was sufficiently widespread, T<sub>E</sub>X switched to tools allowing direct non-Latin input. In the case of Arabic, because of the particular characteristics of this script, this is — even today — not always the optimal solution, especially when we are dealing with short extracts of Arabic text combined with Latin-alphabet text and T<sub>E</sub>X commands. Maybe now is the time to return to methods based on transliteration, as an alternative to direct-input methods. We have implemented this approach, using *only* smart ligatures, as defined by Donald Knuth in 1990 [13], and the large virtual font format introduced by  $\Omega$  and taken over by LuaT<sub>E</sub>X.

We hope that this package will be useful to users seeking a straightforward method to introduce short Arabic extracts into Latin-alphabet documents.

## References

- [1] Adobe Systems. Adobe glyph list. <http://partners.adobe.com/public/developer/en/opentype/glyphlist.txt>, 2002.
- [2] Hàn Thé Thành. Micro-typographic extensions to the T<sub>E</sub>X typesetting system. *TUGboat*, 21(4):317–434, 2000. <http://pdftex.org>.
- [3] John Hammersley, John Lees-Miller, et al. The writeL<sup>A</sup>T<sub>E</sub>X online collaborative L<sup>A</sup>T<sub>E</sub>X editor. <http://www.writelatex.com>.
- [4] Yannis Haralambous. Arabic, Persian and Ottoman T<sub>E</sub>X for Mac and PC. *TUGboat*, 11:520–522, 1990.
- [5] Yannis Haralambous. Towards the revival of traditional Arabic typography through T<sub>E</sub>X. In *Proceedings of EuroT<sub>E</sub>X'92*, pages 293–305. CSTUG, 1992.
- [6] Yannis Haralambous. *Fonts & Encodings*. O'Reilly, 2007.
- [7] Yannis Haralambous, Yassir Elidrissi, and Philippe Lenca. Arabic language text classification using dependency syntax-based feature selection. Submitted to *CITALA 2014*.
- [8] Yannis Haralambous and John Plaice. First applications of  $\Omega$ : Adobe Poetica, Arabic, Greek, Khmer. *TUGboat*, 15:344–352, 1994.
- [9] Yannis Haralambous and John Plaice. Multilingual typesetting with  $\Omega$ , a case study: Arabic. In *Proceedings of the International Symposium on Multilingual Information Processing '97*, pages 137–154. ETL, Tsukuba, Japan, 1997.
- [10] Yannis Haralambous and John Plaice. The design and use of a multiple-alphabet font with  $\Omega$ . In *Electronic Publishing, Artistic Imaging, and Digital Typography*, volume 1375 of LNCS. Springer, 1998.
- [11] Taco Hoekwater. LuaT<sub>E</sub>X. *TUGboat*, 28:312–313, 2007. <http://luatex.org>.
- [12] Jonathan Kew. X<sub>Y</sub>T<sub>E</sub>X, the multilingual lion: T<sub>E</sub>X meets Unicode and smart font technologies. *TUGboat*, 26:115–124, 2005. <http://tug.org/xetex>.
- [13] Donald E. Knuth. The new versions of T<sub>E</sub>X and METAFONT. *TUGboat*, 10:325–328, 1989.
- [14] Klaus Lagally. ArabT<sub>E</sub>X — Typesetting Arabic with vowels and ligatures. In *Proceedings of EuroT<sub>E</sub>X'92*, pages 153–172. CSTUG, 1992.
- [15] Ahmed Lakhdar-Ghazal. *Pour apprendre et maîtriser la langue arabe*. Institut d'études et de recherches pour l'arabisation, Rabat, Morocco, 1991.
- [16] John Mace. *Arabic verbs and essential grammar*. Teach yourself books, 1999.
- [17] Nicole Richert. *Arabisation et technologie*. Institut d'études et de recherches pour l'arabisation, Rabat, Morocco, 1987.
- [18] Karin C. Ryding. *Arabic. A linguistic introduction*. Cambridge University Press, Cambridge, 2014.
- [19] Michael Shell. IEEEtran L<sup>A</sup>T<sub>E</sub>X class. <http://ctan.org/pkg/ieeetran>, 2007.
- [20] John R. Smart. *Arabic*. Teach yourself books, 1986.
- [21] Barakat Ahmad Syed. *Introduction to Quranic script*. Curzon Press, 1984.

◇ Yannis Haralambous  
 Institut Mines Télécom, Télécom Bretagne,  
 UMR CNRS 6285 Lab-STICC  
 Technopôle Brest Iroise CS 83818,  
 29238 Brest Cedex 3, France  
[yannis.haralambous \(at\) telecom-bretagne dot eu](mailto:yannis.haralambous(at)telecom-bretagne dot eu)

<sup>2</sup> These names are either taken from the Adobe Glyph List [1], or using the standard convention `uniXXXX.var` where `XXXX` is the Unicode position of the character and `var`  $\in$  `{ini, med, fin, iso}`. Ligatures are named by the names of their components, concatenated using underscores.

---

## Visual editing (in a specialized case): `prerex`

Bob Tennent

### Abstract

It is sometimes desirable and straightforward to support visual editing for  $\text{\LaTeX}$ ; this article describes one such case — course prerequisite charts, supported by the `(v)prerex` programs.

### 1 Introduction

One of the most frequently asked questions by  $\text{\LaTeX}$  beginners is whether a graphical interface “like a word processor” is available. Most readers of this article know how to respond: we emphasize logical structure and we point out the availability of  $\text{\LaTeX}$ -friendly text editors,  $\text{\LaTeX}$  development environments, `preview-latex`<sup>1</sup> and, in recalcitrant cases, Scientific Word,<sup>2</sup> or the `LYX`<sup>3</sup> or `TeXmacs`<sup>4</sup> “document processors”. Not as often mentioned is the difficulty of parsing *arbitrary*  $\text{\TeX}$  documents; it’s been said that only  $\text{\TeX}$  can process  $\text{\TeX}$ .

In this article, I discuss the design of a system that *allows* (but doesn’t require) a form of visual editing of a *specialized*  $\text{\LaTeX}$  environment for “prerequisite charts”.

### 2 Prerequisite charts

A prerequisite chart gives an attractive graphical presentation of courses in a program (or set of related programs), organized by terms or years, linked by pre- and co-requisite arrows (directed edges), and, when possible, supplemented by timetable information; Figure 1 on page 286 is a small example. Realistic examples may be found at <http://www.cs.queensu.ca/students/undergraduate/prerequisites>.

Some notable properties of these charts:

- Each course box is sized to just enclose the text within it, with uniform standard margins.
- Each arrow between courses is oriented from box centre to box centre, rather than from/to standard “connection points” on the box outlines.
- The arrows are “clipped” by the course boxes, but the arrowheads abut the target box exactly.

These desirable properties are not easily achieved using conventional drawing software, no matter how “user-friendly” it purports to be.

In contrast, the use of  $\text{\LaTeX}$  to produce these charts provides complete flexibility as well as professional quality.

- Text within a course box may be partitioned into regions with varying characteristics. For example, the course code and the timetable information on the first line of course boxes are in a smaller font than the course name. The latter is centered and the former are left- and right-justified, respectively. Arbitrary  $\text{\LaTeX}$  formatting can be used for the text.
- Any available fonts may be used. The  $\text{\TeX}$  typesetting engine takes advantage of kerns and ligatures in the fonts.
- Line thickness for boxes may be varied; in the example diagram, heavier boxes (and bold-face text) are used to indicate that a course is “required” in the program, rather than an option.
- Different styles of connectors can be used, for example to distinguish prerequisites, co-requisites, and recommended prerequisites.
- Various sizes or shapes of course boxes may be used, for example to distinguish between half and full courses.
- Graphic images such as logos can be imported.
- Colours and hyperlinks to on-line course descriptions or calendars are possible.

As an example, the chart in Figure 1 is produced by the  $\text{\LaTeX}$  code in Figure 2. A conventional two-dimensional Cartesian coordinate system is used to specify the locations of diagram elements. The origin (where  $x = 0$  and  $y = 0$ ) is at the lower-left corner of the diagram. The coordinates of boxes are those of its centre point; an arrow is described by the coordinates of the centre points of its source and target boxes. The order of commands is not significant except that the commands for the source and target boxes of an arrow should precede the command for the arrow.

The `prerex` package<sup>5</sup> currently uses `pgf`<sup>6</sup> and other standard packages to implement the `chart` environment and a specified set of commands within that environment. Some implementation details:

- The half-course boxes are assigned a minimum height to give a more uniform appearance to horizontal rows of such boxes.
- Arrows with a small height are always drawn straight (using a specialized and simpler macro) unless a non-zero curvature is explicitly requested.
- A wider white edge is drawn below every arrow to improve the appearance of crossing arrows.

---

<sup>1</sup> <http://preview-latex.sourceforge.net/>

<sup>2</sup> <http://www.sciword.demon.co.uk>

<sup>3</sup> <http://www.lyx.org/>

<sup>4</sup> <http://www.texmacs.org/>

<sup>5</sup> <http://www.ctan.org/pkg/prerex>

<sup>6</sup> <http://www.ctan.org/pkg/pgf>

### 3 The prerex editor

It is certainly possible, though rather tedious and error-prone, to use a conventional text editor to create and revise such descriptions, with some operations, such as global or partial “shifts” of chart elements, being particularly problematic.

The `prerex` editor is a C program that allows chart descriptions to be edited interactively. The editor supports add, remove, cut-and-paste, and edit operations on diagram elements, and vertical or horizontal shifts of: a list of specified elements, all the elements in a rectangular region, or the entire diagram. The edited diagram may be saved, re-processed, and viewed in any PDF viewer, without exiting the editor.

The program reads the source file (possibly for an initial “blank” chart), saving text until the chart environment is found, then parses the chart commands into an internal representation of the diagram. The rest of the source file is saved as text. Macro definitions and calls are not processed or expanded.

When the internal representation of the chart (linked lists of node and arrow data) is available, it is routine to implement editing operations. At any time, the user can ask for a source file to be re-created (using the saved texts and the possibly revised internal representation) and then re-processed. The revised output is available in about four seconds and can be observed in any PDF viewer.

Also observable in most PDF viewers are the coordinates of nodes, or the coordinates of the initial and terminal nodes of arrows, when the cursor is moved over the node or arrow; this is because, during editing, the usual URL associated with nodes is replaced by a special URI containing these coordinates. On initialization, a coordinate grid is generated for the background of the chart in order to facilitate determination of coordinates. If necessary, the user can “escape” from the editor to a shell, for example, to edit the source file with a conventional text editor. A multi-level “undo” command is available. If a box is “cut” and then “pasted” elsewhere, the target or source coordinates of arrows into or out of the box are adjusted automatically, and similarly if nodes are shifted or raised.

The source code for the `prerex` editor is available in the documentation directory of the `prerex` package. The only non-standard library dependence is `readline` (or `libedit`).

### 4 The vprerex interface to prerex

Of course, interactive editing with revisions monitored in a PDF viewer is not what is usually meant by *visual* editing. But most of the convenience of visual diagram editing can be obtained by simply

allowing coordinates of “selected” diagram elements or background points to be conveyed via the “clipboard” from the PDF viewer to the `prerex` editor command line.

To implement this, a bare-bones open-source PDF viewer was hacked so that, when nodes, arrows or background points are mouse-clicked, the relevant chart coordinates are loaded into the clipboard. For nodes and arrows, the relevant chart coordinates are already available in the special URIs. For background points, it is necessary to transform PDF coordinates into chart coordinates. To allow this transformation, two “anchors” (i.e., virtual nodes) are inserted at the southwest and northeast corners of the chart; from their PDF coordinates and their known chart coordinates, it is possible to compute the chart coordinates of arbitrary clicked points on the chart. All the `vprerex` application does is to start up the `prerex` editor in an `xterm` terminal and the `prerex`-enabled PDF viewer. This naive approach to visual editing is relatively simple to implement and quite pleasant to use.

The source code for the `vprerex` application is available in the documentation directory of the `prerex` package. It depends on the `poppler-qt4` and other Qt-4 libraries.

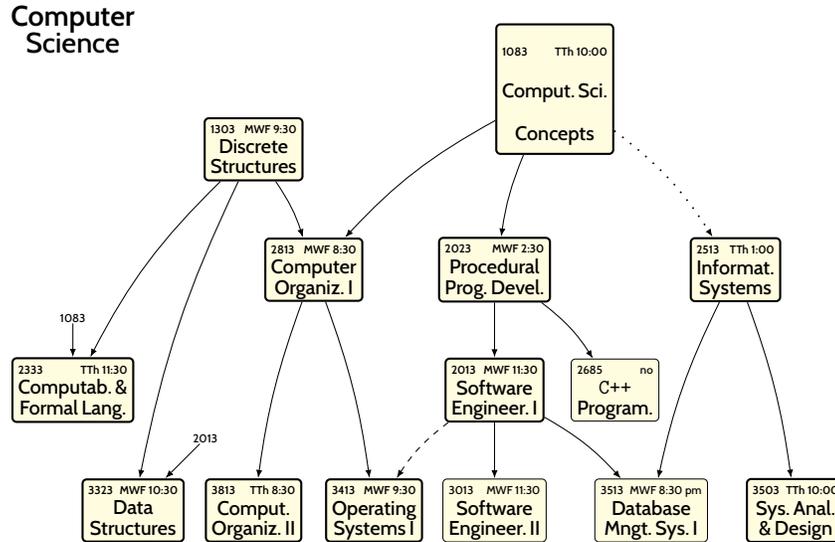
### 5 Discussion

An interactive editor like `prerex` is feasible because it only has to deal with a single very specialized environment and specialized commands within that environment. The “visual editor” `vprerex` is notable for being *unambitious*: it simply puts coordinates into the clipboard for the user to convey to the interactive editor, whereas very ambitious projects such as `VorTeX` (Visually-oriented `TeX`) [1] and `TeXLite` [2] have apparently foundered. Perhaps the approach described here might be applicable to other projects which could benefit from a form of visual editing.

### References

- [1] Pehong Chen et al. The `VorTeX` document preparation environment. In *TeX for Scientific Documentation*, volume 236 of *Lecture Notes in Computer Science*, pages 45–54. Springer, 1986.
- [2] Igor I. Stokov. A WYSIWYG `TeX` implementation. *TUGboat*, 20(4):356–359, December 1999. <http://tug.org/TUGboat/tb20-4/tb65strok.pdf>.

◇ Bob Tennent  
 School of Computing, Queen’s University  
 Kingston, Ontario K7L 3N6 Canada  
 rdtennent (at) gmail dot com  
<http://www.ctan.org/pkg/prerex>



- A solid arrow  $\longrightarrow$  indicates a required prerequisite, a dotted arrow  $\cdots \rightarrow$  indicates a corequisite (to be taken before or concurrently), and a dashed arrow  $-\cdots \rightarrow$  indicates a recommended prerequisite. Core courses are in **bold** boxes; other courses (i.e., options or prerequisites) are in **light** boxes.
- Timetabling abbreviations: M, T, W, Th, F=Mon, Tue, Wed, Thur, Fri, resp.; eve=7:00–9:50 pm; no=not offered.

Figure 1: A prerex-formatted prerequisite chart

```

\begin{chart}
\text 10,50:{\Large Computer\\\Large Science}
\reqfullcourse 50,45:{1083}{Comput.\,Sci.\Concepts}{TTh 10:00}
\reqhalfcourse 25,40:{1303}{Discrete\\Structures}{MWF 9:30}
\reqhalfcourse 30,30:{2813}{Computer\\Organiz.\,I}{MWF 8:30}
\prereq 50,45,30,30:
\prereq 25,40,30,30:
\reqhalfcourse 45,30:{2023}{Procedural\\Prog.\,Devel.}{MWF 2:30}
\prereq 50,45,45,30:
\reqhalfcourse 65,30:{2513}{Informat.\Systems}{TTh 1:00}
\coreq 50,45,65,30:
\mini 10,26:{1083}
\reqhalfcourse 10,20:{2333}{Computab.\,&\\Formal\\Lang.}{TTh 11:30}
\prereq 25,40,10,20:
\prereq 10,26,10,20:
\reqhalfcourse 45,20:{2013}{Software\\Engineer.\,I}{MWF 11:30}
\prereq 45,30,45,20:
\halfcourse 55,20:{2685}{\texttt{C++}\\Program.}{no}
\prereq 45,30,55,20:
\mini 21,16:{2013}
\reqhalfcourse 15,10:{3323}{Data\\Structures}{MWF 10:30}
\prereq 25,40,15,10:
\prereq 21,16,15,10:
\reqhalfcourse 25,10:{3813}{Comput.\Organiz.\,II}{TTh 8:30}
\prereq 30,30,25,10:
\reqhalfcourse 35,10:{3413}{Operating\\Systems\\,I}{MWF 9:30}
\prereq 30,30,35,10:
\recomm 45,20,35,10:
\halfcourse 45,10:{3013}{Software\\Engineer.\,II}{MWF 11:30}
\prereq 45,20,45,10:
\halfcourse 58,10:{3513}{Database\\Mngt.\,Sys.\,I}{MWF 8:30 pm}
\prereq 65,30,58,10:
\prereq 45,20,58,10:
\reqhalfcourse 70,10:{3503}{Sys.\,Anal.\\\&\\Design}{TTh 10:00}
\prereq 65,30,70,10:
\end{chart}

```

Figure 2: L<sup>A</sup>T<sub>E</sub>X source for the prerequisite chart in Fig. 1

---

**l3build — A modern Lua test suite  
for T<sub>E</sub>X programming**

Frank Mittelbach, Will Robertson and  
The L<sup>A</sup>T<sub>E</sub>X3 team

**Contents**

1	Introduction	287
2	History	287
	2.1 The needs in the '90s . . . . .	288
	2.2 The general approach . . . . .	288
	2.3 The new needs (in the new century)	289
3	Overview of the new system	289
	3.1 Modes of testing . . . . .	290
4	Setting up the regression test system	290
	4.1 Creating and checking test output .	290
	4.2 An example driver file . . . . .	291
	4.3 The structure of test files . . . . .	291
	4.4 Options . . . . .	292
5	Operating the system	292
6	Acknowledgements	293

**1 Introduction**

Regression tests are an important tool in any moderately complex programming environment. They allow the programmer to make extensive changes to their code while providing confidence that something that used to work still does. Extensive regression test suites have been an essential component of the maintenance and development of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and L<sup>A</sup>T<sub>E</sub>X3.

A regression test suite is typically composed of a number of individual files that contain one or more testable units of the code being tested. A testable unit might be either a certain computation with an expected outcome, a series of logic tests, or—in particular for T<sub>E</sub>X-based code—material that is typeset and intended to achieve some particular formatting.

During code development and before any new code is released to the public, this test suite can be compiled to ensure that any changes to the code have not introduced bugs or changed the behaviour compared to previous versions. As bugs in the code are reported, minimal examples demonstrating the bug often form test files of their own, showing that the bug has been fixed and won't re-occur.

As T<sub>E</sub>X-based code operates in at least three different 'modes' (mouth, stomach, and output), regression testing is more complex than simply asserting the outcome of certain programming logic. As part

of the work of the L<sup>A</sup>T<sub>E</sub>X3 project, a new Lua-based testing environment has been written to support ongoing development. This testing environment, presented at the 2014 TUG conference in Portland [3], is suitable for use by the general T<sub>E</sub>X community.

**2 History**

The ideas for a regression test suite for L<sup>A</sup>T<sub>E</sub>X date back to the early nineties<sup>1</sup> when L<sup>A</sup>T<sub>E</sub>X 2.09 existed in various incompatible flavours around the world due to its limitations in properly supporting font selection, complex mathematics, and languages other than English. Because of that situation L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> was designed and implemented to reunite the different format and to provide a stable platform for future L<sup>A</sup>T<sub>E</sub>X development.

However, to successfully introduce L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> as an accepted successor of L<sup>A</sup>T<sub>E</sub>X 2.09 it was essential to win over the huge L<sup>A</sup>T<sub>E</sub>X user base and provide them with a system that was as stable and upward compatible as possible. Thus existing user interfaces should be preserved and typesetting should provide identical output except in those cases where bug fixes or deliberate design decisions resulted in changes.

To achieve this we devised a validation mechanism that could be used to ensure that interfaces behave as expected and typesetting results do not change even though the underlying code gets modified. With this in place the L<sup>A</sup>T<sub>E</sub>X3 Project Team together with additional volunteers set out to create a large number of test files and verify them against the current L<sup>A</sup>T<sub>E</sub>X 2.09 implementation. Figure 1 shows the original request for volunteers (exhibiting a severe underestimation of the amount of work involved); see also [2] for a more extensive description of this endeavor.

This effort resulted in something like 200 test files that were then used to assure ourselves that the new L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> implementation was faithfully supporting all interfaces—it was one of the key factors that ensured the new system became an accepted replacement for L<sup>A</sup>T<sub>E</sub>X 2.09 within a reasonably short period.

Once in place this regression test suite was augmented over time and now contains roughly 350 test files altogether. Whenever a bug was found and fixed we added a new test file that would exhibit the undesired behavior if that bug would somehow resurface through later changes.

Though not perfect (after all we introduced a number of bugs that initially were not caught by the

---

<sup>1</sup> As with many ideas in the T<sub>E</sub>X world, this one too can be partly traced back to Don Knuth, who already provided his own regression test for T<sub>E</sub>X a decade earlier [1].

### Validating L<sup>A</sup>T<sub>E</sub>X 2.09

Writing test files for regression testing: checking bug fixes and improvements to verify that they don't have undesirable side effects; making sure that bug fixes really correct the problem they were intended to correct; testing interaction with various document styles, style options, and environments.

We would like three kinds of validation files:

1. General documents.
2. Exhaustive tests of special environments/modules such as tables, displayed equations, theorems, floating figures, pictures, etc.
3. Bug files containing tests of all bugs that are supposed to be fixed (as well as those that are not fixed, with comments about their status).

A procedure for processing validation files has been devised; details will be furnished to anyone interested in this task. Estimated time required: 2 to 3 weeks, could be divided up.

**Figure 1:** Original request for volunteers

regression test suite), the approach served us very well and prevented a number of horrible mistakes that would otherwise have made it into public releases of L<sup>A</sup>T<sub>E</sub>X.

### 2.1 The needs in the '90s

With the initial regression test suite we solved a number of burning problems. First of all we wanted to be confident that the code and the documented user interfaces worked as expected. Whenever we recoded an internal function the test suite would automatically alert us if that resulted in any noticeable changes at the user level or in downright bugs.

Furthermore L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> came with much more documentation and the tests included compiling and checking the documentation files for errors and missing references.

In addition the Makefiles that ran the tests also included goals to build the distribution automatically. Compared to L<sup>A</sup>T<sub>E</sub>X 2.09, which consisted of very few files, the format for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> was generated from many source `.dtx` files, so the housekeeping complexity was greatly increased.

Another issue we had to tackle was that the code was no longer maintained by a single person but by developers living in different places around the world and using different operating systems and

installations. So the regression suite had to function with different installations without creating spurious differences.

Finally all tasks had to work without user intervention or manual work because only in that case will such a system be used on a regular basis and thus benefits be realized.

### 2.2 The general approach

Designing a test system for verifying T<sub>E</sub>X's type-setting behavior is not easy—how do you test for correctness and how do you ensure that the tests are repeatable over time and in different places?

The approach we came up with was to build test files that generate suitable data in their `.log` files. Suitable data would be, for example, the state of counters or dimensions produced with `\showthe`, data written with `\typeout`, and box content shown with `\showbox`. Some of the tracing parameters of T<sub>E</sub>X could be used to verify paragraph building or page breaking decisions, but something like `\tracingall` would be inadvisable, as that would show the internal coding and not the expected functionality.

The result of running such a test file would then be manually verified and stored away as a certified result. However, as many readers will already be aware, L<sup>A</sup>T<sub>E</sub>X's `.log` files contain a lot of irrelevant data, some of which differs from run to run and some of which differs when running on different installations. So to make this approach workable we introduced a cleanup step in which we modified the result files removing irrelevant material and normalized some of the remaining parts. Of course one has to be careful not to sanitize too far, but we found a number of things necessary or at least advisable, including

- shortening file path info to avoid differences between installations
- drop empty lines (different T<sub>E</sub>X implementations put in different numbers of these)
- drop line numbers in `'on line <num>'` to avoid differences just because extra lines got introduced in a test file.

Putting it all together we ended up with a system consisting of test files (with the extension `.lvt`), certified result files to compare against (extension `.tlg`) and a fairly complex Makefile and a number of Perl scripts used to run the different tasks. These tasks included running the test suite, producing the documentation and generating the distribution (ready to be shipped to CTAN). It also contained a number of special functions such as unpacking and locally installing the code, cleaning up the source directories,

checking individual test files, and producing a new `.tlg` file for a given test file.

### 2.3 The new needs (in the new century)

As mentioned above, the initial system served us well, when moving from L<sup>A</sup>T<sub>E</sub>X 2.09 to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and then throughout the '90s, which had very active L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> development with releases produced at half-year intervals.

In this century, development of the core of the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> kernel has slowed to a minimum (releases are now only every couple of years and the changes are small) while it has intensified in other areas such as actively progressing the development of the L<sup>A</sup>T<sub>E</sub>X 3 programming language `expl3`. With this new focus, newly important requirements for a regression test system became apparent.

Instead of a single distribution we now had to deal with a growing number of distributions: core L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and its packages, Babel (with a different release cycle), `expl3` and possibly smaller and larger distributions of third party code that also wanted to benefit from a functional regression test system.

Windows and Mac OS X became the operating systems of choice for several developers and the Makefile approach of the original test suite did not work on Windows and only with modifications on Mac OS X.

Last but not least, a number of new T<sub>E</sub>X-based engines matured and people now wanted to use L<sup>A</sup>T<sub>E</sub>X and friends not only on pdfT<sub>E</sub>X but also on these new engines all of which provided additional capabilities. These new engines showed a number of subtle differences when adding data to the `.log` file, or due to extended capabilities showed additional data (such as extra nodes in listings). Furthermore the new engines still have bugs and a number of them showed up when we initially ran test files and compared their output with the certified `.tlg` data.

Thus testing became a multi-dimensional problem: one had to verify test results with several engines and it had to work on multiple operating systems. Furthermore new code sources posed new or different requirements for building a distribution or doing the testing and we soon found that the original approach made a number of hardwired decisions that were no longer applicable if the system was used with a distribution different from L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>.

For a short while we tried to accommodate the need for Windows support by using a set of `.bat` files in parallel with the Makefile approach but obviously that was doomed to failure, being impractical to maintain. Another avenue we explored was switching to a fully Perl-based approach (using Cons) but that again didn't work well with Windows and fur-

thermore it would have been a solution not available out of the box on any T<sub>E</sub>X installation.

Eventually, we decided to apply the same principle used long ago with `docstrip.tex`: use the scripting language with some operating system capabilities that is available out of the box on all T<sub>E</sub>X installations. Back then the answer was that only T<sub>E</sub>X itself fit that bill and so T<sub>E</sub>X became the tool to build style files, etc., from `.dtx` sources. However, while T<sub>E</sub>X as such is too limited to be used for scripting a regression test system, we now had LuaT<sub>E</sub>X as an engine that offers a full-fledged Lua interpreter — and these days LuaT<sub>E</sub>X is part of all modern T<sub>E</sub>X installations.

Moving to Lua (or `texlua` to be precise) means that the test and distribution system is now not tied to either the operating system (as the script runs on Windows and Unix variants) or to third-party tools (as Lua is available as part of a modern T<sub>E</sub>X system).

— \* —

In the remaining sections of this article we describe the new system and how it can be applied to support arbitrary code within the T<sub>E</sub>X world.

### 3 Overview of the new system

To illustrate, a hypothetical package will be described that uses the new system: consider a package `abc` with a collection of source files in the following layout.

```
abc/
  abc.dtx
  abc.ins
  build.lua
  README
  testfiles/
    test1.lvt
    test1.tlg
    ...
  support/
    abc-test.cls
```

What is added in addition to the normal source files is a short Lua script, normally called `build.lua`. Test files and their certified results are located in the folder `'testfiles/'` with extensions `.lvt` and `.tlg`, respectively. The files in `support/` (if any) are used when running the test files.

Upon running the test suite, a new folder `'build'` is created in which the package is unpacked, support files are copied across, and each test file is run in turn and compared to its original `.tlg` file. Directories and file names are adjustable and other setups are possible; the above structure is simply the default.

### 3.1 Modes of testing

The best way to perform regression tests for  $\TeX$  programming is to use the `.log` file; only here can box content be tested, not just logical and programmatic constructs. Box content is essential for checking from the very highest level that code changes do not result in different typeset output.

$\TeX$  programming can be either *expandable* or not. Code that is expected to be expandable should be tested as such. This can be done by evaluating it within something like `\typeout` (in the case of  $\LaTeX$ ). For non-expandable tests one should output their results to the `.log` once they have been evaluated. As mentioned earlier there are also a number of  $\TeX$  tracing parameters and commands like `\showbox`, `\showlists`, or `\showthe` that can be used to generate relevant test data in the `.log` file.

To aid in producing a structured test suite we provide a number of commands for use in the test files. The `\TYPE` command is used to write material to the `.log` file; it works like `\typeout`, but it allows ‘long’ input. A variety of commands, following, then use `\TYPE` to output strings to the `.log` file.

- `\SEPARATOR` inserts a long line of = symbols to break up the output.
- `\TRUE`, `\FALSE`, `\YES`, `\NO` insert text strings for standardized comparison.
- `\ERROR` is not defined but is commonly used to indicate a code path that should never be reached.

To produce individual tests we offer the commands `\TEST` and `\TESTEXP`. These commands take two arguments: a title and the actual test body. `\TESTEXP` executes the body within a `\TYPE` command to test expandability but with `\TEST` you are responsible for generating test output using `\TYPE`, `\TRUE`, etc. as it is intended to be used for non-expandable tests. Both commands surround the generated output with `\SEPARATORS` and display the title and a test number. Here is an example:

```
\begin{TEST}{stepping counters}
{
  \setcounter{chapter}{2}
  \setcounter{section}{5}
  \setcounter{subsection}{4}
  \stepcounter{chapter}%
  \TYPE{\arabic{chapter}-%
    \arabic{section}-\arabic{subsection}}
  \SEPARATOR
  \refstepcounter{section}
  \TYPE{\arabic{chapter}-%
    \arabic{section}-\arabic{subsection}}
}
```

This test will then produce the following output, as in standard  $\LaTeX$  only a counter directly “within” is reset to zero (e.g., the `subsection` counter is not touched when `chapter` is stepped):

```
=====
TEST 8: stepping counters
=====
3-0-4
=====
3-1-0
=====
```

(Assuming it’s the eighth test in the file.)

## 4 Setting up the regression test system

Consider the case that a  $\LaTeX$  package consists of one or more `.dtx` files in a flat directory structure. By default, to set up a regression test suite, you would create a driver file named ‘`build.lua`’ and sub-folder named ‘`testfiles/`’ to contain the test files. An example driver file is shown in Section 4.2.

The test files can be called basically anything (but should be logical in some way), and by default have the extension `.lvt`. These are accompanied by a pre-saved `.tlg` file which contains the ‘results’ of the test file to be checked against subsequent compilation of that test. If a test file has different results for different engines it is possible to “certify” `.tlg` files for each engine; those then have extensions such as `.luatex.tlg`.

### 4.1 Creating and checking test output

The first time a `.lvt` test file is written, it will need to be compiled to obtain the necessary `.tlg` output for future tests. This is performed with:

```
texlua build.lua save <test name>
```

(To produce an engine-specific `.tlg` file an additional `<engine>` argument can be given.) This task can be re-run as many times as necessary until the test file demonstrates the necessary behaviour being tested. At this point,

```
texlua build.lua check <test name>
```

will then re-run the `.lvt` file and compare the result to the original `.tlg` output. If no `<test name>` is specified all tests in the test directory are run. Presuming no code has changed to affect the output of the tests, the console output of this task will show the name of the test files being processed followed by the line:

```
All checks passed
```

If only one test file is run the usual console output from the  $\TeX$  compilation is also shown otherwise it is suppressed.

```
#!/usr/bin/env texlua

-- Build script for abc package

module = "abc"

-- variable overwrites (if needed)

-- call standard script

kpse.set_program_name ("kpsewhich")
dofile (kpse.lookup ("l3build.lua"))
```

**Figure 2:** Driver file for a hypothetical `abc` package

```
\documentclass{breqn-test}
\input{regression-test}
\usepackage{breqn}
\begin{document}
\START
\AUTHOR{Will Robertson}
\begin{dmath}
a+b+c+d+e+f+g+h+i+j+k+l+m+
n+o+p+q+r+s+t+u+v+w+x+y+z
\end{dmath}
\showoutput
\end{document}
```

**Figure 3:** Example test from `breqn`

These compilations take place in the subdirectory ‘`build/test`’, and if a test fails, a diff file is deposited there with the information about what has changed in the output of the test file. Also deposited there are the full `.log` files for each *engine* (i.e., without modifications from the cleanup step) which can be helpful to debug complex issues.

## 4.2 An example driver file

For a simple setup such as shown in the overview in Section 3, the driver file (`build.lua`) is quite simple. An example of such a driver file is shown in Figure 2; it need do little more than inform the build system of the name of the package and perhaps set some flags or change some defaults if they are not adequate.

The main script is `l3build.lua`, which is automatically found in the `texmf` tree (via `kpsewhich`) and then loaded. Thus, there is no need to hardwire locations in the driver file and it will work on different installations.

## 4.3 The structure of test files

As mentioned previously, the method of using the `.log` file allows various types of tests to be conducted. The most simple test might load a package and exe-

cute some commands to produce a small amount of typeset output. A complete example of such a test is shown in Figure 3. Some points to note:

1. The first line, `\input{regression-test}` loads the necessary settings and commands to format the `.log` file properly for testing.
2. It is not necessary to load a special document class (most tests use `article` or `minimal`), but a package author may wish to adjust page margins, etc., without repeating the commands for each test. Such a special test class or package could then be kept in the `support/` directory.
3. The test begins proper at `\START`—everything before that point in the `.log` file will be ignored. This prevents, for example, package version numbers displayed while the preamble is processed from becoming part of the test. The `\AUTHOR` declaration is an optional way of indicating who might know how to fix the problem should the test begin failing.
4. In this example `\showoutput` generates the actual test data by generating a symbolic representation of the page content in the `.log` file.
5. A slightly modified version of `\end{document}` finishes the test document. Alternatively, one can end the test file with `\END` which avoids the final processing done by `\end{document}` and thus prevents unwanted material from becoming part of the test data. In this example `\END` cannot be used as that would stop the run immediately without producing a page—which is our goal here.

Not shown is the `\OMIT . . . \TIMO` construction, which puts flags into the `.log` file between which no test comparisons will be made. This can be used around code that generates variable log information that is known to be irrelevant for the test. For example, statements like `\newlength` or `\newcounter` write some tracing information into the `.log` that shows the allocated register number. If the code gets revised these numbers might change and thereby unnecessarily invalidate the test result.

`\OMIT` can also be used before `\end{document}` if you need the final processing to happen, but want to ensure that nothing written at that time becomes part of the test.

An example of a more structured test from the  $\LaTeX$ 3 test suite is shown in Figure 4. Here, a number of different tests are contained within a single file, and a few of these are included in the example. The content of the test is not really important here (it is testing aspects of the integer module from `expl3`) but it does show a few best practices.

`\OMIT/\TIMO` is used to hide the register allocation numbers from `\int_new:N`. The first test then exercises integer addition and subtraction which is not expandable (therefore `\TEST` together with `\TYPE` is used) and it consists in fact of several small tests. The expected results are written as comments into the test file which is helpful in case it ever fails.

Converting integers is supposed to be expandable so `\TESTEXP` is used for the second test. The same is true for the case selection commands. Here the test output is generated by `\YES` or `\NO`.

Can you guess the test results, even if you are not familiar with the `expl3` language? They are shown in Figure 5.

#### 4.4 Options

While the examples shown previously demonstrate the behaviour in the standard setup, the new build system provides significantly greater flexibility. This is achieved by providing a large number of variables that can be (re)set as necessary in the driver file. For example, the new system supports building complex distributions consisting of several modules in different directories with dependencies between them. You can also control if the processing should happen in a sandbox or if it is allowed to draw any support files needed for the tests (e.g., extra packages or classes) from the TDS tree. The latter is the default as this is better for most distributions. For details consult the documentation in [4].

There is one option that one may have to modify even for simple setups: `checkruns`. This controls the number of times each test file is run; to speed up processing it defaults to 1. If, however, the codes require multiple runs to function (e.g., if you test material that is passed through the `.aux` file) you have to set this variable to 2 or higher to ensure that your tests actually work correctly.

#### 5 Operating the system

As indicated earlier the system does a bit more than managing a set of test files, so here is a short description of the main tasks that can be executed once the setup is in place. Each task takes zero or more arguments as described below and is executed by running the driver file (default `build.lua`) through a Lua interpreter (`texlua`) and passing it the task name and any further argument as necessary, e.g.,

```
texlua build.lua check <test name>
```

would run the check on `<test name>` using all engines. So here is the list of available tasks:

`check <name> <engine>` Without arguments, runs all test files found in the directory that contains

```
\documentclass{minimal}
\input{regression-test}
\RequirePackage{expl3}
\begin{document}
\START
\AUTHOR{Frank Mittelbach, LaTeX3 Project}
\ExplSyntaxOn

\OMIT
\int_new:N \l_testa_int
\int_new:N \g_testa_int
\TIMO

\TEST { adding~and~subtracting }
{
  \int_zero:N \l_testa_int
  \int_add:Nn \l_testa_int { 5 * 7 }
  \int_add:Nn \l_testa_int { 15 }
  % we hope for a value of 50
  \TYPE { \int_use:N \l_testa_int }
  \int_sub:Nn \l_testa_int { 3 * 5 }
  % we hope for a value of 35
  \TYPE { \int_use:N \l_testa_int }
  \int_gzero:N \g_testa_int
  {
    \int_gadd:Nn \g_testa_int
      { (2 + 13) / (2 * 3) }
    \int_gadd:Nn \g_testa_int { 3 }
  }
  % we hope for a value of 6
  \TYPE { \int_use:N \g_testa_int }
  \int_gsub:Nn \g_testa_int { 5 * 5 }
}
% we hope for a value of -19
\TYPE { \int_use:N \g_testa_int }
}

\TESTEXP { converting~from~and~to~base }
{
  \int_to_base:nn { 17 } { 8 } ~
  \int_from_base:nn { 21 } { 8 }
}
\TESTEXP{ Case~statements }
{
  \int_case:nnn
    { -1 + 1 }
    { { -1 } { \NO }
      { 3 - 3 } { \YES } }
    { \NO }
  \NEWLINE
  \int_case:nnn
    { 7 - 2 }
    { { -1 + 3 } { \NO } }
    { \YES }
}
% more tests here omitted
\END
```

Figure 4: Expandable and non-expandable tests

```

=====
TEST 1: adding and subtracting
=====
50
35
6
-19
=====

=====
TEST 2: converting from and to base
=====
21 17
=====

=====
TEST 3: Case statements
=====
YES
YES
=====

```

Figure 5: Test results

the `.lvt` files. It reports progress by displaying each test file name currently processed (but otherwise hides any  $\TeX$  output to avoid cluttering the screen) and at the end displays a summary indicating success or failure.

If  $\langle name \rangle$  is specified, it will run only the tests for that `.lvt` file, and if additionally given an  $\langle engine \rangle$  name will run only the test for that specific engine. In either case it will show everything on the screen, which is helpful if the run shows abnormal behaviour (especially if it ends up in an endless loop and never returns for some reason).

**clean** Cleans up the source tree, removing temporary files and directories.

**ctan** Runs all tests, typesets all documentation and if there are no errors, generates a `.zip` file suitable for uploading to CTAN.

**doc** Typesets all documentation (by default `.dtx` files), thus checking them for trivial processing errors.

**install** This installs the distribution in the local tree of the user.

**save**  $\langle name \rangle$   $\langle engine \rangle$  This generates (or regenerates) the `.tlg` file for  $\langle name \rangle$ . If additionally supplied an  $\langle engine \rangle$  argument it generates a specific `.tlg` as discussed above.

It is the responsibility of the developer to verify that the data placed into the `.tlg` produces the desired result, i.e., is actually correct. Once produced or updated with **save**, the output is

considered certified and will be used to verify future check runs!

## 6 Acknowledgements

The original test suite system was a joint effort by the whole  $\LaTeX$  project team at that time, i.e., Frank Mittelbach, Rainer Schöpf, David Carlisle, Michael Downes, Alan Jeffrey, and Chris Rowley. We also had significant help when writing the initial set of test files from a number of volunteers, in particular Daniel Flipo and Chris Martin.

Around 2008 Rainer replaced the Makefile approach used for  $\LaTeX 2_{\epsilon}$  by Cons (a Perl-based solution) as the Makefile got so complex over time that it was difficult to manage.

For the  $\LaTeX 3$  development we stayed with Make as the requirements of the `expl3` distribution were initially much simpler.

Joseph Wright wrote a first set of `.bat` files for `expl3`, as by then many developers worked on Windows. Modelled after this, Frank replaced the Cons solution for  $\LaTeX 2_{\epsilon}$  in 2013.

Finally in 2014 Joseph then implemented most of the new Lua-based system and it is now successfully used to manage the  $\LaTeX 3$  (`expl3`) distribution as well as several smaller package distributions. The  $\LaTeX 2_{\epsilon}$  distribution will follow shortly.

## References

- [1] Donald E. Knuth. A torture test for  $\TeX$ . Report STAN-CS-84-1027, 1984.
- [2] Frank Mittelbach. A regression test suite for  $\LaTeX 2_{\epsilon}$ . *TUGboat*, 18(4):309–311, December 1997. <http://tug.org/TUGboat/tb18-4/tb57mitt.pdf>
- [3] Frank Mittelbach. A modern regression test suite for  $\TeX$  programming, July 2014. Talk given at TUG conference in Portland. Video and slide material available at <http://www.latex-project.org/papers>.
- [4]  $\LaTeX 3$  Project. The `l3build` package: Checking and building packages, September 2014. <http://ctan.org/pkg/l3build>

- ◇ Frank Mittelbach  
Mainz, Germany
- ◇ Will Robertson  
School of Mechanical Engineering,  
The University of Adelaide,  
Australia
- ◇ The  $\LaTeX 3$  team  
<http://www.latex-project.org>

## MetaPost path resolution isolated

Taco Hoekwater

### Abstract

A new interface in MPlib version 1.800 allows one to resolve path choices programmatically, without the need to go through the MetaPost input language.

### 1 MetaPost path solving ...

As readers may agree, MetaPost is pretty good at finding pleasing control points for paths. What may be less commonly known is that besides drawing on a picture, MetaPost can also display the found control points in the log file.

An initial illustration at this point is useful. Here is the MetaPost path input source of a very simple path (as well as a visualisation of the path):

```
tracingchoices := 1;
path p;
p := (0,0) ..(10,10) ..(10,-5) ..cycle;
```



And here is what MetaPost outputs in the log file (with some editorial line breaks):

```
Path at line 5, before choices:
(0,0)
..(10,10)
..(10,-5)
..cycle
```

```
Path at line 5, after choices:
(0,0)
..controls (-1.8685,6.35925) and (4.02429,12.14362)
..(10,10)
..controls (16.85191,7.54208) and (16.9642,-2.22969)
..(10,-5)
..controls (5.87875,-6.6394) and (1.26079,-4.29094)
..cycle
```

A more complex path of course creates more output, as in:

```
p := (0,0)..(2,20){curl1}..{curl1}(10, 5)
..controls (2,2) and (9,4.5)
..(3,10)..tension 3 and atleast 4..(1,14){2,0}
..{0,1}(5,-4);
```



Editor's note: Originally published in *ConTExT Group Proceedings, 6th meeting*, pp.13–18. Reprinted with permission.

Taco Hoekwater

which produces:

```
Path at line 7, before choices:
(0,0){curl 1}
..{curl 1}(2,20){curl 1}
..{curl 1}(10,5)..controls (2,2) and (9,4.5)
..(3,10)..tension 3 and atleast4.5
..{4096,0}(1,14){4096,0}
..{0,4096}(5,-4)
```

```
Path at line 7, after choices:
(0,0)
..controls (0.66667,6.66667) and (1.33333,13.33333)
..(2,20)
..controls (4.66667,15) and (7.33333,10)
..(10,5)
..controls (2,2) and (9,4.5)
..(3,10)
..controls (2.34547,10.59998) and (0.48712,14)
..(1,14)
..controls (13.40117,14) and (5,-35.58354)
..(5,-4)
```

### 2 ... outside MetaPost?

But what if you want to use that functionality outside of MetaPost, for instance in a C program? Before MPlib 1.8000, you would have to ...

- compile MPlib into your program;
- create a MetaPost language input string;
- execute it;
- and parse the log result.

All of that is not very appealing. It would be much better if you could ...

- compile MPlib into your program;
- create a path programmatically;
- run the MetaPost path solver directly,
- automatically updating the original path.

And that is what the current version of MPlib allows you to do.

### 3 How it works

Once again, it is easiest to show how it works by using a source code example:

```
#include "mplib.h"
int main (int argc, char ** argv) {
    MP mp;
    MP_options * opt = mp_options ();
    opt -> command_line = NULL;
    opt -> noninteractive = 1;
    mp = mp_initialize (opt);
    my_try_path (mp); /* the crux */
    mp_finish (mp);
    free (opt);
    return 0;
}
```

Most of the example code above is just what one would need, to do anything with MPlib programmatically. The new line for our purpose here calls `my_try_path(mp)`:

```

void my_try_path(MP mp) {
  mp_knot first, p, q;
  first = p = mp_append_knot (mp, NULL, 0, 0);
  q = mp_append_knot (mp, p, 10, 10);
  p = mp_append_knot (mp, q, 10, -5);
  mp_close_path_cycle (mp, p, first);
  if (mp_solve_path (mp, first)) {
    mp_dump_solved_path (mp, first);
  }
  mp_free_path (mp, first);
}

```

This function uses a new type, `mp_knot`, as well as several new library functions in MPlib available as of version 1.800.

- `mp_append_knot` creates a new knot, appends it to the path that is being built, and returns it as the new tail of the path.
- `mp_close_path_cycle` is like `cycle` in the MetaPost language.
- `mp_solve_path()` finds the control points of the path. `solve_path` does not alter the state of the given MPlib instance in any way, it only modifies its argument path.
- `mp_dump_solved_path()` *user defined function, see below for its definition.*
- `mp_free_path()` releases the used memory.

Our user-defined `mp_dump_solved_path` routine uses even more new functions. First let us look at its definition:

```

#define SHOW(a,b) mp_number_as_double \
                  (mp,mp_knot_##b(mp,a))
void mp_dump_solved_path (MP mp, mp_knot h) {
  mp_knot p, q;
  p = h;
  do {
    q = mp_knot_next(mp, p);
    printf("(%g,%g)\n "
           "..controls (%g,%g) and (%g,%g)",
           SHOW(p,x_coord), SHOW(p,y_coord),
           SHOW(p,right_x), SHOW(p,right_y),
           SHOW(q,left_x), SHOW(q,left_y));
    p = q;
    if (p != h
        || mp_knot_left_type(mp, h) != mp_endpoint)
      printf ("\n ..");
  } while (p != h);
  if (mp_knot_left_type(mp, h) != mp_endpoint)
    printf ("cycle");
  printf ("\n");
}

```

Somewhat hidden in the source above is the existence of another new type, `mp_number`, which is the data structure representing a numerical value inside MPlib.

The MPlib library functions used in our routine `mp_dump_solved_path` are as follows:

- `mp_knot_next()` moves to the next knot in the path.

- `mp_knot_x_coord()`, `mp_knot_y_coord()`, `mp_knot_right_x()`, `mp_knot_right_y()`, `mp_knot_left_x()`, `mp_knot_left_y()` all return the value of a knot field, as an `mp_number` object (the calls to these functions are hidden inside the definition of the `SHOW` macro).
- `mp_knot_left_type()` returns the type of a knot, normally either `mp_endpoint` or `mp_open`.
- `mp_number_as_double()` converts an `mp_number` to `double`.

To satisfy our curiosity, here is the actual output of the example program listed above:

```

(0,0)
..controls (-1.8685,6.35925) and (4.02429,12.1436)
..(10,10)
..controls (16.8519,7.54208) and (16.9642,-2.22969)
..(10,-5)
..controls (5.87875,-6.6394) and (1.26079,-4.29094)
..cycle

```

which is almost exactly the same as in the log file (except we've altered the line breaks for this article):

```

(0,0)
..controls (-1.8685,6.35925) and (4.02429,12.14362)
..(10,10)
..controls (16.85191,7.54208) and (16.9642,-2.22969)
..(10,-5)
..controls (5.87875,-6.6394) and (1.26079,-4.29094)
..cycle

```

The numerical output is not exactly the same because MetaPost itself does not use `mp_number_as_double` and `printf`'s `%g` for printing the scaled values that are (by default) used to represent numerical values.

This difference is not really relevant, since any programmatic use of the path solver should not have to be 100% compatible with the MetaPost programming language.

#### 4 More complex paths

Of course there are also new functions to create the more complex paths that make use of `curl`, `tension` and/or `direction` specifiers.

Here is how to encode the second MetaPost path in the earlier example:

```

first = p = mp_append_knot (mp, NULL, 0, 0);

q = mp_append_knot (mp, p, 2, 20);
p = mp_append_knot (mp, q, 10, 5);
if (!mp_set_knotpair_curls (mp, q, p, 1.0, 1.0))
  exit (EXIT_FAILURE);

q = mp_append_knot(mp, p, 3, 10);
if (!mp_set_knotpair_controls (mp, p, q,
                              2.0, 2.0, 9.0, 4.5))
  exit (EXIT_FAILURE);

```

```

p = mp_append_knot (mp, q, 1, 14);
if (!mp_set_knotpair_tensions (mp, q, p, 3.0, -4.0))
  exit (EXIT_FAILURE);

q = mp_append_knot (mp, p, 5, -4);
if (!mp_set_knotpair_directions (mp, p, q,
                                2.0, 0.0, 0.0, 1.0))
  exit (EXIT_FAILURE);

mp_close_path (mp, q, first);

```

Elaborate documentation for these extra functions (and a few more) is in the file `mplibapi.tex`, included in the MetaPost distribution.

## 5 Lua interface

There is also a Lua interface for use in LuaTeX, which is a bit higher-level:

```

<boolean> success
  = mp.solve_path(<table> knots,
                 <boolean> cyclic)

```

This modifies the `knots` table, which should contain an array of points in a path, with the substructure explained below, by filling in the control points. The boolean `cyclic` is used to determine whether the path should be the equivalent of `--cycle`. If the return value is `false`, there is an extra return argument containing the error string.

On entry, the individual knot tables can contain the six knot field values mentioned above (but typically the `left_x,y` and `right_x,y` will be missing). `x,y_coord` are both required. Also, some extra

values are allowed, all numbers:

```

left_tension   A tension specifier
right_tension  Like left_tension
left_curl      A curl specifier
right_curl     Like left_curl
direction_x    x displacement of a
               direction specifier
direction_y    Like direction_x

```

## 6 Issues to watch out for

All the ‘normal’ requirements for MetaPost paths still apply using this new interface. In particular:

- A knot has either a direction specifier, or a curl specifier, or a tension specification, or explicit control points, with the additional note that tensions, curls and control points are split in a left and a right side (directions apply to both sides equally).
- The absolute value of a tension specifier should be more than 0.75 and less than 4096.0, with negative values indicating ‘atleast’.
- The absolute value of a direction or curl should be less than 4096.0.
- If a tension, curl, or direction is specified, any existing control points will be replaced by the newly computed value.

◇ Taco Hoekwater  
 Docwolves B.V.  
<http://metapost.org>  
<http://luatex.org>

---

## Typeset MMIX programs with T<sub>E</sub>X

Udo Wermuth

### Abstract

A T<sub>E</sub>X macro package is presented as a literate program. It can be included in programs written in the languages MMIX or MMIXAL without affecting the assembler. Such an instrumented file can be processed by T<sub>E</sub>X to get nicely formatted output. Only a new first line and a new last line must be entered. And for each end-of-line comment a flag is set to indicate that the comment is written in T<sub>E</sub>X.

### How to read the following program

The text that starts in the next chapter is a *literate program* [2, 1] written in a style similar to *noweb* [7]. Readers who are not familiar with literate programming might find the following remarks useful.

The program is divided into *sections*. Each section has a number that is written in bold at the beginning of the section. A section contains two parts and at least one of them must be present: (1) a *documentation part* with one or more paragraphs, and (2) a *code part* starting with a *headline* that is followed either by  $\equiv$  or  $+\equiv$  and the *replacement code*. The headline has the format “ $\langle$  Name Number  $\rangle$ ”.

*Example:* Sections 9 and 10 of the program have the respective headlines “ $\langle$ List symbols that are special in T<sub>E</sub>X 9 $\rangle \equiv$ ” and “ $\langle$ List symbols that are special in T<sub>E</sub>X 9 $\rangle +\equiv$ ”. Section 9 has five lines of replacement code and section 10 three.

The *Name* is the name under which the replacement code can be called by other sections. The *Number* is either the number of the section (the case with  $\equiv$ ) or a smaller number when a previously defined replacement code is extended with the new code lines (case  $+\equiv$ ). In the second case the code part of the previous section that owns the smaller section number is followed by a line “See also sections ...” and the current section number is somewhere listed in the “...”. Also, in the first case the code part is also often followed by a line “This code is used in sections ...”. Then the headline of this section is used inside the replacement code of other sections (listed in the “...”), in which the final output of the complete replacement code with all extensions must be inserted. The number in the headline states the first section that contains code for it. So a reader sees in a call of a section where it starts and under this section he finds the other sections that add replacement code.

*Example:* In section 9 the lines “See also section 10.” and “This code is used in section 24.” are given. No such line appears in section 10 as it only extends the replacement code of section 9. (Note that section 10 has in its headline the number 9.) In section 24 the reference to section 9 stands for all of the eight code lines stated in sections 9 and 10.

If a section is not used in any other section then it is a *root* and during the extraction of the code a file is created that has the name of the root. This file collects all the code in the sequence of the referenced sections from the code part. The collection process for all root sections is called *tangle*. A second process is called *weave*. It outputs the documentation and the code parts as a T<sub>E</sub>X document.

*Example:* The following program has only one root that is defined in section 4 with the headline “ $\langle$ mmix.tex 4 $\rangle \equiv$ ”. The file that is created by the tangle process is therefore called “mmix.tex”.

The tangled output in the original WEB system for literate programming is intended to be read only by computers (see [2], p. 116). In the present system output is created that is readable by humans. But changes to the program should only be made in the original source of the literate program.

The following text is the output that T<sub>E</sub>X has produced from the woven document. (A few edits have been made to follow the style of this journal.)

### Contents

Introduction . . . . .	§ 1
Format of the output . . . . .	§ 6
Preparation . . . . .	§ 8
Line numbers . . . . .	§ 15
Times column . . . . .	§ 22
Setting the output format . . . . .	§ 25
Input format . . . . .	§ 29
Activation . . . . .	§ 33
Last line . . . . .	§ 40
Shortcuts . . . . .	§ 44
More shortcuts . . . . .	§ 50
Final remarks . . . . .	§ 52
Index and List of sections . . . . .	§ 55

### Introduction

1. Algorithms in *The Art of Computer Programming* (TAOCP) [3] by Donald E. Knuth are stated in plain English. But every time an implementation is needed a machine language or assembler language is used. In the first three volumes the language is MIX; in Volume 4a the algorithms are implemented in the language of a new computer called MMIX [4]. For the next editions of the TAOCP volumes all the

MIX programs of the first three volumes must be rewritten either in MMIX or in the new assembler language MMIXAL. On his web page <http://www-cs-staff.stanford.edu/~uno/mmix.html>, Knuth asks volunteers to start the conversion of the MIX programs before he has finished Volume 5 and new editions of Vols. 1–3 are created.

**2.** I decided to be one of the volunteers to do the conversion (although I'm not an MMIXmaster; see [6]). I asked myself the question: How to present the result? The MMIX programs are stored in `mms` files, which allow a very flexible input format. For example, a line that starts with a backslash will be treated as a comment and is ignored during the assembly.

So my idea is to write the `mms` files in a way that they can be processed not only by the assembler but also by  $\text{\TeX}$ . The output of  $\text{\TeX}$  shall reflect the style that is used in the TAOCP volumes to present the MIX and MMIX programs. And  $\text{\TeX}$  can be used to implement a second idea: Not only shall the conversion be done but an analysis of the new implementation shall be added.

A macro package for  $\text{\TeX}$  is developed that is included in the `mms` files. Then  $\text{\TeX}$  is able to process and pretty-print such `mms` files.

**3.** Other volunteers for the conversion from MIX to MMIX have obviously felt the same need for  $\text{\TeX}$  output. The solution on the MMIX *home page* [6] is a `lex` script `mmixtotex.1` to create a program that reads the `mms` file and outputs a  $\text{\TeX}$  file that can be typeset with a macro package `mmstotex.sty`.

**4.** Here is the plan for the macro package.

```
<mmix.tex 4> ≡
<Initialization 5>
<Definitions 24>
<Useful commands and shortcuts 44>
<Add an analysis of the algorithm 40>
<Format the mms file 19>
<Take off 33>
```

This is a root.

**5.** The file `mmix.tex` might be shipped with an `mms` file without this description. Therefore I will be adding plenty of comments to the  $\text{\TeX}$  code to help others to read and understand the macro package.

```
<Initialization 5> ≡
% Package to format MMIX programs with TeX
% (and some useful commands to document them)
% Author: Udo Wermuth
% %%%
<Description 14>
% %%%
```

See also sections 11, 12, and 13.

This code is used in section 4.

Udo Wermuth

## Format of the output

**6.** Most of the time programs are not written in MMIX but in the MMIX *Assembly Language*, called MMIXAL. MMIXAL allows labels, alphabetic names, etc. and introduces new operations that are called pseudo-ops. As we are only interested in formatting the source lines of a program the details of the extensions provided by MMIXAL are not discussed here. The reference [4] defines not only MMIX but gives all the information about MMIXAL too. A source line of a MMIXAL program has up to five elements. Three elements are of principal interest for the program behavior: (1) an optional label, (2) the operation (or short: the *op-code*), and (3) an expression field. The other two elements are not needed for the execution of the program but for the analysis and the comprehension: (4) optional timing information, i.e., the number of times the statement is executed in a run, and (5) an optional comment.

For the presentation of the program one more element is printed: an optional line number. The line number is printed in italics, the elements 1–3 are output verbatim in a monospaced font, element 4 is written in math mode, and element 5 is formatted by  $\text{\TeX}$  as normal text. Therefore the output shall look like this:

```
line label      op-code expression time comment
07 Maximum SL   kk,$0,3  1  MI.Initialize
```

**7.** Following this example, let us state the complete requirements for the output format:

- R1 The line number is either empty or has two or three digits; leading zeros are printed. It is written left-aligned in 9 pt italics.
- R2 The label is optional. If it is present it is written verbatim in a 10 pt monospaced font.
- R3 The op-code is written verbatim in the monospaced font.
- R4 The expression field may contain one or more items but it does not contain a blank (except in a string). Like the label and the op-code it is printed verbatim in the monospaced font.
- R5 The timing information is optional. If present, it is printed in 9 pt as a math expression centered in its column.
- R6 The optional comment is written in a 9 pt roman font. It is written in  $\text{\TeX}$ .
- R7 Lines that contain only a comment written in the monospaced font are allowed. Lines with more than one source statement are allowed.
- R8 The program source ends with a thick vertical bar in the comment area.
- R9 It is possible to add a runtime analysis. The text uses a 10 pt roman font.

R10 The output shall show the name and the source of the MIX program, the name of the author, who programmed the MMIX source, and the date of the conversion.

### Preparation

8. The monospaced font is a 10 pt font (see R2), the other fonts have size 9 pt (R1, R5, and R6). The 9 pt fonts (and the 6 pt fonts for subscripts) are not activated in plain T<sub>E</sub>X. Let's give them names.

```
<Fonts 8> ≡
% name 9pt and 6pt fonts
\font\ninerm=cmr9 \font\sixrm=cmr6
\font\ninesy=cmsy9 \font\sixsy=cmsy6
\font\ninei=cmmi9 \font\sixi=cmmi6
\font\nineit=cmti9 \font\ninesl=cmsl9
\font\ninett=cmtt9
\font\ninebf=cmbx9 \font\sixbf=cmbx6
```

This code is used in section 24.

9. The example in section 6 shows that MMIXAL uses characters that have a special meaning in T<sub>E</sub>X, for example, the dollar sign and hash mark are important symbols in MMIXAL. Therefore the output must be filtered and the functions assigned to the special characters in T<sub>E</sub>X have to be deactivated.

Plain T<sub>E</sub>X provides a `\dospecials` command, but let us separate the MMIXAL and T<sub>E</sub>X special characters.

```
<List symbols that are special in TEX 9> ≡
% special in TeX but common in MMIX
\def\mmixdospecials{\do\ \do\$\do\&\do\#\%
\do\^\do\_ \do\% \do\~}
% remaining special characters in TeX
\def\texdospecials{\do\\ \do\{ \do\}}
```

See also section 10.

This code is used in section 24.

10. To switch off the special meaning of the above listed characters a command from *The T<sub>E</sub>Xbook* [5], p. 380, is used.

```
<List symbols that are special in TEX 9> +=≡
\def\uncatcodespecials{% redef special chars
\def\do##1{\catcode'##1=12 }%
\mmixdospecials\texdospecials}
```

11. In the header the author, the name of the program and the original source are listed. The footer states the date and provides a page number. This fulfills requirement R10.

```
<Initialization 5> +=≡
% Header and Footer
\headline={\sevenrm Author: \authorH\hfill
Program: \pgmnameH.mms (\sourceH)}%
\footline={\sevenrm Date: \dateF\hfill
\sevenbf\folio}%
```

12. The printed document shall not only show the program but also provide the possibility of including an analysis of the algorithm (R9). The analysis is placed in a second file (a plain T<sub>E</sub>X file) and it is included with an `\input` statement. The name of the file is created from the name of the program extended by the suffix `_aoa` and, of course, with file extension `.tex`.

```
<Initialization 5> +=≡
% file name for the ‘‘Analysis of Algorithm’’
\def\AoAfile{\pgmnameH_aoa.tex}
```

13. The name of the program is by default the name of the MMIXAL file—but the user has the ability to override that name. The name of the author, the source location and the date are initialized with some text, but it is expected that the user specifies them before `mmix.tex` is loaded. The external control sequences are copied and made `\undefined`.

```
<Initialization 5> +=≡
\def\checkextdata{%
\ifundef pgmname \def\pgmnameH{\jobname}%
\else\let\pgmnameH=\pgmname
\let\pgmname=\undefined
\fi
\ifundef author \def\authorH{Unknown}%
\else\let\authorH=\author
\let\author=\undefined
\fi
\ifundef source \def\sourceH{TAOCP}%
\else\let\sourceH=\source
\let\source=\undefined
\fi
\ifundef date \def\dateF{\number\year}%
\else\let\dateF=\date
\let\date=\undefined
\fi}
```

14. The values for date, author and source must be declared outside of the package. Let us document this at the beginning of the package.

```
<Description 14> ≡
% before the macro package is loaded the
% following must be \def'ed
% required: \date, \author, \source
% the program name is taken from the mms-file
% but it can be overwritten
% optional: \pgmname
```

See also sections 26, 41, and 53.

This code is used in section 5.

### Line numbers

15. The output format states all the information about line numbers, as they are not part of the input file. Of course a counter for the numbers is needed.

```

⟨Counters 15⟩ ≡
% count registers
\newcount\lncnt % counter for line numbers

```

See also section 25.

This code is used in section 24.

**16.** Next the width of the column for the line numbers has to be defined. Two cases are stated in the requirements: 2 and 3 digits (see R1).

```

⟨Dimensions 16⟩ ≡
% dimen registers
\newdimen\lnotwodigitwidth % 2 digits col
\newdimen\lnothreedigitwidth % or 3 digits

```

See also section 22.

This code is used in section 24.

**17.** Here are the default widths of the columns.

```

⟨Set values of dimen-registers 17⟩ ≡
{\setbox0=\hbox{\nineit 00\tentt\quad}%
\global\lnotwodigitwidth=\wd0
\global\lnothreedigitwidth=1.25\wd0 }%

```

See also section 23.

This code is used in section 35.

**18.** Lines can only be numbered if the space is reserved for the column of numbers: `\colforlnotrue` must be set. Then a number is printed if the flag `\ifnumberlines` is true. A third flag is needed to set the number of digits for line numbers.

```

⟨Flags 18⟩ ≡
% if flags
\newif\ifcolforlno % true: add col for lno
\newif\ifnumberlines % true: number the lines
\newif\ifthreedigitlno % true: use 001..999

```

See also section 43.

This code is used in section 24.

**19.** The output routine for line numbers prints leading zeros (R1).

```

⟨Format the mms file 19⟩ ≡
\def\printlinenumber{% with leading 0s
\ifthreedigitlno % how many digits?
\hbox to \lnothreedigitwidth{\it
\ifnum\lncnt<100 0\fi
\ifnum\lncnt<10 0\fi \number\lncnt
\hss}%
\else\hbox to \lnotwodigitwidth{\it
\ifnum\lncnt<10 0\fi \number\lncnt
\hss}%
\fi}

```

See also sections 20, 21, 27, 28, 29, 30, 31, 32, 34, 35, and 38.  
This code is used in section 4.

**20.** But before the output routine can be called it must be checked that line numbers shall be printed at all. Therefore the following macro is called to output the line number.

```

⟨Format the mms file 19⟩ +≡
\def\numbermmline{% shall no. be printed?
\ifcolforlno
\ifnumberlines
\global\advance\lncnt by 1
\printlinenumber % yes
\else\phantom{\printlinenumber}% no
\fi\fi}

```

**21.** To have the style of line numbers available if a comment or the text of the analysis of the algorithm needs to reference a line number, one more control sequence is provided. The command is used in text printed in roman type. It gets either a single line number or a range of line numbers and prints this in italics. So a simple solution is implemented which doesn't force the user to type in the italic correction.

```

⟨Format the mms file 19⟩ +≡
\def\pgmline#1{% print #1 as line number
\gdef\argpgmline{#1}% store #1 and look ahead
\futurelet\next\pgmlinex}
\def\pgmlinex{% check if \next is . or ,
\if.\next {\it\argpgmline}% no \
\else\if,\next {\it\argpgmline}% no \
\else {\it\argpgmline\}% add \
\fi\fi}

```

## Times column

**22.** The optional column for the timing information (R5) gets its own dimen register.

```

⟨Dimensions 16⟩ +≡
\newdimen\timecolumnwidth % column for time

```

**23.** To allow entries like “ $A - 1$ ” the column must be wider than three symbols.

```

⟨Set values of dimen-registers 17⟩ +≡
{\setbox0=\hbox{\$2M+M\$}% 10pt gives white space
\global\timecolumnwidth=\wd0 }%

```

**24.** Of course, as the column is optional one more flag needs to be declared.

It is time to collect all definitions in a sorted list. Such a list might be easier to understand if the `mml.tex` file comes without this documentation, so some sub-entries are created.

```

⟨Definitions 24⟩ ≡
% %%% Definitions
⟨Counters 15⟩
⟨Dimensions 16⟩
⟨Flags 18⟩
\newif\iftimeinfostated % true: add time col
⟨Fonts 8⟩
⟨List symbols that are special in TEX 9⟩

```

See also section 39.

This code is used in section 4.

## Setting the output format

**25.** To identify the size of the field for the line numbers a hint must be given by the author of the program. This hint is a counter called `\mmixtype`. Three cases must be considered according to R1: The values 0 and 1 mean no line numbers are used, 2 and 3 stand for two-digit line numbers, and 4 and 5 for three-digit numbers.

And requirement R5 is also covered: If the value is odd the timing information is present: 0, 2, and 4 format the program without timing information, but 1, 3, and 5 have such information.

And a third bit of information is included: if the number is positive the line numbering starts immediately. Otherwise a command must be given to start the numbering.

```
<Counters 15> +=
\newcount\mmixtype % a value between -5 and 5
% -1,0,1: no line numbers, no space reserved
% absolute value 2,3: add column for 2 digits
% absolute value 4,5: add column for 3 digits
% > 1: start line numbering directly
% <-1: a user command starts numbering
% write a line ('!' is the commchar) with
% even value: label op expr ! comment
% odd value: label op expr ! time ! comment
```

**26.** I decided to set this counter by an “assignment” to the macro package. Therefore, a typical first line looks like the following lines in the comment.

```
<Description 14> +=
% start a programm with all \def's in one line
% (n is the \mmixtype explained elsewhere):
% \def\date{<date>}\def\author{<name>}
% \def\source{<volume, page>}\input mmix =n
```

**27.** The value of `\mmixtype` determines the values of all flags. They are set even when `\mmixtype` has a value outside the defined range from  $-5$  to  $+5$ . Such an error situation is tested and reported later.

```
<Format the mms file 19> +=
%%% Format
\def\setflagsformmixtype{% analyse \mmixtype
\ifnum\mmixtype<0
\mmixtype=-\mmixtype % wait with numbering
\else
\numberlinestrue % prep. to number 1st line
\fi
\ifnum\mmixtype>1 % activate numbering
\colforlnotrue
\ifnum\mmixtype>3 % use 3 digits
\threedigitlnotrue
\fi\fi
\ifodd\mmixtype % timing info is present
\timeinfo statedtrue
\fi}
```

**28.** In the case that `\mmixtype < -1` the numbering of lines is activated by user commands. They are placed in the comment of a source line.

```
<Format the mms file 19> +=
% start and stop line numbering
\let\startnumbering=\numberlinestrue
\let\stopnumbering=\numberlinesfalse
```

## Input format

**29.** How shall the MMIXAL program line of section 6 be entered into the input file? Requirements R2–R4 state that a monospaced font is used for the above defined elements 1–3. As special symbols of  $\TeX$  might be present the best way to typeset them is to use verbatim mode. My idea is to use a special character that ends the verbatim mode, which is automatically started in every line, and then to format the rest of the line in  $\TeX$ . Such a flag makes it possible to fulfill the requirements R6 and R7. I call this special character the *commchar* and by default the exclamation mark is used for it.

A single *commchar* is required if no timing information is present and two are used to identify the timing information. The above stated program line is therefore coded like this:

```
Maximum SL kk,$0,3 !! \step M1. Initialize.
The label, the op-code, and the expression must always start at the same column to be properly aligned in the output. The macros shouldn't destroy other input styles, for example, several MMIXAL statements might be written in one line (see R7).
```

Note: The control sequence `\step` is one of the useful macros defined later in this package.

```
<Format the mms file 19> +=
\def\setcommchar#1{% boundary for verbatim
\vskip-\baselineskip% for first end of line
\gdef\commchar{#1}%
\def\par{\endgraf\verbatim#1}}
```

**30.** The verbatim mode is defined in a standard way (see *The  $\TeX$ book* [5], pp. 380–382).

```
<Format the mms file 19> +=
% Verbatim macros
\def\verbatim{\begingroup % ends in \doverbatim
\setupverbatim
\doverbatim}
\def\setupverbatim{%
\def\par{\leavevmode\endgraf\noindent}%
\catcode'\ '= \active
\obeylines
\uncatcodespecials
\obeyspaces}
% now make a blank a control space
{\obeyspaces\global\let = \ }%
% and avoid ligatures of ? and ! with '
{\catcode'\ '= \active \gdef'\relax\lq}}%
```

**31.** When the verbatim mode is executed the tests for the line numbering and the timing information are made. The change of `\everypar` and `\par` is reverted in the comment field to allow, for example, a command like `\smallskip`.

Note that a missing second commchar with an odd `\mmixtype` results in a couple of errors (runaway argument) in `\printtimeinfo`, but forgetting the second commchar seems unlikely.

```
(Format the mms file 19) +=
\long\def\doverbatim#1{% #1 is the commchar
\everypar{\numbermmixline}%
\def\nextmmixline##1#1{\noindent
\tentt##1%
\endgroup % opened in \verbatim
\printtimeinfo}%
\iftimeinfostated
\gdef\printtimeinfo##1#1{% #1<time>#1
\hbox to \timecolumnwidth{%
\hss$ ##1 $hss}\resetpar}%
\else\global\let\printtimeinfo\resetpar
\fi
\nextmmixline}
\def\resetpar{\everypar{}}\let\par=\endgraf
\ignorespaces}
```

**32.** The exclamation mark seems to work fine as the commchar, but there might be reasons to switch the commchar in a program.

```
(Format the mms file 19) +=
\def\newcommchar#1{% change the commchar
\gdef\commchar{#1}%
\def\par{\endgraf\verbatim#1}% new \par
\obeylines}% end-of-line is the new \par
```

### Activation

**33.** In order to get all macros working together they must be called in a certain sequence. First, the macro `\setflagsformmixtype` has to be executed, which makes the value of `\mmixtype` positive and sets the flag for immediate line numbering. Next, the commchar must be defined. The command starts the verbatim mode after the first `\par` command and reads in the first line of the MMIXAL program. And of course the assignment statement for the `\mmixtype` must be executed. All this is done in the following way:

- With `\afterassignment` a (not yet opened) group is closed; the counter `\mmixtype` gets the value that appears after the `\input mmix`.
- A group is opened. It will be closed with the construction in a).
- With `\aftergroup` the following sequence is prepared:
  - `\setflagsformmixtype` is called;
  - `\setcommchar` is called (in a group);

- an exclamation mark is given as an argument for `\setcommchar`;
- `\obeylines` is called;
- and `\par` starts the verbatim mode.

d) Finally, `\global\mmixtype` is the left side of the assignment statement (see a)).

And here is the place where we call the test for the value of `\mmixtype`. If an error would be reported in the macro `\setflagsformmixtype` the user would see a bunch of tokens that have to be read again. Therefore the test for an error is made just before `\par` at the end of part c) is executed and the verbatim mode starts.

```
(Take off 33) ≡
% %%% Start
(Last-minute procedures 36)
\def\getmmixtype{% needs a ‘right side’
(Please prepare the environment 37)
\afterassignment\egroup% an assignment ends
\bgroupp % this group
\aftergroup\setflagsformmixtype
\aftergroup\beginngroup% closed in last line
\aftergroup\setcommchar
\aftergroup!% this is the default commchar
\aftergroup\obeylines
\aftergroup\testvalueofmmixtype
\aftergroup\par
\global\mmixtype}% now get \mmixtype
\getmmixtype
```

This code is used in section 4.

**34.** What has to be done to prepare the environment? Answer: set the fonts, initialize the variables (external and internal) and handle  $\TeX$  comments.

The `\ninepoint` macro of *The  $\TeX$ book* [5], pp. 414–415, does more than we need, but it shows how to switch to the 9 pt fonts.

```
(Format the mms file 19) +=
% switch to 9pt fonts
\def\setupfonts{\def\rm{\fam0\ninerm}%
\textfont0=\ninerm \scriptfont0=\sixrm
\textfont1=\ninei \scriptfont1=\sixi
\textfont2=\ninesy \scriptfont2=\sixsy
% no changes for fam3
\def\it{\fam\itfam\nineit}%
\textfont\itfam=\nineit
\def\sl{\fam\slfam\ninesl}%
\textfont\slfam=\ninesl
\def\tt{\fam\ttfam\ninettt}%
\textfont\ttfam=\ninett
\def\bf{\fam\bffam\ninebf}%
\textfont\bffam=\ninebf
\scriptfont\bffam=\sixbf
\def\oldstyle{\mit\ninei}%
\rm}
```

**35.** Here the variables are set to their initial values.

```

<Format the mms file 19> +≡
% set the variables to their default values
\def\setvariables{\mmixtype=0 \nocnt=0
  <Set values of dimen-registers 17>
  \colforlnofalse \numberlinesfalse
  \threedigitlnofalse \timeinfostatedfalse}

```

**36.** One problem remains: If a T<sub>E</sub>X comment is placed at the end of the comment field the end-of-line information is not available and the verbatim mode isn't restarted. So the % is made active. It gobbles the comment and behaves like the current definition of \par.

```

<Last-minute procedures 36> ≡
{\obeylines \catcode'\%=\active
  \gdef\handleTeXcomments{\catcode'\%=\active
    {\obeylines \gdef###1
      {\endgraf\expandafter\verbatim\commchar}}}}%
\def\resetTeXcomment{\catcode'\%=14 }

```

This code is used in section 33.

**37.** Now we collect the pieces together to prepare the environment.

```

<Prepare the environment 37> ≡
\checkextdata \setupfonts
\setvariables \handleTeXcomments

```

This code is used in section 33.

**38.** One task is open: To give a warning message if \mmixtype is out of range. I prefer to issue an error message. The following macro is called after \mmixtype was made positive.

```

<Format the mms file 19> +≡
\def\testvalueofmmixtype{% value must be < 6
  \ifnum\mmixtype>5
    \errhelp\mmixtypeerror
    \errmessage{The number \string\mmixtype
      \space must be between -5 and 5}%
  \fi}

```

**39.** We append the help message to the definitions.

```

<Definitions 24> +≡
% help messages
\newlinechar='\^^J
\newhelp\mmixtypeerror{%
  mmixtype is the number
  stated after \string\input\space mmix.^^J%
  It must be between -5 and 5.
  Three aspects are coded into it:^^J%
  if it is odd
    time information is given (use two !);^^J%
  if it is -1, 0, 1
    no line numbers are present;^^J%
  if it is -3, -2, 2, 3
    line numbers have two digits;^^J%
  if it is -5, -4, 4, 5
    line numbers have three digits;^^J%
  if it is >1
    immediate numbering of lines is started.}%

```

## Last line

**40.** At the end of the program source the possibility of including a separate file with the analysis of the algorithm shall be given (see R9). The name of the file was already defined above. The text shall be printed in a roman font of size 10pt. Therefore we must switch back to 10pt before that section can start.

```

<Add an analysis of the algorithm 40> ≡
% %%% Macros for the last line(s)
% typeset the Analysis of the Algorithm
\def\Analysis{\medbreak
  \def\rm{\fam0\tenrm}% back to 10pt
  \textfont0=\tenrm \scriptfont0=\sevenrm
  \textfont1=\teni \scriptfont1=\seveni
  \textfont2=\tensy \scriptfont2=\sevensy
  % fam3 was not changed
  \def\it{\fam\itfam\tenit}%
    \textfont\itfam=\tenit
  \def\sl{\fam\slfam\tensl}%
    \textfont\slfam=\tensl
  \def\tt{\fam\ttfam\tentt}%
    \textfont\ttfam=\tentt
  \def\bf{\fam\bffam\tenbf}%
    \textfont\bffam=\tenbf
    \scriptfont\bffam=\sevenbf
  \def\oldstyle{\mit\teni}%
  \rm % activate \tenrm
  \noindent{\tenbf Analysis}\par% the headline
  \nobreak\smallskip\noindent}

```

See also section 42.

This code is used in section 4.

**41.** The section with the analysis is started with the last line of the file. Similar to the first line it follows a special convention.

All programs have to use the control sequence \eop. It typesets a thick vertical rule as it is stated in requirement R8. It also stops the line numbering.

```

<Description 14> +≡
% use \eop in the comment of the last
% source line
% end the input file with a line that
% contains either (! is the commchar)
% ‘!\endprogram\bye’ (even \mmixtype)
% or ‘!!\endprogram\bye’ (odd \mmixtype)
% or use \endwAoA instead of \endprogram
% to input a file with an analysis

```

**42.** At the end of the input file a group is still open that must be closed. And % gets back its default meaning. But first, up to two empty hboxes are deleted (that might have been created on the current horizontal line) to avoid a line break in front of this “empty line”.

```

<Add an analysis of the algorithm 40> +≡
\def\eop{% end of program symbol
  \quadd\vrule height 7pt depth 1pt width 3pt
  \eopusedtrue\stopnumbering}

```

```

\def\clearline{% remove 0--2 empty hboxes
  {\setbox0=\lastbox \setbox0=\lastbox}}
\def\endprogram{\clearline
  \ifeopused\eopusedfalse
  \else\message{^^JWarning: end the program
    with \string\eop^^J}%
  \fi
  \endgroup% opened in \getmmixtype
  \resetTeXcomment}
\def\endwAoA{\endprogram\bigskip
  \Analysis \input\AoAfile}

```

43.  $\langle$ Flags 18 $\rangle + \equiv$   
`\newif\ifeopused % true: ‘‘\eop’’ was used`

### Shortcuts

44. When the analysis is written some commands for often-used idioms reduce the amount of typing.

```

 $\langle$ Useful commands and shortcuts 44 $\rangle \equiv$ 
% %% Useful commands
\def\MIX{{\ninett MIX}}
\def\MMIX{{\ninett MMIX}}
\def\MMIXAL{{\ninett MMIXAL}}
\let\NULL\Lambda % the null link
\def\AVAIL{\hbox{\ninett AVAIL}}% free space
\let\Gets\Leftarrow % get space from AVAIL
\let\implies\Rightarrow % more ‘‘logical’’
% units for the analysis: oops and mems
\def\oops{\hbox{\$}\upsilon\$}\let\oop=\oops
\def\mems{\hbox{\$}\mu\$} \let\mem=\mems
% reference to equation numbers of TAOCP
\def\numeq(#1){\hbox{\$({\oldstyle#1})\$}}
\def\eq(#1){% outputs Eq. (...)
  \hbox{Eq.\thinspace\numeq(#1)}}
\def\Eq(#1){% outputs Equation (...)
  \hbox{Equation \numeq(#1)}}

```

See also sections 45, 46, 48, 49, 50, 51, and 52.  
 This code is used in section 4.

45. Some commands and shortcuts are needed in the comments to a program. For example, the steps of an algorithm are labeled with the identifying letter of the algorithm, a number, and a phrase. This information is often stated in the comment to a program. (The phrase might be omitted; for example, see [3], Vol. 1, p. 236.)

```

 $\langle$ Useful commands and shortcuts 44 $\rangle + \equiv$ 
% steps in algorithms
\def\algidphrase#1#2#3.#4.{% #1 #arguments;
  #2 phrase delimiter; #3 step id; #4 phrase
  \$\underline{\hbox{\sl
    \vphantom{y}}#3.%
    \ifnum #1>1 \enspace #4#2\fi}}%
  \$\space}
\def\step#1.#2.{\algidphrase2.#1.#2.}
\def\steq#1.#2?{\algidphrase2?#1.#2.}
\def\stepid#1.{\algidphrase1-#1.-.}

```

46. Sometimes several lines get a single comment: In [3], Vol. 1, p. 258 and 278 (and in [4], p. 107)

a right brace is used to collect the statements for a comment. Place the command `\mlsc` (multiple lines, single comment) in the middle or just above the middle of the lines that get one comment.

```

 $\langle$ Useful commands and shortcuts 44 $\rangle + \equiv$ 
% place the command in (odd number) or
% just above (even number) the middle
\def\mlsc#1:#2{% #1 #lines; #2 comment
  \smash{\ifodd#1\else\lower.45\baselineskip\fi
    \hbox{\$% next line: see \TeX book, p.194
      \openup-1\jot % cancel for \eqalign
      \Compute \dimen255 from #1 (i.e., #lines) 47}
    \left.\kern-.5em % empty left brace
    \eqalign{\vrule height\dimen255
      width 0pt depth 0pt }%
    \right\}% visible right brace
  \$\thinspace#2}}

```

47. The height of the brace is of course roughly the number of lines, which are combined by the brace, multiplied by the `\baselineskip`. I use the formula:  
 number of lines  $\times$  (`\baselineskip` + 3pt) – 8pt.

```

 $\langle$ Compute \dimen255 from #1 (i.e., #lines) 47 $\rangle \equiv$ 
% compute height of brace
\dimen255=\baselineskip
\advance\dimen255 by 3pt
\multiply\dimen255 by #1\relax
\advance\dimen255 by -8pt

```

This code is used in section 46.

48. Next some special constructions: The dot minus (*monus operation* or *saturating subtraction*) is a binary operation defined by  $a \dot{-} b = \max(0, a - b)$ . It isn’t coded like `\doteq` as it is not a relation and some care must be taken with the position of the dot. The notation of the conditional expression is changed in TAOCP, Vol. 4a.

```

 $\langle$ Useful commands and shortcuts 44 $\rangle + \equiv$ 
% special operations
\def\dm{% dot minus: saturating subtraction
  \mathbin{\mathop{\kern0pt \smash{-}}}%
  \limits^{\raise.55ex\hbox{\$}\textstyle.\$}}
\def\ite(#1?#2:#3){% if-then-else; Vol.4a, p.96
  (#1\,{\rm?} \ }#2{\rm:}\enspace#3)}

```

49. The following shortcuts make special symbols of  $\TeX$  available for comments.

The plain  $\TeX$  command for `\l` is redefined here. The original definition is stored in `\lstroke`.

```

 $\langle$ Useful commands and shortcuts 44 $\rangle + \equiv$ 
% symbols of \TeX
\def\vs{{\tt\char32 }}% visible space
\def\bs{{\tt\char92 }}% backslash
\def\bo{{\tt\char123 }}% open brace
\def\bc{{\tt\char125 }}% close brace
\let\lstroke\l
\def\l_{{\tt\char95 }}% long underline
\def\h\#{\hbox{\$}\^{\#\$}}% high # (hex no.)

```

## More shortcuts

**50.** Here are some shortcuts that I find useful. In a comment short words must be processed in math mode either as roman text or monospaced text. So I define a couple of commands for that.

Often text must be placed in an hbox. And a short cut for an array with a roman or monospaced name and a math mode index is quite useful.

```
<Useful commands and shortcuts 44> +=
% my shortcuts
% output rm or tt in math with 1 to 3 chars
\def\r#1{\rm #1}
\def\rr#1#2{\rm #1#2}
\def\rrr#1#2#3{\rm #1#2#3}
\def\m#1{\tt #1}
\def\mm#1#2{\tt #1#2}
\def\mmm#1#2#3{\tt #1#2#3}
% output of rm or tt text in boxes or arrays
\def\rb#1{\hbox{\rm #1}}% rm box
\def\mb#1{\hbox{\tt #1}}
\def\ra#1[#2]{\hbox{\rm #1[#2$]}}% rm array
\def\ma#1[#2]{\hbox{\tt #1[#2$]}}
```

**51.** I add a comment in front of a subroutine or procedure. A few lines describe the calling sequence, the entry and exit conditions, and changed special or global registers. This is described on page 55 of [4]. I start such comments indented at the column of the op-code and with a > that sticks out to the left. To get this alignment in the case when timing information is present some care must be taken.

```
<Useful commands and shortcuts 44> +=
\def\gts{% align 'g' with the op-code col
\iftimeinfostated
  {% omit time column if \mmixtype is odd
  \ninerm\hskip-\timecolumnwidth
  {\tentt\ }}% add space for 2nd commchar
\fi
{\tentt>\space}}
```

## Final remarks

**52.** The following command doesn't produce any output. Nevertheless I find it useful in the analysis of the algorithm. The timing information states how often a source line is executed but for a line with a branch instruction it is also useful to know how often a bad branching decision was made.

Therefore I place the following command directly after the second commchar and state the number of bad decisions.

```
<Useful commands and shortcuts 44> +=
% used to state number of bad decisions
\def\bad#1\bad{\ignorespaces}% no output
```

**53.** The macros have been presented for a single mms file. But the conversion project needs to rewrite

many MIX programs. So, to create a book of converted programs, for example, for a complete chapter of TAOCP, the individual files can be `\input` in a main file which is then processed by `TEX`. (Of course the main file must include a definition like, for example, `\let\goodbye=\bye` and then the redefinition of `\bye`: `\outer\def\bye{\par\vfill\supereject\endinput}`.)

To avoid reloading this package a test is added that determines if the package is already known. And all the counters, fonts etc. are reset to their initial value.

Note that `\endinput` and `\fi` must appear in the same line (see *The T<sub>E</sub>Xbook* [5], p. 214).

```
<Description 14> +=
% my shortcuts
% don't load the file several times
% but reset variables, fonts etc.
\def\ifundef #1 {% see \TeX book, ex. 7.7
  \expandafter\ifx\csname #1\endcsname\relax}
\ifundef \mmixisloaded \def\mmixisloaded{true}%
\else\getmmixtype\endinput\fi
```

**54.** To test the scripts and to give an example of how to use the macro package a small example is shown in the appendix.

## Index and List of sections

**55.** A literate program comes usually with an index of the names of used identifiers — variables, types, functions, procedures, or whatever the used programming language offers. It includes also certain aspects of the program that might be of interest to users or developers who want to change the code. For example, error messages are listed.

The index lists the section numbers, in which the entry appear. The section number, in which an identifier is defined, is written in slanted digits.

(space): 30	\commchar: 29, 32, 33, 36
?: 36	conditional expression: 48
': 30	\date: 13, 14, 26
\algidphrase: 45	\dateF: 11, 13
\Analysis: 40, 42	default values: 12, 13, 17, 23,
\AoAfile: 12, 42	29, 33
\argpgmline: 21	\dm: 48
\author: 13, 14, 26	\do: 9, 10
\authorH: 11, 13	\doverbatim: 30, 31
\AVAIL: 44	\endprogram: 41, 42
\bad: 52	\endwAoA: 41, 42
\bc: 49	\eop: 41, 42
\bo: 49	\eopusedfalse: 42
\bs: 49	\eopusedtrue: 42
changed plain T <sub>E</sub> X	\Eq: 44
commands: 30, 36, 49	\eq: 44
\checkextdata: 13, 37	\everypar: 31
\clearline: 42	\footline: 11
\colforlnofalse: 35	\getmmixtype: 33, 42, 53
\colforlnottrue: 27	\Gets: 44

`\gts`: 51  
`\h`: 49  
`\handleTeXcomments`: 36, 37  
`\headline`: 11  
`\ifcolforlno`: 18, 20  
`\ifeopused`: 42, 43  
`\ifnumberlines`: 18, 20  
`\ifthree digitlno`: 18, 19  
`\iftimeinfostated`: 24, 31, 51  
`\ifundef`: 13, 53  
`\implies`: 44  
`\ite`: 48  
Knuth, Donald Ervin: 1  
`\l`: 49  
`\lncnt`: 15, 19, 20, 35  
`\lnothreedigitwidth`: 16, 17, 19  
`\lnotwodigitwidth`: 16, 17, 19  
`\lstroke`: 49  
`\m`: 50  
`\ma`: 50  
`\mb`: 50  
`\mem`: 44  
`\mms`: 44  
MIX: 1  
`\MIX`: 44  
`\mlsc`: 46  
`\mm`: 50  
MMIX: 1  
MMIX home page: 3  
`\MMIX`: 44  
`mmix.tex`: 4, 5, 24  
`MMIXAL`: 6, 9, 29  
`\MMIXAL`: 44  
`\mmixdospecials`: 9, 10  
`\mmixisloaded`: 53  
`\mmixtype`: 25, 26, 27, 33, 35, 38, 41, 51  
`\mmixtypeerror`: 38, 39  
`\mmm`: 50  
mms files: 2, 53  
`\newcommchar`: 32  
`\newlinechar`: 39  
`\next`: 21  
`\nextmmixline`: 31  
`\ninebf`: 8, 34  
`\ninei`: 8, 34  
`\nineit`: 8, 17, 34  
`\niner`: 8, 34, 51  
`\ninesl`: 8, 34  
`\ninesy`: 8, 34  
`\ninett`: 8, 34, 44  
`\NULL`: 44  
`\numberlinesfalse`: 28, 35  
`\numberlinestrue`: 27, 28  
`\numbermmixline`: 20, 31  
`\numeq`: 44  
`\oop`: 44  
`\oops`: 44  
`\par`: 29, 30, 31, 32, 33  
`\pgmline`: 21  
`\pgmlinex`: 21  
`\pgmname`: 13, 14  
`\pgmnameH`: 11, 12, 13  
`\printlinenumber`: 19, 20  
`\printtimeinfo`: 31  
`\r`: 50  
`\ra`: 50  
`\rb`: 50  
`\resetpar`: 31  
`\resetTeXcomment`: 36, 42  
`\rr`: 50  
`\rrr`: 50  
Runaway argument: 31  
saturating subtraction: 48  
`\setcommchar`: 29, 33  
`\setflagsformmixtype`: 27, 33  
`\setupfonts`: 34, 37  
`\setupverbatim`: 30  
`\setvariables`: 35, 37  
`\sixbf`: 8, 34  
`\sixi`: 8, 34  
`\sixrm`: 8, 34  
`\sixsy`: 8, 34  
`\source`: 13, 14, 26  
`\sourceH`: 11, 13  
`\startnumbering`: 28  
`\step`: 45  
`\stepid`: 45  
`\steq`: 45  
`\stopnumbering`: 28, 42  
TAOCP: 1, 2, 6, 45, 46, 48  
`\testvalueofmmixtype`: 33, 38  
`\texdospecials`: 9, 10  
The number `\mmixtype` ...: 38  
`\threedigitlnofalse`: 35  
`\threedigitlnotrue`: 27  
`\timecolumnwidth`: 22, 23, 31, 51  
`\timeinfostatedfalse`: 35  
`\timeinfostatedtrue`: 27  
`\uncatcodespecials`: 10, 30  
usage, first line: 14, 26  
last line: 41  
program lines: 25, 29, 41  
value of `\mmixtype`: 25, 39  
user commands: 14, 21, 28, 32, 42, 44, 45, 46, 48, 49  
my collection: 50, 51, 52  
`\verbatim`: 29, 30, 31, 32, 36  
`\vs`: 49  
Warning: end the ...: 42  
Wermuth, Udo: 5

**56.** The second index collects all headlines of the code parts. Here the headlines contain all section numbers that define the replacement code for the section name.

⟨Add an analysis of the algorithm 40, 42⟩ Used in 4.

Udo Wermuth

⟨Compute `\dimen255` from #1 (i.e., `\lines`) 47⟩ Used in 46.  
⟨Counters 15, 25⟩ Used in 24.  
⟨Definitions 24, 39⟩ Used in 4.  
⟨Description 14, 26, 41, 53⟩ Used in 5.  
⟨Dimensions 16, 22⟩ Used in 24.  
⟨Flags 18, 43⟩ Used in 24.  
⟨Fonts 8⟩ Used in 24.  
⟨Format the mms file 19, 20, 21, 27, 28, 29, 30, 31, 32, 34, 35, 38⟩ Used in 4.  
⟨Initialization 5, 11, 12, 13⟩ Used in 4.  
⟨Last-minute procedures 36⟩ Used in 33.  
⟨List symbols that are special in  $\TeX$  9, 10⟩ Used in 24.  
⟨`mmix.tex` 4⟩ Root.  
⟨Prepare the environment 37⟩ Used in 33.  
⟨Set values of `\dimen`-registers 17, 23⟩ Used in 35.  
⟨Take off 33⟩ Used in 4.  
⟨Useful commands and shortcuts 44, 45, 46, 48, 49, 50, 51, 52⟩ Used in 4.

## References

- [1] Bart Childs, “Thirty years of literate programming and more?” *TUGboat* **31**(2010), 183–188.  
<http://tug.org/TUGboat/tb31-2/tb98childs.pdf> (accessed: August 4, 2014)
- [2] Donald E. Knuth, *Literate Programming*, CSLI Lecture Note No. 27, 1992.  
<http://www-cs-staff.stanford.edu/~uno/lp.html> (accessed: August 4, 2014)
- [3] Donald E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Vol. 1 (3rd ed.), 1997; Vol. 2 (3rd ed.), 1998; Vol. 3 (2nd ed.), 1998; Vol. 4a (1st ed.), 2011.  
<http://www-cs-staff.stanford.edu/~uno/taocp.html> (accessed: August 4, 2014)
- [4] Donald E. Knuth, *The Art of Computer Programming — MMIX: A RISC Computer for the new Millennium*, Vol. 1, Fascicle 1, Addison-Wesley, 2005.  
<http://www-cs-staff.stanford.edu/~uno/mmix.html> (accessed: August 4, 2014)
- [5] Donald E. Knuth, *The  $\TeX$ book*, Volume A of *Computers & Typesetting*, Addison-Wesley, 1984.
- [6] MMIX home page, hosted by: The MMIX Group at Munich University of Applied Sciences.  
<http://mmix.cs.hm.edu> (accessed: August 4, 2014)
- [7] Norman Ramsey, “Literate programming simplified”, *IEEE Software* **11** (1994), 97–105.  
<http://www.cs.tufts.edu/~nr/nobew/> (accessed: August 4, 2014)

◊ Udo Wermuth  
Babenhäuser Straße 6  
63128 Dietzenbach  
Germany  
u dot wermuth (at) icloud dot com

## Appendix: An Example

First the input (file 1-3-3I.mms) is shown. Note that commchars are used only in lines that contain a comment. Line numbers and timing information are given only for the lines that belong to the subroutine.

The value of `\mmixtype` is `-3` meaning that (a) the first line is not numbered (the value is negative), (b) line numbers need only two digits (i.e., value is `-2` or `-3`), and (c) timing information is given (so value must be odd).

```
\def\date{04 Aug 2014}\def\source{V1, p.\ 177}\def\author{Udo Wermuth}\input mmix =-3
!!\clearline{\tenbf Program I} ({\tenit Inverse in place\})% use a lot of ‘features’
!!\clearline\smallskip\timecolumnwidth=2.5em % (some are not necessary in this conversion)
n      GREG      6          !! Number of elements in the permutation
j      IS        $0        !! Variables of the algorithm
i      IS        $1
mm     IS        $2        !! $\mm mm = 8m$
      LOC      Data_Segment
X      GREG      @
      OCTA     0          !! $X[0]$ is not used
      OCTA     6,2,1,5,4,3 !! The data of Table 1.3.3--3
      LOC      #100
      !!\gts Inverse a permutation in place
      !!\gts Entry condition: $X[1]\,\ldots\,X[n]$ is a permutation of $\{1,\ldots,n\}$
      !!\gts Exit condition: array $$X$ contains inverted permutation \startnumbering
:Invert SL      mm,n,3      !! \step I1. Initialize. $m\gets n$.
      NEG      j,1          !! $j\gets-1$.
2H      LDO      i,X,mm     !N! \step I2. Next element. $i\gets X[m]$.
      PBN      i,5F         !N!\bad C\bad To I5 if $i<0$.
3H      STO      j,X,mm     !N! \step I3. Invert one. $X[m]\gets j$.
      SR       j,mm,3      !N! \mlsc 2:{$j\gets-m$} % multi-line comment
      NEG      j,j          !N!
      SL       mm,i,3      !N! $m\gets i$.
      LDO      i,X,mm     !N! $i\gets X[m]$.
4H      PBP      i,3B       !N!\bad C\bad \steq I4. End of cycle? To I3 if $i>0$.
      SET      i,j          !C! Otherwise set $i\gets j$.
5H      NEG      i,i        !N! \step I5. Store final value.
      STO      i,X,mm     !N! $X[m]\gets-i$. \newcommchar. % change the commchar
6H      SUB      mm,mm,8    .N. \step I6. Loop on $m$.
      PBP      mm,2B       .N.\bad 1\bad To I2 if $m>0$. \stopnumbering
* inspect memory locations of array X for the result
      TRAP     0,Halt,0
Main   IS      :Invert     .. \eop
..\endwAoA\bye
```

The last line of the input file ends the source with `\endwAoA`. So a second file with the analysis of the algorithm is needed; it is the file 1-3-3I\_aoa.tex:

In step~I3 each slot of the array~\$\$X\$ once receives a negative value and in step~I5 it is filled with a positive number. Using Kirchhoff’s law the number of times step~I6 is executed is equal to the number of times steps~I5 and~I2 are executed; that is steps~I2 and~I6 have count \$N\$. Step~I5 is entered from I4 \$C\$~times, so I2 goes \$N-C\$~times to step~I5 and \$C\$~times to~I3. And step~I3 goes \$N\$~times to step~I4, which must return \$N-C\$~times to~I3.

Of course, \$N\$ is the number of elements in the permutation and~\$C\$ is the number of its cycles. The `\mb{PB..}` instructions in lines~\pgmline{04} and~\pgmline{10} are based on the assumption that in most cases \$C\leq N/2\$. An analysis of~\$C\$ shows that its average value is the harmonic number~\$H\_n\$. So the assumption is correct.

The program needs \$4N\mems + (12N+5C+4)\oops\$. The execution with the test data, the permutation \$(4\ 5)(2\ 1\ 6\ 3)\$, gives the statistic for `\mb{Invert}`: `{\tt 78~instructions, 24~mems, 91~oops; 11~good guesses, 7~bad}`. (The total run time is `96\oops` as the `\mb{TRAP}` instruction needs `5\oops`.) As in this case \$N=6\$ and \$C=3\$ the above formula calculates \$(4\times 6)\mems=24\mems\$ and \$(12\times 6+5\times 3+4)\oops=(72+15+4)\oops=91\oops\$ in agreement with the measured data.

And this shows the final output (with simulated headline and footline). I assume that in a TAOCP volume only the numbered lines appear.

Author: Udo Wermuth

Program: 1-3-3I.mms (V1, p. 177)

**Program I** (*Inverse in place*)

n	GREG	6	Number of elements in the permutation
j	IS	\$0	Variables of the algorithm
i	IS	\$1	
mm	IS	\$2	mm = 8m
	LOC	Data_Segment	
X	GREG	@	
	OCTA	0	X[0] is not used
	OCTA	6,2,1,5,4,3	The data of Table 1.3.3-3
	LOC	#100	

> Inverse a permutation in place  
> Entry condition: X[1] ... X[n] is a permutation of {1, ..., n}  
> Exit condition: array X contains inverted permutation

01	:Invert	SL	mm,n,3	1	<u>I1. Initialize.</u> m ← n.
02		NEG	j,1	1	j ← -1.
03	2H	LDO	i,X,mm	N	<u>I2. Next element.</u> i ← X[m].
04		PBN	i,5F	N	To I5 if i < 0.
05	3H	STO	j,X,mm	N	<u>I3. Invert one.</u> X[m] ← j.
06		SR	j,mm,3	N	} j ← -m.
07		NEG	j,j	N	
08		SL	mm,i,3	N	m ← i.
09		LDO	i,X,mm	N	i ← X[m].
10	4H	PBP	i,3B	N	<u>I4. End of cycle?</u> To I3 if i > 0.
11		SET	i,j	C	Otherwise set i ← j.
12	5H	NEG	i,i	N	<u>I5. Store final value.</u>
13		STO	i,X,mm	N	X[m] ← -i.
14	6H	SUB	mm,mm,8	N	<u>I6. Loop on m.</u>
15		PBP	mm,2B	N	To I2 if m > 0.

\* inspect memory locations of array X for the result  
TRAP 0,Halt,0  
Main IS :Invert

**Analysis**

In step I3 each slot of the array X once receives a negative value and in step I5 it is filled with a positive number. Using Kirchhoff's law the number of times step I6 is executed is equal to the number of times steps I5 and I2 are executed; that is steps I2 and I6 have count N. Step I5 is entered from I4 C times, so I2 goes N - C times to step I5 and C times to I3. And step I3 goes N times to step I4, which must return N - C times to I3.

Of course, N is the number of elements in the permutation and C is the number of its cycles. The PB.. instructions in lines 04 and 10 are based on the assumption that in most cases  $C \leq N/2$ . An analysis of C shows that its average value is the harmonic number  $H_n$ . So the assumption is correct.

The program needs  $4N\mu + (12N + 5C + 4)v$ . The execution with the test data, the permutation (45)(2)(163), gives the statistic for Invert: 78 instructions, 24 mems, 91 oops; 11 good guesses, 7 bad. (The total run time is  $96v$  as the TRAP instruction needs  $5v$ .) As in this case  $N = 6$  and  $C = 3$  the above formula calculates  $(4 \times 6)\mu = 24\mu$  and  $(12 \times 6 + 5 \times 3 + 4)v = (72 + 15 + 4)v = 91v$  in agreement with the measured data.

## A Citation Style Language (CSL) workshop

Daniel Stender

### Abstract

CSL is a free and open XML-based language for the programming of citation styles. With these styles, bibliographical references can be printed out in different ways from several database formats, including  $\text{\LaTeX}$ . The so-far over 7000 CSL styles which are currently available can be used with several popular applications like Zotero, Mendeley, or Pandoc. This article is an introduction into the programming of citation styles with CSL, based on a few example  $\text{\LaTeX}$  bibliographic database records.

### 1 Introduction

Bibliographical references, as used in scientific publications, are pointers to cited or regarded literature. Regularly, they consist of two standardized components: an in-line citation (the “cite”) refers to an entry in the publication’s bibliography. Despite the common concepts, there is no uniform outline for references; rather, each scientific discipline and every publishing house has its own traditional set of conventions, which also might change between series.

In electronic typesetting, bibliographical information is often gathered in comprehensive, reusable data files. CSL<sup>1</sup> is a programming language for citation styles, with which differently formatted references can be generated from the same bibliographic databases. CSL (current version: 1.0.1) is XML-based, open and free, and was substantially developed for the all-around reference manager Zotero.<sup>2</sup>

This article demonstrates how a rudimentary example citation style could be implemented with CSL, with reference to the  $\text{\LaTeX}$  data format. The usage of several programs refers to a Debian GNU/Linux based system (like Ubuntu and Linux Mint), but CSL styles could also be easily developed on other operating systems. Some basic knowledge of XML and the  $\text{\LaTeX}$  data format is definitely needed to follow every detail.

### 2 CSL styles

The citation styles which have already been implemented in CSL (file extension: `.csl`) are collected by the CSL developers in the official style repository,<sup>3</sup> and in the Zotero style repository.<sup>4</sup> The so far more than 7000 styles are distributed under the

<sup>1</sup> <http://citationstyles.org/>

<sup>2</sup> <http://www.zotero.org/>

<sup>3</sup> <http://github.com/citation-style-language/styles/>

<sup>4</sup> <http://zotero.org/styles/>

*Attribution-ShareAlike 3.0 Unported* license by Creative Commons.<sup>5</sup> For the search of specific citation styles the online CSL style editor<sup>6</sup> is quite useful because it provides, in addition to other features, a search by example.

### 3 Pandoc-citeproc

Among the several current applications which already know how to use CSL styles for the automatic generation of references, there is the popular universal markup converter Pandoc [1].<sup>7</sup> With short citation keys like `@doe2014 [p. 40–42]` for its extended Markdown lightweight markup, Pandoc can query bibliographical data files and recognize CSL styles to put out variously formatted references for documents either in HTML,  $\text{\LaTeX}$  or  $\text{\ConTeXt}$  markup, along with several others [3]. Pandoc is a command line interface application; thus, the CSL style which is going to be processed and the bibliographical database(s) are given as arguments in the program call. Here’s an example (some line breaks are editorial) for Pandoc’s  $\text{\LaTeX}$  output of a random  $\text{\LaTeX}$  database record (see below for details), formatted using `chicago-author-date.csl`:<sup>8</sup>

```
$ echo "On this, see @reference2 [p. 127]." \
  | pandoc --to=latex \
    --csl=chicago-author-date.csl \
    --bibliography=references.bib
```

On this, see Flom (2007, 127).

```
Flom, Peter. 2007. ‘‘LaTeX for Academics and
Researchers Who (Think They) Don’t Need It.’’
\emph{TUGboat} 28 (1): 126--128.
```

As shown in this example, the bibliography is printed at the end of a document. Incidentally, in the input (shown later), the title is given in lowercase; the titlecasing done here is automatic, a feature of this style [8, chp. 14].

Processors which produce formatted citations out of bibliographic databases according to CSL styles are called CiteProcs.<sup>9</sup> CiteProcs are being developed in several programming languages. The one which is used normally by Pandoc is *pandoc-citeproc*,<sup>10</sup> written in the same functional programming language Haskell as Pandoc itself, and developed closely together with it. This CiteProc (currently: 0.3.0.1) already deals with a number of different database

<sup>5</sup> <http://creativecommons.org/licenses/by-sa/3.0/>

<sup>6</sup> <http://editor.citationstyles.org/about/>

<sup>7</sup> <http://johnmacfarlane.net/pandoc/>

<sup>8</sup> Although given explicitly here, this CSL style is the default in Pandoc.

<sup>9</sup> <http://en.wikipedia.org/wiki/CiteProc/>

<sup>10</sup> <http://github.com/jgm/pandoc-citeproc/>

formats, but it's said that it works best with  $\text{BIB}\text{T}\text{E}\text{X}$  resp.  $\text{BIB}\text{L}\text{A}\text{T}\text{E}\text{X}$  [4] databases so far.<sup>11</sup>

#### 4 Developing CSL styles

CSL styles are XML, and therefore the whole related XML tool chain can also be used with them. For somebody who deals with XML regularly, a specialized editor is useful, but fundamentally CSL styles can be created and modified with any text editor. CSL is described in detail in the specification [10], and the primer which has been written by the developers [9] is a good starting point for beginners.

CSL is standardized as an XML grammar in the schema language RELAX NG,<sup>12</sup> and in principle any CSL style file can be checked with an XML validator against the CSL schema to determine if it is correct (valid).<sup>13</sup> Unfortunately, some validators, for example *xmllint* (Debian package: *libxml2-utils*), cannot cope with XML schemes in RELAX NG compact syntax like the one shipped by the CSL developers (file extension: *.rnc*); the scheme has to be converted (e.g. with *Trang*) into the regular RELAX NG syntax (file extension: *.rng*) before a CSL style can be validated against it:

```
$ git clone https://github.com/\
citation-style-language/schema.git
Cloning into 'schema' [...]
$ trang schema/csl.rnc schema/csl.rng
$ xmllint --noout -relaxng schema/csl.rng \
  chicago-author-date.csl
chicago-author-date.csl validates
```

#### 5 XML declaration and CSL header

The standard XML declaration commonly starts a CSL style file:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Incidentally, although it is often suggested to be included, specifying the UTF-8 encoding like this could be omitted because UTF-8 is the XML default [2, p. 28].

The next thing which is needed is a well-formed CSL header to specify that the XML file is a CSL style. A standard CSL header goes like this:

```
<style xmlns="http://purl.org/net/xbiblio/csl"
  version="1.0" class="in-text">
```

This defines  $\langle style \rangle$  to be the root element, and the CSL namespace is set as the default with *xmlns*.<sup>14</sup> That the style is compatible with CSL 1.0 is stated

<sup>11</sup> cf. *Readme.md*.

<sup>12</sup> <http://relaxng.org/>

<sup>13</sup> <http://github.com/citation-style-language/schema/tree/v1.0.1/>

<sup>14</sup> At no time will a connection to this url be established; it's only for the purpose of identification.

by *version*, while the *class* attribute determines that this style provides cites in the running text by default, rather than as footnotes or end notes (which would be “note”).

#### 6 Info block

The next mandatory unit of a CSL style is an  $\langle info \rangle$  block, which provides metadata for labeling and identification. A typical info block looks like this:

```
<info>
  <title>An example CSL style</title>
  <id>http://www.danielstender.com/csldemo</id>
  <updated>2014-09-18T23:53:00+02:00</updated>
</info>
```

Even if it is not intended to publish the style, there are at least three mandatory child elements that are meant for this purpose:

$\langle title \rangle$  is the title of the CSL style as it is going to be displayed to users,

$\langle id \rangle$  contains, like *xmlns* in the CSL header (see above), a random URI, which may be real or fictitious, and which is solely for identification purposes,

$\langle updated \rangle$  carries a *xsd:dateTime* compliant time stamp<sup>15</sup> of the last modification.

#### 7 Example $\text{BIB}\text{T}\text{E}\text{X}$ records

Before getting any deeper into CSL style programming, here are a few sample  $\text{BIB}\text{T}\text{E}\text{X}$  records to be referred to hereafter to demonstrate how CSL works. A typical  $\text{BIB}\text{T}\text{E}\text{X}$  entry type [5, chp. 13.2] goes like this

```
@Book{reference1,
  author   = {Kopka, Helmut and Daly, Patrick W.},
  title    = {A Guide to LaTeX and Electronic
             Publishing},
  publisher = {Addison-Wesley},
  year     = 2004,
  address  = {Boston},
  edition  = {Fourth}}
```

The next one is an  $\text{BIB}\text{T}\text{E}\text{X}$  data set of a (well-known) journal whose issues are counted as volume numbers:

```
@Article{reference2,
  author = {Flom, Peter},
  title  = {LaTeX for academics and researchers
             who (think they) don't need it},
  journal = {TUGboat},
  year    = 2007,
  volume  = 28,
  number  = 1,
  pages   = {126-128}}
```

<sup>15</sup> <http://books.xmlschemata.org/relaxng/ch19-77049.html>

And another `@Article` data set of a journal which appears monthly and doesn't have any issue numbers:

```
@Article{reference3,
author   = {Sharma, Tushar},
title    = {Why I never close Emacs},
journal  = {Open Source For You},
year     = 2014,
pages    = {53-55},
month    = {jan}}
```

The main body of a CSL style consists of several instructions for how exactly the CiteProc is going to typeset citations, thus the cites and the corresponding references in the bibliography of a document, from standardized data records like these.

## 8 The cite

Another child element of `<style>` is `<macro>`. Multiple macros are permitted, and this element serves to deliver data and reusable formatting sets. The following macro provides the author surnames out of the `BIBTEX` data field `author`:

```
<macro name="surname">
  <names variable="author">
    <name form="short" and="symbol"/>
  </names>
</macro>
```

From the CSL variable `author`, which has the same name as the `BIBTEX` data field and initially also contains the full author names, `<name form="short"/>` extracts the surnames, while `symbol` for the attribute `and` specifies the ampersand as delimiter between multiple authors. Macro names (like “surname”) are user-defined. Their result could be typeset with the rendering element `<text>` (see below), but macros have necessarily to be written *before* they are referenced (the XML parser needs it this way).

Another macro fetches the year of the publication out of the date variable<sup>16</sup> `issued`:

```
<macro name="year">
  <date variable="issued">
    <date-part name="year"/>
  </date>
</macro>
```

In CSL, the year of the publication from the `BIBTEX` data field `year` isn't available independently; for example, `issued` also contains the `BIBTEX` field `month`, if the data item holds that. Thus, the year of the publication has to be extracted from `issued` like this, before it is available.

With these two macros it's already possible to set up an author–year cite. Also needed is the standard variable `locator`, which provides the page refer-

ence out of the citation key in the document source. The cite is described within the element `<citation>`, looking, for example, like this:

```
<citation>
  <layout>
    <text macro="surname" suffix=" "/>
    <group prefix="(" suffix=")">
      <text macro="year"/>
      <text variable="locator" prefix=":\,"/>
    </group>
  </layout>
</citation>
```

Along with the `<layout>` element, `<citation>` has another child for sorting, `<sort>`, which defines the order of multiple references within one and the same cite. When `<sort>` isn't used, the order of appearance is kept as is the case here.

With a `<citation>` block like this one the bibliographic records above could be cited easily, for example again with Pandoc:

```
$ echo "@reference1 [p. 100], @reference2
[p. 127], @reference3 [54-55]." \
  | pandoc --to=latex --cs1=example.cs1 \
    --bibliography=references.bib
```

```
Kopka & Daly (2004:\,100), Flom (2007:\,127),
Sharma (2014:\,54--55).
```

The parentheses and colon punctuation were set up using the `prefix` and `suffix` attributes of the `<group>` element, above.

## 9 The bibliography

In CSL, how the full records appear is defined within the `<bibliography>` block. Using the bibliographical records above as an example, we will set up a citation style resulting in the following (with `\frenchspacing` in effect):

```
[Kopka & Daly 2004] Helmut Kopka, Patrick W.
Daly: “A Guide to LaTeX and Electronic Publishing”.
Fourth edition. Boston: Addison-Wesley 2004.
```

```
[Flom 2007] Peter Flom: “LaTeX for academics and
researchers who (think they) don't need it”. In: TUG-
boat 28,1 (2007), p. 126–128.
```

```
[Sharma 2014] Tushar Sharma: “Why I never close
Emacs”. In: Open Source For You 1/2014, p. 53–55.
```

The individual elements of this style can be implemented in CSL as follows. As was the case with `<citation>`, everything which is going to be printed has to be put within the `<layout>` child element of `<bibliography>` (see p. 313 for the complete code).

<sup>16</sup> The CSL variables are divided into four different classes: standard, number, date, and name.

### 9.1 Label

The first element that is going to be set up is a label which matches the cite. That’s for the purpose of easily locating the corresponding record in the bibliography. As we did with `<citation>` above, a label like this can be constructed using the macros `surname` and `year`:

```
<group delimiter=" " prefix=[" suffix="] ">
  <text macro="surname"/>
  <text macro="year"/>
</group>
```

### 9.2 Authors

As explained above, the CSL variable `author` contains the full author names. This information can easily be output directly with the rendering element `<names>`, when it is needed:

```
<names variable="author" delimiter=", "
  suffix=":"/>
```

### 9.3 Title

The variable `title` is meant to be used with the rendering element `<text>`:

```
<text variable="title" prefix="“" suffix="”."/>
```

Features like the wanted quotation marks are implemented in Unicode.<sup>17</sup> The CiteProc converts them into L<sup>A</sup>T<sub>E</sub>X standard quotes, if this is chosen as the output format (see below for the result).

### 9.4 container-title

The variable `container-title` represents the BIB<sub>T</sub>E<sub>X</sub> data field `journal`:

```
<text variable="container-title" prefix="In: "
  font-style="italic"/>
```

This variable remains empty when entry types like `@Book` are processed, and therefore nothing would be printed out in this case.

### 9.5 Journal issue

The following macro extracts the month of publication, which comes from the BIB<sub>T</sub>E<sub>X</sub> field `month`, from the date variable `issued`:

```
<macro name="month">
  <date variable="issued">
    <date-part name="month" form="numeric"/>
  </date>
</macro>
```

In the same process, the month of appearance, which is commonly recorded in BIB<sub>T</sub>E<sub>X</sub> in the form of “jan”,

<sup>17</sup> These are the Unicode entities U+201C LEFT DOUBLE QUOTATION MARK and U+201D RIGHT DOUBLE QUOTATION MARK.

“feb”, etc.,<sup>18</sup> gets converted into the corresponding number by setting the attribute `form` to `numeric`.

Using this macro, we can define a second macro to render the desired output of the journal issue either as “*volume,number (year)*”, or as “*month/year*” for journals appearing monthly:

```
<macro name="issue">
  <choose>
    <if type="article-journal" variable="volume">
      <text variable="volume"/>
      <text variable="issue" prefix=","/>
      <text macro="year" prefix=" (" suffix=")"/>
    </if>
    <else-if type="article-journal">
      <text macro="month"/>
      <text macro="year" prefix=""/>
    </else-if>
  </choose>
</macro>
```

The CSL variable `issue` represents the BIB<sub>T</sub>E<sub>X</sub> field `number`. This macro checks whether the publication type is `article-journal` (a condition which is satisfied by `@Article` entry types of BIB<sub>T</sub>E<sub>X</sub>), and, depending on whether `volume` data for the record is available, prints the element of the reference in the desired way.

Generally, it’s best to keep formatting switches dependent on publication type out of `<bibliography>`,<sup>19</sup> but it’s fine to have a macro like this:

```
<text macro="issue" suffix=","/>
```

### 9.6 Page numbers

The page numbers in a BIB<sub>T</sub>E<sub>X</sub> data set are directly available through the rendering variable `page`. If a record carries any, they can be output with, for example:

```
<text variable="page" prefix="p. "/>
```

### 9.7 Edition

The same is true for `edition`, which also could be rendered directly, like this:

```
<text variable="edition" suffix=" edition."/>
```

However, although in theory it ought to be homogeneous,<sup>20</sup> when dealing with data files from different origins the literal BIB<sub>T</sub>E<sub>X</sub> field `edition` often carries

<sup>18</sup> “For reasons of consistency the standard three-letter abbreviations (jan, feb, mar, etc.) should be used” [5, p. 765].

<sup>19</sup> “The use of macros can improve style readability, compactness and maintainability. It is recommended to keep the contents of `cs:citation` and `cs:bibliography` compact and agnostic of item types (e.g., books, journal articles, etc.) by depending on macro calls” [10].

<sup>20</sup> “However, the edition field poses a bit of a challenge. The BIB<sub>T</sub>E<sub>X</sub> standard way of specifying edition numbers is to use ordinal words with capital first letters such as “First”, “Second”, “Third” and so forth” [7, Q13].

Figure 1: The full code of `<bibliography>`

```

<bibliography>
  <sort>
    <key macro="surname"/>
  </sort>
  <layout suffix=".">
    <group delimiter=" " prefix="[" suffix="]" >
      <text macro="surname"/>
      <text macro="year"/>
    </group>
    <group delimiter=" ">
      <names variable="author" delimiter=", " suffix=":"/>
      <text variable="title" prefix=""" suffix="'"."/>
      <text variable="container-title" prefix="In: " font-style="italic"/>
      <text macro="issue" suffix=","/>
      <text variable="page" prefix="p. "/>
      <text variable="edition" suffix=" edition."/>
      <text macro="published"/>
    </group>
  </layout>
</bibliography>
</style>

```

record types of different natures, such as “Second”, “second”, “2nd”, “2”, etc. Therefore, if a CSL style needs to be robust, and requires an exact format for edition information, type queries and conversions routines may be needed especially for this field. For this purpose, CSL provides a number of different tests for complex, conditional processing of data fields; for example, `is-numeric` returns a (Boolean) “false” if a variable has the form “Second”, “second”, etc.

### 9.8 Publication details

The rendering of the publication details of books can be implemented like this:

```

<macro name="published">
  <choose>
    <if variable="publisher">
      <group delimiter=" ">
        <text variable="publisher-place" suffix=":"/>
        <text variable="publisher"/>
        <text macro="year"/>
      </group>
    </if>
  </choose>
</macro>

```

This macro first checks whether the variable `publisher` is defined (which is not the case with `@Article`), and, if this is true, renders it together with `publisher-place` (which adopts the `BIBTEX` field `address`) and again the macro `year` in the desired way for this citation

style. This macro could be employed similarly to the others, with:

```
<text macro="published"/>
```

### 9.9 Sorting key

With an author–date citation style like this it’s useful to install a sort order, or the records are going to appear in the order of occurrence of the corresponding cites, which is typically not wanted. The following sort key puts the entries of the bibliography into the alphabetical order of the author’s surnames:

```

<sort>
  <key macro="surname"/>
</sort>

```

## 10 Result

With these features and the closing `</style>` root element, the very basic citation style which we intended to implement is completed. Like the others, this CSL style could be used to produce complete formatted citations out of the example `BIBTEX` data.

The `LATEX` formatted Pandoc output of the example references looks like this (with some editorial line breaks):

```
{[]Flom 2007{[]} Peter Flom: ‘‘LaTeX for academics and researchers who (think they) don’t need it’’. In: \emph{TUGboat} 28,1 (2007), p. 126--128.
```

```
{[]Kopka \& Daly 2004{[]} Helmut Kopka,
```

Patrick W. Daly: ‘‘A Guide to LaTeX and Electronic Publishing’’. Fourth edition. Boston: Addison-Wesley 2004.

{[]Sharma 2014[]} Tushar Sharma: ‘‘Why I never close Emacs’’. In: \emph{Open Source For You} 1/2014, p. 53--55.

To be sure, what has been set up here is far from robust and is just for demonstration purposes. The experienced bibliography writer knows that even with only the basic publication types which have been discussed, plenty of open questions remain which would go beyond our scope here. A more refined style would need additional features such as book titles set in italics, using the prefix ‘‘pp.’’ for page ranges, using ‘‘et al.’’ for multiple authors if required by the style, etc. These topics and several others are planned to be the subject of a follow-up article.

In general, CSL offers features for every last detail of bibliographical typesetting; the styles which are actually used in production are much more complex than what has been demonstrated here.

## 11 Conclusion

CSL provides a sophisticated and versatile tool (e.g. it also supports localization) for the programming of citation styles. It has already become widespread, for good reason.

In my opinion, CSL responds to the natural complexity of the subject ‘‘citation’’ with a very elegant, intuitive and simple XML-based user interface. This distinguishes CSL from the, for example, difficult-to-penetrate stack-based BIBTEX language for .bst styles [6].

Although a CSL preprocessor for LATEX, to the best of my knowledge, still remains a desideratum, it is still highly recommended to become familiar with CSL when dealing with bibliographical typesetting. Finally, until a CSL capable replacement for the BIBTEX preprocessor becomes available, Pandoc’s LATEX output is useful.

## References

- [1] Massimiliano Dominici. An overview of pandoc. *TUGboat*, 35(1):44–50, 2014. URL: <http://tug.org/TUGboat/tb35-1/tb109dominici.pdf>.
- [2] Joe Fawcett, Liam R.E. Quin, and Danny Ayers. *Beginning XML*. John Wiley & Sons Inc., Indianapolis, fifth edition, 2012.
- [3] Axel Kielhorn. Multi-target publishing. *TUGboat*, 32(3):272–277, 2011. URL: <http://tug.org/TUGboat/tb32-3/tb102kielhorn.pdf>.
- [4] Philipp Lehman, Philip Kime, Audrey Boruvka, and Joseph Wright. The BIBLATEX package: Programmable bibliographies and citations. version 2.9a. 24/06/2014, 2014. URL: <http://ctan.org/pkg/biblatex>.
- [5] Frank Mittelbach, Michel Goossens, et al. *The LATEX Companion*. Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, second edition, 2004.
- [6] Oren Patashnik. Designing BIBTEX styles, 1988. URL: <http://mirror.ctan.org/biblio/bibtex/base/btxhak.pdf>.
- [7] Michael Shell and David Hoadley. BIBTEX tips and FAQ. version 1.1, 2007. URL: <http://mirror.ctan.org/biblio/bibtex/contrib/doc/btxFAQ.pdf>.
- [8] University of Chicago Press staff, editor. *The Chicago Manual of Style*. University of Chicago Press, Chicago, Ill., sixteenth edition, 2010.
- [9] Rintze M. Zelle. Citation style language 1.0: Primer, 2011. URL: <http://citationstyles.org/downloads/primer.html>.
- [10] Rintze M. Zelle, Frank G. Bennet, Jr., and Bruce D’Arcus. Citation style language 1.0.1: Language specification, 2012. URL: <http://citationstyles.org/downloads/specification.html>.

◇ Daniel Stender  
Hamburg, Germany  
daniel (at) danielstender.com  
<http://www.danielstender.com/>



## The Treasure Chest

This is a list of selected new packages posted to CTAN (<http://ctan.org>) from March through September 2014, with descriptions based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. 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'd especially like to point out the welcome proliferation of font packages. A wide variety of fonts are available to (L<sup>A</sup>)T<sub>E</sub>X users nowadays, almost all usable with any T<sub>E</sub>X engine. We recommend the DK-TUG Font Catalogue (<http://www.tug.dk/FontCatalogue>) for exploration.

We hope this column and its companions will help to make CTAN a more accessible resource to the T<sub>E</sub>X community. Comments are welcome, as always.

◇ Karl Berry  
<http://tug.org/ctan.html>

---

### fonts

- `almfixed` in **fonts**  
Arabic Unicode extending Latin Modern Mono.
- \*`baskervaldx` in **fonts**  
Greatly extended and modified BaskervaldADF.
- `caladea` in **fonts**  
Caladea fonts.
- `calibri` in **fonts**  
Carlito sans serif fonts.
- `cinzel` in **fonts**  
Cinzel and Cinzel Decorative fonts.
- `clearsans` in **fonts**  
Clear Sans fonts.
- `dantelogo` in **fonts**  
Using the DANTE e.V. logo.
- `drm` in **fonts**  
Revised modern meta-font.
- `ebgaramond-maths` in **fonts**  
L<sup>A</sup>T<sub>E</sub>X support for using EBGaramond in math.
- `erewhon` in **fonts**  
Extends Heuristica which extends Utopia.
- `fira` in **fonts**  
Fira fonts, designed for Firefox.
- `heuristica` in **fonts**  
Heuristica fonts, extending Utopia with Cyrillic.
- `newtxtt` in **fonts**  
Enhancement of typewriter fonts from `newtx`.
- `obnov` in **fonts**  
Obyknovennaya Novaya Cyrillic font.  
(See article in this issue.)

- `parisa` in **fonts**  
Persian fonts derived from FarsiT<sub>E</sub>X et al.
- `playfair` in **fonts**  
Playfair Display fonts.
- `ptmsc` in **fonts**  
Use proprietary Adobe TimesSC with `newtx`.
- `roboto` in **fonts**  
Roboto fonts.
- \*`universalis` in **fonts**  
Universalis fonts, alternatives to Univers and Frutiger.

---

### graphics

- `asypictureb` in **graphics**  
User-friendly integration of Asymptote into L<sup>A</sup>T<sub>E</sub>X.
- `blox` in **graphics/pgf/contrib**  
Draw block diagrams.
- `dspticks` in **graphics/pstricks/contrib**  
Digital signal processing plots.
- `interactiveplot` in **graphics**  
Creating interactive 2D/3D functions inside a PDF.
- `pst-spirograph` in **graphics/pstricks/contrib**  
Simulate operation of a spirograph.
- `qcircuit` in **graphics**  
Macros to generate quantum circuits.

---

### info

- `latexsource-ng` in **info**  
Introduction to L<sup>A</sup>T<sub>E</sub>X, with setup information.

---

### language

- `dad` in **language/arabic**  
Typesetting Arabic and mixed Arabic/Latin.  
(See article in this issue.)

---

### macros/generic

- `bagpipe` in **macros/generic**  
Typesetting bagpipe music.
- `docbytex` in **macros/generic**  
Creating documentation from source code.
- `lpform` in **macros/generic**  
Linear programming formulations.
- `tracklang` in **macros/generic**  
Determining user-requested languages.

---

### macros/latex/contrib

- `afparticle` in **macros/latex/contrib**  
Typeset articles for the open access journal *Archives of Forensic Psychology*.
- `assocnt` in **macros/latex/contrib**  
Advancing many counters simultaneously.
- `bangorcsthesis` in **macros/latex/contrib**  
Thesis class for Bangor University.
- `bnumexpr` in **macros/latex/contrib**  
Extends eT<sub>E</sub>X's `\numexpr` to big integers.
- `clrscode3e` in **macros/latex/contrib**  
Typeset pseudo-code as in *Introduction to Algorithms*.

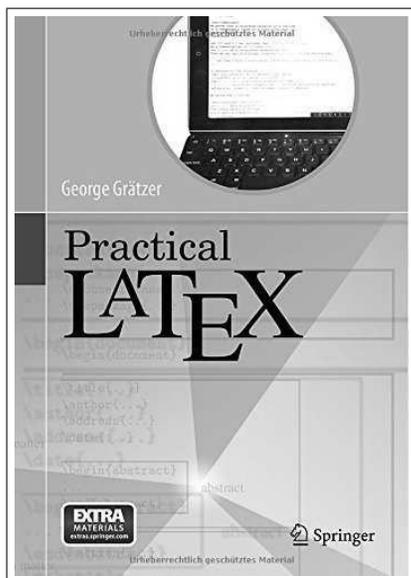
`macros/latex/contrib/clrscode3e`

- `dithesis` in `macros/latex/contrib`  
Undergraduate theses at the University of Athens.
- `doctools` in `macros/latex/contrib`  
Tools for documentation of L<sup>A</sup>T<sub>E</sub>X code.
- `efbox` in `macros/latex/contrib`  
Enhanced inline box with optional frames and colors.
- `environ` in `macros/latex/contrib`  
New interface for L<sup>A</sup>T<sub>E</sub>X environments.
- `fifo-stack` in `macros/latex/contrib`  
FIFO and stack implementations.
- `fullminipage` in `macros/latex/contrib`  
Minipage spanning a complete page.
- `getmap` in `macros/latex/contrib`  
Downloading OpenStreetMap maps.
- `gitinfo2` in `macros/latex/contrib`  
Use metadata from git repositories in L<sup>A</sup>T<sub>E</sub>X.
- `graphbox` in `macros/latex/contrib`  
Provide more options for placement of graphics.
- `grundgesetze` in `macros/latex/contrib`  
Typeset Frege's *Grundgesetze der Arithmetik*.
- `handout` in `macros/latex/contrib`  
Handout for audiences at a talk.
- `komacv` in `macros/latex/contrib`  
Typeset CV with various style options.
- \*`l3build` in `macros/latex/contrib`  
Test and build system for (L<sup>A</sup>)T<sub>E</sub>X.
- `labyrinth` in `macros/latex/contrib`  
Drawing labyrinths and related.
- `lastpackage` in `macros/latex/contrib`  
Defines last point where packages can be loaded.
- \*`latexdemo` in `macros/latex/contrib`  
Demonstrate L<sup>A</sup>T<sub>E</sub>X code with resulting output.
- `logicproof` in `macros/latex/contrib`  
Box proofs for propositional and predicate logic.
- `longfigure` in `macros/latex/contrib`  
Figure-like environment that breaks over pages.
- `matlab-prettyfier` in `macros/latex/contrib`  
Pretty-print Matlab source code.
- `mugsthesis` in `macros/latex/contrib`  
Marquette University Graduate School theses.
- `listlbls` in `macros/latex/contrib`  
List of all labels used in a document.
- `pressrelease` in `macros/latex/contrib`  
Class for typesetting press releases.
- `pygmentex` in `macros/latex/contrib`  
Typeset code listings using Pygments.
- `qrcode` in `macros/latex/contrib`  
Generate QR codes.
- `repltext` in `macros/latex/contrib`  
Control text copied from a PDF.
- `sclang-prettyfier` in `macros/latex/contrib`  
Pretty-print SuperCollider source code.
- `sphdthesis` in `macros/latex/contrib`  
Theses at National University of Singapore.
- `sympytexpackage` in `macros/latex/contrib`  
Support for sympy (Symbolic Python) expressions.
- `tablestyles` in `macros/latex/contrib`  
Separation of text and style in tables.
- `templatetools` in `macros/latex/contrib`  
Conditionals helpful in templates.
- `testhyphens` in `macros/latex/contrib`  
Testing hyphenation patterns.
- `tudscr` in `macros/latex/contrib`  
Technische Universität Dresden documents.
- `ucbthesis` in `macros/latex/contrib`  
UC Berkeley thesis class, based on `memoir`.
- `yathesis` in `macros/latex/contrib`  
Writing a thesis following French rules.
- 
- macros/latex/contrib/babel-contrib**
- `latvian` in `m/l/c/babel-contrib`  
Babel support for Latvian.
- 
- macros/latex/contrib/beamer-contrib**
- `themes/beamerdarkthemes` in `m/l/c/beamer-contrib`  
Bundle of dark color (black background) themes.
- 
- macros/latex/contrib/biblatex-contrib**
- `biblatex-anonymous` in `m/l/c/b-c`  
Managing anonymous works.
- `biblatex-bookinarticle` in `m/l/c/b-c`  
New entry type `@bookinarticle`.
- `biblatex-multiple-dm` in `m/l/c/b-c`  
Load multiple datamodels in `biblatex`.
- `biblatex-realauthor` in `m/l/c/b-c`  
Indicate real author of a work.
- `biblatex-true-citepages-omit` in `m/l/c/b-c`  
Avoid limitations of standard `citepages=omit` option.
- 
- macros/luatex**
- `luatodonotes` in `macros/luatex/latex`  
Add editing annotations in margins.
- `placeat` in `macros/luatex/latex`  
Absolute content positioning for LuaL<sup>A</sup>T<sub>E</sub>X.
- 
- macros/xetex**
- `bidi-atbegshi` in `macros/xetex/latex`  
Bidi-aware shipout macros.
- `bidicontour` in `macros/xetex/latex`  
Bidi-aware version of `contour`.
- `bidipagegrid` in `macros/xetex/latex`  
Bidi-aware version of `pagegrid`.
- `bidipresentation` in `macros/xetex/latex`  
Bidi-aware presentations.
- `bidishadowtext` in `macros/xetex/latex`  
Typesetting bidi-aware shadow text.
- 
- support**
- `texlive-dummy` in `support`  
Dummy RPM to satisfy package requirements.
- 
- systems**
- `hktex` in `systems/android`  
T<sub>E</sub>Xish formula parsing software for Android.

**Book review: *Practical L<sup>A</sup>T<sub>E</sub>X*,  
by George Grätzer**

William Adams

George Grätzer, *Practical L<sup>A</sup>T<sub>E</sub>X*, [http://www.springer.com/new+%26+forthcoming+titles+\(default\)/book/978-3-319-06424-6](http://www.springer.com/new+%26+forthcoming+titles+(default)/book/978-3-319-06424-6). Paperback, 216 pp., Springer, 2014.



For those interested, Grätzer is also the author of *Math Into L<sup>A</sup>T<sub>E</sub>X*, *More Math Into L<sup>A</sup>T<sub>E</sub>X*, several other L<sup>A</sup>T<sub>E</sub>X books, and many non-L<sup>A</sup>T<sub>E</sub>X books also.

*Practical L<sup>A</sup>T<sub>E</sub>X* is a slim, well-named volume, since it is eminently practical. It is an excellent introductory text, covering contemporary markup, macros and packages, eschewing obsolete material. Covering the essentials of document production including niceties such as B<sup>I</sup>B<sup>T</sup>E<sub>X</sub> and T<sub>i</sub>k<sub>Z</sub>, it is an excellent, up-to-date introduction which one can hand to any potential user with confidence that it will help them to rapidly achieve a basic proficiency allowing for the production of documents with an efficiency which would be the envy of other tools.

Nicely designed and typeset, it covers document elements including text, environments, formulas, illustrations, symbols and bibliographies as well as

presentations and the customization of L<sup>A</sup>T<sub>E</sub>X. The coverage of illustrations is especially nice, including both the basics of placing image files using the `graphicx` package and creating illustrations using the code-oriented tool T<sub>i</sub>k<sub>Z</sub> (this chapter is based on Jacques Crémer’s publicly available *A very minimal introduction to T<sub>i</sub>k<sub>Z</sub>*).

There is an excellent index as well as several very useful appendices for symbols and commands which also make the book a useful quick reference.

If the book has a flaw, it is the rather inexplicable appendix “L<sup>A</sup>T<sub>E</sub>X on the iPad” which unnecessarily limits itself to a specific platform and swerves into a political screed which at once complains that the GNU Public License (GPL) “stops you from having it used on the fastest growing platform of all time” while at the same time discussing several different L<sup>A</sup>T<sub>E</sub>X apps for the iPad which are capable of typesetting L<sup>A</sup>T<sub>E</sub>X documents.

As an historical footnote and commentary: Duncan P. Steele of Valletta Ventures had initially authored a blog post complaining of the L<sup>A</sup>T<sub>E</sub>X codebase and noting that the GPL licensing of the mainstream T<sub>E</sub>X distributions made producing a T<sub>E</sub>X editor for the iPad impossible due to the interactions of Apple’s licensing and the GPL, <http://vallettaventures.com/2011/12/10/messy-latex/>. However, after being informed of the existence of KerT<sub>E</sub>X (<http://www.kergis.com/en/kertex.html>), which is available under a permissive license, and being convinced that T<sub>E</sub>X itself was available in the public domain, Steele was able to produce a version of T<sub>E</sub>Xpad which runs on the iPad: <http://vallettaventures.com/2012/09/07/texpad-ipad-v1.1/>.

Highly recommended for beginners, in particular those who might wish to make use of L<sup>A</sup>T<sub>E</sub>X on their iPad, as well as the occasional user who needs a reference. An updated version which eschewed the political commentary and included coverage of at least one of the many L<sup>A</sup>T<sub>E</sub>X editors for Android tablets would be especially welcome.

◇ William Adams  
608 Wayne Drive  
Mechanicsburg, PA  
willadams (at) aol dot com

**Book review: *Apprendre à programmer en T<sub>E</sub>X*, by Christian Tellechea**

Jacques André

Christian Tellechea, *Apprendre à programmer en T<sub>E</sub>X* (Learning programming in T<sub>E</sub>X), <http://www.lulu.com/us/en/shop/christian-tellechea/apprendre-%C3%A0-programmer-en-tex/paperback/product-21816783.html>. Paperback, 580 pp., Lulu, 2014. Revision of 21/9/2014.



This book is in French and let me say right out that it deserves an English translation.

There are many books on T<sub>E</sub>X, as a typesetting tool (e.g., see list at [1]). Very few are dedicated to T<sub>E</sub>X as a programming language. Not a functional or general-purpose language, rather the kind of “programming language with a documentation language, thereby making programs more robust, more portable, more easily maintained, and arguably more fun to write than programs that are written only in a high-level language” (preface of [2]).

Tellechea’s book could be seen as a rewrite of a subset of *The T<sub>E</sub>Xbook* [3]. However it avoids everything concerning, e.g., mathematical formula setting and focuses only on matters that are used today, e.g., to write (L<sup>A</sup>)T<sub>E</sub>X macros. Furthermore it uses a new framework, since it is a tutorial manual

(even if, in the end, it can also be considered a reference manual, since it offers complete coverage of its chosen topics).

In the first hundred or so pages, the author explains the very low-level concepts such as catcodes, commands, active characters, arguments, developments, expansions, ... At first glance, this part appears a bit verbose or slow, and you can’t see the forest for the trees. Nevertheless, this is probably the best way to make sure these concepts are fully understood by people not familiar with such a language. The odds are that even experienced macro users will learn something!

The second part describes numbers, recursion and control structures. Exercises allow the readers used to conventional programming to write macros to simulate their conventional structures such as *for...do...or various forms of if then else fi*.

Boxes, dimensions and input/output are studied in the following chapters and exercises answer everyday needs (lists, stacks, grids, etc., not to mention curve drawing without waste of memory or time!), as well as classical algorithms.

A final part revisits all the material in the context of long and thoroughly-commented examples such as the layout of paragraphs.

This book is easy to read and progressive. It leaves no question unanswered, and the exercises are useful both to understand the underlying concepts and to be reused in our programs.

As I said at first: a most worthwhile book to be translated into English.

## References

- [1] *Books about T<sub>E</sub>X and Friends*, <http://tug.org/books>
- [2] Donald E. Knuth, *Literate Programming*, CSLI Lecture Notes, no. 27, Stanford, 1992.
- [3] Donald E. Knuth, *The T<sub>E</sub>Xbook*, Reading, Massachusetts: Addison-Wesley, 1984.

◇ Jacques André  
 jacques dot andre35 (at) gmail  
 dot com  
<http://jacques-andre.fr>

**Book review: *The Imitation Game*,  
by Jim Ottaviani & Leland Purvis**

Michael S. Berry

Jim Ottaviani & Leland Purvis, *The Imitation Game*. <http://www.tor.com/stories/2014/06/the-imitation-game-jim-ottaviani-leland-purvis>, 2014.



*From the editors.* While Alan Turing of course died long before  $\text{\TeX}$  existed, there are several reasons why we think his biography as a graphic novel might be interesting for *TUGboat* readers. First, many of us are involved with computer science, and Turing was one of the founding fathers of this fascinating field. Second, both DEK and (more recently) Leslie Lamport are recipients of the Turing Award — the highest distinction for computer scientists. Third, many *TUGboat* readers are interested in modern book design, and therefore might appreciate an online graphic novel. Last but not least, this is an interesting and unusual book.

— \* —

Alan Turing was a young man in search of himself as well as universal truths, or so he is cast by Ottaviani and Purvis in *The Imitation Game*. Socially awkward and eccentric, he nonetheless managed to gain success through a combination of innate genius and happenstance, the latter due to Great Britain's involvement in World War II. The authors take us through Turing's life from childhood, advanced mathematical and philosophical education, his lead role in breaking the German Enigma code, subsequent lack of recognition for theoretical accomplishments, and, finally, his persecution and premature death. The story is told by Turing, himself, his mother and a host of associates from various periods of his life. The dialog is, of course, conjectural reconstruction, but the authors do a convincing job of portraying the "essential" Turing and, for the most part, the narrative flows well.

The novel is decidedly not a primer on the Turing Machine or its underlying philosophical issues. It traverses critical points in the evolution of Turing's thought processes in a manner assuming a fairly sophisticated knowledge on the part of the reader, knowledge this reviewer did not possess. The salutary effect (in my case and I suspect the same will be

true for other readers) is to stimulate sufficient curiosity to research the critical issues. The Internet is replete with references to the *entscheidungsproblem* (Church–Turing thesis), working code for various iterations of Turing Machines and detailed analyses of the Enigma code and its decipherment (some heavy lifting involved but it fleshes out an understanding of the novel).

As noted earlier, Turing was not fully appreciated in his time beyond a small circle of colleagues. His brilliant, leading effort to break the Enigma code was largely masked due to national security issues. Then there was the familiar academic practice of extensive "borrowing" (some would say plagiarism) of ideas on the part of established, senior scholars (e.g. von Neumann) at the expense of lesser known contributors. But such slights paled in comparison to the tragic events of his final years. The authors walk us through Turing's arrest and trial for homosexuality (a fact he did not challenge), the subsequent conviction and his untimely death (the authors accept the designation of suicide whereas others opt for accidental ingestion of cyanide). I believe that the authors have here missed an opportunity to place these events in broader context so as to clearly convey the ludicrous irony involved.

The issue involved the academic imprimatur afforded eugenics at the time. Led by Charles Davenport in the United States, positive (encouraging reproduction by superior lineages) and negative (elimination of inferior lineages) eugenics was a widely accepted, if terribly wrongheaded, derivation of Darwinian natural selection. Nazi Germany appropriated Davenport's eugenics theory in toto and extended it to its logical ends: the enormity of genocide and the holocaust. Turing was awarded the OBE (Most Excellent Order of the British Empire) for his role in defeating Hitler and the Nazi regime. Yet he was convicted and chemically castrated for a crime, the rationale of which was based in the assumed validity of the theory of eugenics (i.e., the elimination of the "unfit" from the gene pool) and suffered the consequent physical impairment and ignominy. It should not be lost to history, or the readers of this novel, that Winston Churchill, who lauded Turing's World War II contributions, was a vocal advocate of eugenics for the improvement of the British people.

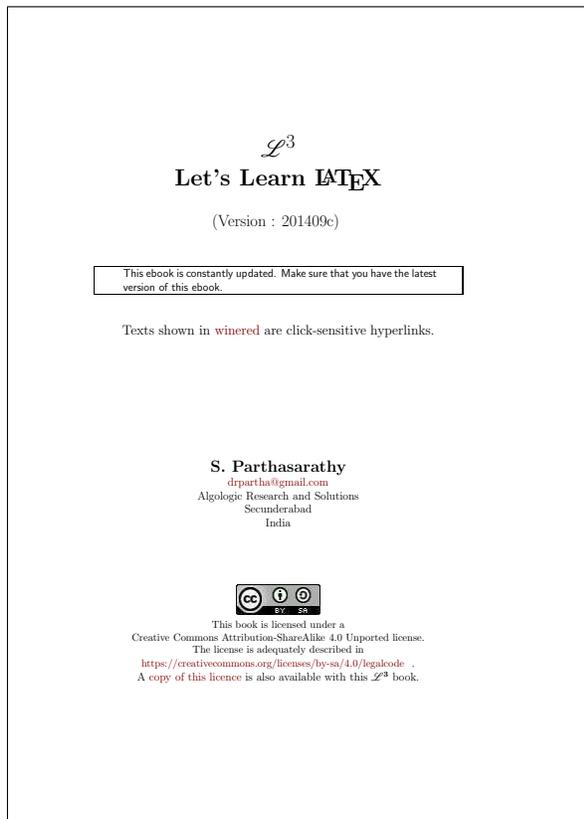
In sum, with the above exception noted, I found the novel both illuminating for those unfamiliar with Turing's too-brief life history and stimulative of additional research for those interested in the more arcane aspects addressed. A good read.

◇ Michael S. Berry  
Dominguez Archaeological Research Group  
msberry49 (at) gmail dot com

**Book review: *Let's Learn L<sup>A</sup>T<sub>E</sub>X*,  
by S. Parthasarathy**

Nicola L. C. Talbot

S. Parthasarathy, *Let's Learn L<sup>A</sup>T<sub>E</sub>X*, <http://www.freewebs.com/profpartha/teachlatex.htm>, 2014. 24 pp., free ebook. Version 201408e.



This is a free ebook licensed under the Creative Commons Attribution-ShareAlike 4.0 Unported license. The preface starts with a tribute to Richard Stallman and brief information about FOSS (free and open source software—the acronym could do with an expansion in the book). The preface then states that the purpose of the book is to encourage hacking as a method of learning L<sup>A</sup>T<sub>E</sub>X. The book, rather than being a reference text containing instructions and definitions, provides a list of example documents that the reader can try out and modify as a learning tool.

The idea of learning by hacking example code is a useful concept, and one that I often employ when investigating a new programming language. However, I'm concerned that the sample documents provided with this book use obsolete code and deprecated practices. Some illustrations follow below.

Most of the sample documents use the obsolete L<sup>A</sup>T<sub>E</sub>X 2.09 font commands, such as `\bf`. These are

deprecated in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and should be avoided [5]. There are also some instances where a font changing declaration, such as `\Huge`, is followed by an unnecessary group, which may confuse the reader into thinking the command requires an argument. For example, on line 50 of `certif0.tex` there are two sets of redundant braces in

```
{\bf{\Huge{'\LaTeX\ hands-on'}}}
```

since neither `\bf` nor `\Huge` have an argument. Only the outermost set of braces scope the effects of the declarations.

Many of the sample documents load the `epsfig` package [7]. The original `epsfig` style that was provided with L<sup>A</sup>T<sub>E</sub>X 2.09 is now obsolete and should not be used. Current T<sub>E</sub>X distributions provide a newer `epsfig` package that is just a wrapper package that loads the `graphicx` package [2]. The recommended practice is to use `graphicx` directly and not specify the image file extensions [4]. Incidentally, there is also no longer any need to specify the `dvips` package option, which occurs in many of the sample documents. The only time a driver option is needed is in the cases where it can't be determined, such as `dvipdfm` [10]. Omitting the driver and the file extensions helps to make the document more portable.

Curiously, some of the documents, such as the file `spacing.tex` load both `epsfig` and `graphicx`, which is redundant. There are other instances of unnecessary repetition where there are multiple attempts to load the same package. For example, in the file `torture1.tex` not only are both `epsfig` and `graphicx` loaded on line 5, but there is also an attempt to load `epsfig` on line 11 and `graphicx` on line 13. Similarly, there are two attempts to load `amssymb` (on lines 4 and 10), `amsmath` (on lines 4 and 8) and `amsfonts` (on lines 4 and 9). Removing this duplication would provide a more streamlined example.

On the subject of images, the image files aren't actually provided with the sample documents for copyright and licensing issues, so I think it would be useful if the author could mention the use of the `demo` option provided by the `graphicx` package, which would enable the documents to be compiled without error. Alternatively, perhaps mention the image files provided with the `mwe` package [9].

There is prolific use of `\` within paragraphs in the sample documents, which is generally best avoided [1]. I think using paragraph breaks instead of `\` would be more appropriate in most of these cases, and blank lines would additionally help readability of the code.

More surprising is the use of `\` immediately before paragraph breaks. The  $\LaTeX$  user guide [6, p. 213] warns against this as it produces underfull `\hbox` warnings and extra vertical space. (If vertical spacing is required between paragraphs, there are more appropriate methods of achieving this [3].)

I was somewhat bemused by the line `%\documentstyle[epsfig, picinpar, 12pt]{article}` in the file `latexography.tex`. Even though the line is commented, `\documentstyle` should not appear in any modern  $\LaTeX$  tutorial, except where it is being pointed out as obsolete. There's a danger here that new curious users may uncomment the line and try it out.

I was interested to see that the sample document `kuralengtam3a.tex` loads the `fontspec` package [8], which is a  $\XeTeX$  and  $\LuaTeX$  package. It's not often that a  $\LaTeX$  tutorial uses a different engine. I think this is a good idea, but it would help if the author pointed out to the reader in a comment at the start of the file that  $\XeTeX$  or  $\LuaTeX$  is required, otherwise users eager to compile the sample document before reading it may not realise they need a different engine.

This sample document requires the Arial font. As a GNU/Linux user, I don't have any commercial fonts installed, and it wasn't immediately obvious at what point the document was switching to Arial, but the command-line invocation of `grep Arial *` tracked it down to eight of the accompanying files. After I had replaced all instances of `Arial` with `Liberation Sans` in those files, I was able to get the document to compile without error. Windows users won't have this problem, but I was puzzled by the author's choice of Arial rather than a free font given the book's FOSS ethos.

If the author updates the sample documents so that the redundancy, obsolescence and deprecated practices are removed, this book could be a useful tool in learning  $\LaTeX$  and introducing the reader to  $\XeTeX$ .

## References

- [1] David Carlisle. Answer to: When to use `\par` and when `\`. TeX–LaTeX Stack Exchange, 2012. URL: <http://tex.stackexchange.com/a/82666> (version: 2012-11-14).
- [2] David Carlisle. The `graphicx` package, April 2014. Available from CTAN, `macros/latex/required/graphics` (version: 1.0g, 2014-04-25).
- [3] TeX FAQ. Zero paragraph indent. URL: <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=parskip> (version: 3.28, 2014-06-10).
- [4] TeX FAQ. Portable imported graphics, 2014. URL: <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=graph-pspdf> (version: 3.28, 2014-06-10).
- [5] TeX FAQ. What's wrong with `\bf`, `\it`, etc.?, June 2014. URL: <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=2letterfontcmd> (version: 3.28, 2014-06-10).
- [6] Leslie Lamport. *LaTeX: A document preparation system*. Addison-Wesley, 1994.
- [7] Sebastian Rahtz and David Carlisle. The `epsfig` package, February 1999. Available from CTAN, `macros/latex/required/graphics` (version: 1.7a, 1999-02-16).
- [8] Will Robertson and Khaled Hosny. The `fontspec` package: Font selection for  $\XeTeX$  and  $\LuaTeX$ , June 2014. Available from CTAN, `macros/latex/contrib/fontspec` (version: 2.4a, 2014-06-21).
- [9] Martin Scharrer. The `mwe` package, May 2012. Available from CTAN, `macros/latex/contrib/mwe` (version: 0.3, 2012-05-15).
- [10] Joseph Wright. Answer to: Driver specification for hyperref and graphicx. TeX–LaTeX Stack Exchange, 2010. URL: <http://tex.stackexchange.com/a/6949> (version: 2010-12-12).
  - ◊ Nicola L. C. Talbot  
School of Computing Sciences  
University of East Anglia  
Norwich Research Park  
Norwich  
NR4 7TJ  
United Kingdom  
N.Talbot (at) uea dot ac dot uk  
<http://theoval.cmp.uea.ac.uk/~nlct/>

---

### Die $\TeX$ nische Komödie 2–3/2014

*Die  $\TeX$ nische Komödie* is the journal of DANTE e.V., the German-language  $\TeX$  user group (<http://www.dante.de>). [Non-technical items are omitted.]

#### Die $\TeX$ nische Komödie 2/2014

ANDREAS ENTENMANN and WALTER ENTENMANN,  
Zum Entwurf von Postern [On the creation of posters];  
pp. 37–51

Thanks to the `aoposter` package by Kettl and Weiser one can create posters for scientific meetings using  $\LaTeX$ 's standard formatting commands. Conventions of corporate design can be incorporated without problems, if the logo files and colors are available. After a short introduction to the `aoposter` class this article provides some general insights on the design of posters for scientific conferences. We also present some little tricks to, e.g. create A4 testprints, to convert the output format or to slice the poster into printable pieces. Based on a specific example the different steps are described and bundled in a package.

DOMINIK WAGENFÜHR, Registerhaltiger Satz mit  $\LaTeX$  [Grid typesetting with  $\LaTeX$ ]; pp. 52–64

If one looks at a modern newspaper, one will likely see that in multi-column typesetting the adjacent lines are always on the same height. This property is called grid typesetting. While  $\LaTeX$  does not offer this functionality out-of-the-box one may achieve good results with some manual interventions.

ULRIKE FISCHER,  $\text{BIB}\LaTeX$ -Variationen [BIB $\LaTeX$  variations]; pp. 65–75

[Translation published in this issue of *TUGboat*.]

UWE ZIEGENHAGEN, Spendenbescheinigungen [Creating donation receipts using  $\LaTeX$ , SQL and Python]; pp. 76–82

In my capacity as treasurer for the Cologne-based Dingfabrik “fab lab” one of my tasks is to create the annual donation receipts for all donors. The process in place until recently involved manual aggregation in Excel and manual creation of the receipts in MS Word, not a desirable way to go for a  $\TeX$ ie. This article describes how the forms were created from scratch with  $\LaTeX$  and filled using an intelligent combination of Quicken, Excel, MySQL and Python.

AXEL KIELHORN, Präsentationen mit Beamer [Presentations with Beamer]; pp. 83–93

The `beamer` document class offers an easy way to create presentations. Due to the numerous options and templates, getting started with Beamer may not seem that easy. A presentation example shows many of the available options and presents some of the challenges (and their solutions) a new Beamer user might face.

AXEL KIELHORN,  $\LaTeX$  für Nichtlateiner [LaTeX for non-Latinates]; pp. 94–98

For historical reasons, different operating systems use different character encodings. Windows uses CP-1252,

Mac OS X MacRoman, the various Unix derivatives offer HPRoman8, CP-850 and ISO Latin 1, among others. But these are only sufficient for Western Europe and parts of the Americas; for central Europe one needs additional encodings.

GÜNTER PARTOSCH, Anforderungen an wissenschaftliche Abschlussarbeiten [Requirements for scientific theses]; pp. 94–98

The usual way to finish a course of studies in Germany is to write a thesis. Form, length and other parameters are usually defined not just by the university but also by the thesis supervisor. Additional requirements are introduced when the thesis is to fulfill good scientific work or to be published on the Internet. In this article it is shown how  $\LaTeX$  can be successfully applied.

#### Die $\TeX$ nische Komödie 3/2014

JACOB WIERSMA, Mehr Möglichkeiten mit Fußnoten [More options with footnotes]; pp. 6–13

For some time there have been packages that extend the limits of  $\LaTeX$ 's standard footnote algorithms. This article presents the `bigfoot`, `manyfoot` and `footmisc` packages and discusses a few suggestions for improvements.

STEVE ZAKRZOWSKY, Paket `skmath` für mathematische Formeln [The `skmath` package for mathematical formulas]; pp. 14–18

The `skmath` package was developed by Simon Sigurdhsson, who has also created a few special document classes. Writing mathematical expressions and equations can easily make documents confusing. The `skmath` package offers some extensions for the simple and intuitive entry of mathematical expressions.

IDRIS SAMAWI HAMID, DANTE summary report: Introducing Arabic-Latin Modern Fixed; pp. 19–46

The Oriental  $\TeX$  project was initiated in 2006 to facilitate the development of high quality typography and typesetting of academic and scholarly texts that require the Arabic script, such as critical editions and monographs. Although support for the Arabic script in modern typesetting software has been slowly improving over the past decade or so, the situation is still very far behind the Latin script in terms of features, available high-quality typefaces, and layout-processing software. For academic and scholarly work, it's still very much a wilderness out there. A full solution to the problems of advanced Arabic-script typography and typesetting, particularly one based on OpenType and Unicode standards, is still some ways off.

CHRISTINE RÖMER, Mit `etoc` Inhaltsverzeichnisse anpassen [Adjusting tables of contents with `etoc`]; pp. 47–54

The new `etoc` package extends  $\LaTeX$ 's capabilities to create individual tables of contents. It is especially useful to create local tables of contents.

[Received from Herbert Voß.]

---

### **Les Cahiers GUTenberg issue 57 (2012)**

*Les Cahiers GUTenberg* is the journal of GUTenberg, the French-language T<sub>E</sub>X user group ([www.gutenberg.eu.org](http://www.gutenberg.eu.org)).

THIERRY BOUCHE, Éditorial; pp. 3–4

CHARLES BIGELOW, Histoire d'O, d'o et de 0 [Oh, oh, zero!]; pp. 5–53

Published in *TUGboat* 34:2.

LA LISTE TYPOGRAPHIE, Microtypographie digitale [Digital microtypography]; pp. 55–63

Since its inception, the French TYPOGRAPHIE mailing list has always devoted a large part of its discussions to microtypography, with special emphasis on non-alphabetic glyphs and complex constructs. It is thus no surprise that the design of digits has been regularly discussed, as well as the problem of having them cohabit with letters within typeset pages. Here, we extract from the list archives ([sympa.inria.fr/sympa/arc/typographie](http://sympa.inria.fr/sympa/arc/typographie)) three discussion threads dealing with issues such as the shape or the width of digits, especially oldstyle figures.

[Received from Thierry Bouche.]

---

### **T<sub>E</sub>X Development Fund 2013–2014 report**

T<sub>E</sub>X Development Fund committee

#### **MetaPost 2: Numerical engines**

Applicant: Taco Hoekwater, The Netherlands,

<http://tug.org/metapost>.

Amount: US\$2000; acceptance date: 2 Dec 2009 (completed 24 May 2011).

Implement better numerical handling in MetaPost, among other enhancements. An article about the initial MetaPost 2 project goals, by Hans Hagen and Taco Hoekwater, was published in *TUGboat* 30:3. MetaPost 1.802, included in T<sub>E</sub>X Live 2013, has support for several numeric representations, for example via the `-numbersystem` option.

#### **Lineno and related updates**

Applicant: Uwe Lueck, Germany,

<http://www.ctan.org/pkg/lineno>.

Amount: US\$1000; acceptance date: 17 Sep 2011.

For updates to the complex `lineno` package, and related efforts, such as factoring out functionality into separate packages.

#### **X<sub>q</sub>T<sub>E</sub>X math and other updates**

Applicant: Khaled Hosny, Egypt,

<http://www.ctan.org/pkg/xetex>.

Amount: US\$4000; acceptance date: 24 Apr 2012 (completed 25 Jul 2013).

For updates to the X<sub>q</sub>T<sub>E</sub>X engine, especially relating to OpenType math typesetting, and including updates as needed to LuaT<sub>E</sub>X to keep the engines in sync. Several

important external libraries had been deprecated and needed to be replaced. Other areas of work include finding fonts and syncing `xdvipdfmx` with `dvipdfmx`, as well as handling general bug reports. A report on the completed work was given in *TUGboat* 34:2.

#### **Dynamic library support in LuaT<sub>E</sub>X**

Applicant: Luigi Scarso, Italy,

<http://www.luatex.org/swiglib.html>

Amount: US\$2000; acceptance date: 31 May 2013.

Support shared libraries in LuaT<sub>E</sub>X using SWIG (<http://www.swig.org>). Some libraries are already supported, e.g., `mysql` and `graphicsmagick`.

#### **Metaflop: METAFONT via the web**

Applicant: Marco Müller, Switzerland,

<http://www.metaflop.com>.

Amount: US\$1000; acceptance date: 20 Jun 2013

(completed 10 Aug 2014).

Enhance the Metaflop web application, which provides a graphical interface for adjusting Metafont parameters, with improvements to the underlying fonts, the preview mechanism, and the generation.

#### **T<sub>E</sub>X Live for Android**

Applicant: Clerk Ma, China,

<http://code.google.com/p/texlive-for-android>.

Amount: US\$2000; acceptance date: 26 Jun 2013.

Add a native editor and package manager GUI to the T<sub>E</sub>X Live for Android project. <http://tug.org/tug2013/abstracts/ma.txt> has more background.

#### **Project Fandol: Free Chinese fonts and Russian-style math fonts**

Applicants: Clerk Ma and Jie Su, China,

<http://code.google.com/p/fandol-font>.

Amount: US\$1000; acceptance date: 9 Aug 2013.

(Information below is from the applicants.) Most math books in China are produced by Founder Bookmaker. This system has used a set of Russian style math fonts for more than 30 years. These commercial fonts are designed with a unique encoding by Founder. And, these fonts cannot work in T<sub>E</sub>X or other programs.

We have a set of metal types which contain two Russian style fonts (serif and sans serif). By analyzing these metal types, we find Founder's fonts are derived from these fonts, and Founder only provided a serif version (we will provide these math fonts in both serif and sans serif). These metal types were imported from the U.S.S.R. in 1953.

We will trace the metal fonts to outlines (initially in EPS format). For more detailed adjusting, we will be using FontForge. Parts of our Chinese fonts are already processed in this workflow. For these Russian style fonts, we will also work in this way.

- ◇ T<sub>E</sub>X Development Fund committee  
<http://tug.org/tc/devfund>

---

## 2015 T<sub>E</sub>X Users Group election

Kaja Christiansen  
for the Elections Committee

The positions of TUG President and nine members of the Board of Directors will be open as of the 2015 Annual Meeting, which will be held in July 2015 in Darmstadt, Germany.

The current President, Steve Peter, has stated his intention to step down, and the current Vice-President, Jim Hefferon, has stated his intention to run for President.

The directors whose terms will expire in 2015: Barbara Beeton, Karl Berry, Susan DeMeritt, Michael Doob, Taco Hoekwater, Ross Moore, Cheryl Ponchin, Philip Taylor, and Boris Veytsman.

Continuing directors, with terms ending in 2017: Kaja Christiansen, Steve Grathwohl, Jim Hefferon, Klaus Hoppner, Arthur Reutenauer, David Walden.

The election to choose the new President and Board members will be held in Spring of 2015. Nominations for these openings are now invited.

The Bylaws provide that “Any member may be nominated for election to the office of TUG President/to the Board by submitting a nomination petition in accordance with the TUG Election Procedures. Election . . . shall be by written mail ballot of the entire membership, carried out in accordance with those same Procedures.” The term of President is two years.

The name of any member may be placed in nomination for election to one of the open offices by submission of a petition, signed by two other members in good standing, to the TUG office at least two weeks (14 days) prior to the mailing of ballots. (A candidate’s membership dues for 2015 will be expected to be paid by the nomination deadline.) The term of a member of the TUG Board is four years.

A nomination form follows this announcement; forms may also be obtained from the TUG office, or via <http://tug.org/election>.

Along with a nomination form, each candidate must supply a passport-size photograph, a short biography, and a statement of intent to be included with the ballot; the biography and statement of intent together may not exceed 400 words. The deadline for receipt of nomination forms and ballot information at the TUG office is **1 February 2015**. Forms may be submitted by FAX, or scanned and submitted by e-mail to [office@tug.org](mailto:office@tug.org).

Ballots will be mailed to all members within 30 days after the close of nominations. Marked ballots must be returned no more than six (6) weeks following the mailing; the exact dates will be noted on the ballots.

Ballots will be counted by a disinterested party not affiliated with the TUG organization. The results of the election should be available by early June, and will be announced in a future issue of *TUGboat* as well as through various T<sub>E</sub>X-related electronic lists.

## 2015 TUG Election — Nomination Form

Only TUG members whose dues have been paid for 2015 will be eligible to participate in the election. The signatures of two (2) members in good standing at the time they sign the nomination form are required in addition to that of the nominee. **Type or print** names clearly, using the name by which you are known to TUG. Names that cannot be identified from the TUG membership records will not be accepted as valid.

The undersigned TUG members propose the nomination of:

**Name of Nominee:** \_\_\_\_\_

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

for the position of (check one):

**TUG President**

**Member of the TUG Board of Directors**

for a term beginning with the 2015 Annual Meeting, **July 2015**.

1. \_\_\_\_\_  
(please print)

\_\_\_\_\_ (signature) \_\_\_\_\_ (date)

2. \_\_\_\_\_  
(please print)

\_\_\_\_\_ (signature) \_\_\_\_\_ (date)

Return this nomination form to the TUG office (forms submitted by FAX or scanned and submitted by e-mail will be accepted). Nomination forms and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received in the TUG office no later than **1 February 2015**.<sup>1</sup> It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will incomplete applications be accepted.

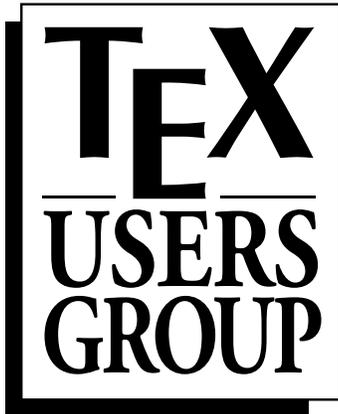
- nomination form
- photograph
- biography/personal statement

T<sub>E</sub>X Users Group **FAX:** +1 815 301-3568  
**Nominations for 2015 Election**  
P. O. Box 2311  
Portland, OR 97208-2311  
U.S.A.

---

<sup>1</sup> Supplementary material may be sent separately from the form, and supporting signatures need not all appear on the same form.

# TeX Users Group Membership Form 2015



*Promoting the use  
of TeX throughout  
the world.*

address:  
P.O. Box 2311  
Portland, OR 97208-2311 USA

phone: +1 503-223-9994  
fax: +1 815-301-3568  
email: office@tug.org  
web: http://www.tug.org

President Steve Peter  
Vice-President Jim Hefferon  
Treasurer Karl Berry  
Secretary Susan DeMeritt  
Executive Director Robin Laakso

TUG membership rates are listed below. Please check the appropriate boxes and mail the completed form with payment (in US dollars) to the mailing address at left. If paying by credit/debit card, you may alternatively fax the form to the number at left or join online at <http://tug.org/join.html>. The web page also provides more information than we have room for here.

Status (check one)  New member  Renewing member

Automatic membership renewal in future years

Will use given payment information; contact office to change/cancel.

	Rate	Amount
<input type="checkbox"/> <b>Early bird membership</b> for 2015 After March 31, dues are \$105.	\$85	_____
<input type="checkbox"/> <b>Special membership</b> for 2015 You may join at this special rate (\$75 after March 31) if you are a senior (62+), student, new graduate, or from a country with a modest economy. Please circle accordingly.	\$55	_____
If financially feasible for you, please consider checking here to donate \$30 (the difference from the regular membership rate). <input type="checkbox"/>	\$30	_____
<input type="checkbox"/> <b>Subscription</b> for 2015 (non-voting)	\$110	_____
<input type="checkbox"/> <b>Institutional membership</b> for 2015 Includes up to eight individual memberships and site-wide electronic access.	\$500	_____
<input type="checkbox"/> <b>Don't ship any physical benefits</b> (TUGboat, software) deduct \$20 TUGboat and software are available electronically.		_____
<b>Purchase last year's materials:</b>		
<input type="checkbox"/> TUGboat volume for 2014 (3 issues)	\$20	_____
<input type="checkbox"/> TeX Collection 2014 DVD with proTeXt, MacTeX, TeX Live, CTAN.	\$10	_____

**Voluntary donations** (more info at <https://www.tug.org/donate.html>)

- General TUG contribution \_\_\_\_\_
- Bursary Fund contribution \_\_\_\_\_
- TeX Development Fund contribution \_\_\_\_\_
- CTAN contribution \_\_\_\_\_
- LaTeX contribution \_\_\_\_\_
- LuaTeX contribution \_\_\_\_\_
- MacTeX contribution \_\_\_\_\_
- TeX Gyre fonts contribution \_\_\_\_\_

**Total \$** \_\_\_\_\_

*Tax deduction:* The membership fee less \$40 is generally deductible, at least in the US.  
*Multi-year orders:* To join for more than one year at this year's rate (up to ten years, non-refundable), please multiply by the number of years desired.

Payment (check one)  Payment enclosed  Visa  MasterCard  AmEx

Account Number: \_\_\_\_\_ Exp. date: \_\_\_\_\_

Signature: \_\_\_\_\_

**Privacy:** TUG uses your personal information only to send products, publications, notices, and (for voting members) official ballots. TUG does not sell or otherwise provide its membership list to anyone.

Name \_\_\_\_\_

Department \_\_\_\_\_

Institution \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State/Province \_\_\_\_\_

Postal code \_\_\_\_\_ Country \_\_\_\_\_

Email address \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

Position \_\_\_\_\_ Affiliation \_\_\_\_\_

## TUG Institutional Members

American Mathematical Society,  
*Providence, Rhode Island*

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

Center for Computing Sciences, *Bowie, Maryland*

CSTUG, *Praha, Czech Republic*

diacriTech, *Chennai, India*

Fermilab, *Batavia, Illinois*

Google, *San Francisco, California*

IBM Corporation, T J Watson Research Center,  
*Yorktown, New York*

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

Marquette University, Department of Mathematics,  
Statistics and Computer Science,  
*Milwaukee, Wisconsin*

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

MOSEK ApS, *Copenhagen, Denmark*

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

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*

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

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

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

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

University of Wisconsin, Biostatistics &  
Medical Informatics, *Madison, Wisconsin*

VTeX UAB, *Vilnius, Lithuania*

## 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 <http://tug.org/consultants.html>. If you'd like to be listed, please see that web page.

### Aicart Martinez, Mercè

Tarragona 102 4<sup>o</sup> 2<sup>a</sup>

08015 Barcelona, Spain

+34 932267827

Email: [m.aicart \(at\) ono.com](mailto:m.aicart@ono.com)

Web: <http://www.edilatex.com>

We provide, at reasonable low cost, L<sup>A</sup>T<sub>E</sub>X or T<sub>E</sub>X 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

PO Box 532281

Los Angeles, CA 90053

+1 213-617-8483

Email: [typesetting \(at\) dangerouscurve.org](mailto:typesetting@dangerouscurve.org)

Web: <http://dangerouscurve.org/tex.html>

We are your macro specialists for T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X fine typography specs beyond those of the average L<sup>A</sup>T<sub>E</sub>X macro package. If you use X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X 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 T<sub>E</sub>X book.

### Latchman, David

4113 Planz Road Apt. C

Bakersfield, CA 93309-5935

+1 518-951-8786

Email: [david.latchman \(at\) texnical-designs.com](mailto:david.latchman@texnical-designs.com)

[texnical-designs.com](http://www.texnical-designs.com)

Web: <http://www.texnical-designs.com>

L<sup>A</sup>T<sub>E</sub>X consultant specializing in: the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized 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](mailto:speter@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.

**Sievers, Martin**

Klaus-Kordel-Str. 8, 54296 Trier, Germany

+49 651 4936567-0

Email: [info \(at\) schoenerpublizieren.com](mailto:info@schoenerpublizieren.com)Web: <http://www.schoenerpublizieren.com>

As a mathematician with ten years of typesetting experience I offer  $\TeX$  and  $\LaTeX$  services and consulting for the whole academic sector (individuals, universities, publishers) and everybody looking for a high-quality output of his documents. From setting up entire book projects to last-minute help, from creating individual templates, packages and citation styles (BIB $\TeX$ , bib $\LaTeX$ ) to typesetting your math, tables or graphics — just contact me with information on your project.

**Sofka, Michael**

8 Providence St.

Albany, NY 12203

+1 518 331-3457

Email: [michael.sofka \(at\) gmail.com](mailto:michael.sofka@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,

**Sofka, Michael (cont'd)**

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](mailto:borisv@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 eighteen years of experience in  $\TeX$  and three decades of experience in teaching & training. I have authored several packages on CTAN, published papers in  $\TeX$  related journals, and conducted several workshops on  $\TeX$  and related subjects.

**Young, Lee A.**

127 Kingfisher Lane

Mills River, NC 28759

+1 828 435-0525

Email: [leeayoung \(at\) morrisbb.net](mailto:leeayoung@morrisbb.net)Web: <http://www.thesiseditor.net>

Copyediting your  $\text{.tex}$  manuscript for readability and mathematical style by a Harvard Ph.D. Your  $\text{.tex}$  file won't compile? Send it to me for repair. Experience: edited hundreds of ESL journal articles, economics and physics textbooks, scholarly monographs,  $\LaTeX$  manuscripts for the Physical Review; career as professional, published physicist.

**TUG2015**  
**Darmstadt, Germany**  
**July 20–22, 2015**  
[\*\*http://tug.org/tug2015\*\*](http://tug.org/tug2015)

## Calendar

### 2014

- Nov 8–9 The Twelfth International Conference on Books, Publishing, and Libraries, “Disruptive Technologies and the Evolution of Book Publishing and Library Development”, Simmons College, Boston, Massachusetts. [booksandpublishing.com/the-conference](http://booksandpublishing.com/the-conference)
- Nov 13–14 The Printing Historical Society’s 50<sup>th</sup> Anniversary, “Landmarks in Printing: from origins to the digital age”, St Bride Institute, London, UK. [printinghistoricalsociety.org.uk/forthcoming\\_phs\\_events/#144](http://printinghistoricalsociety.org.uk/forthcoming_phs_events/#144)
- Nov 14 TYPO Day, “Business Typography Talks”, München, Germany. [typotalks.com/day/muenchen-2014](http://typotalks.com/day/muenchen-2014)
- Nov 28–29 5<sup>th</sup> Meeting of Typography, “Ubiquitous”, Escola Superior de Tecnologia – IPCA, Barcelos, Portugal. [www.atypi.org/events/5th-meeting-of-typography](http://www.atypi.org/events/5th-meeting-of-typography)
- Dec 13 Phototypesetting Day, Museum of Printing, North Andover, Massachusetts. [www.museumofprinting.org](http://www.museumofprinting.org)

### 2015

- Feb 1 **TUG election:** nominations due. [tug.org/election](http://tug.org/election)
- Mar 6 *TUGboat* 36:1, submission deadline.
- Mar 7–9 Typography Day 2015, “Typography, Sensitivity and Fineness”, Industrial Design Center, Indian Institute of Technology, Bombay, India. [www.typoday.in](http://www.typoday.in)
- Mar 9 *TUGboat* 36:1, submission deadline (regular issue)
- Mar 19–21 “Publish or Perish? Scientific periodicals from 1665 to the present”. The Royal Society, London, UK. [royalsociety.org/events/](http://royalsociety.org/events/)

- Apr 16–19 DANTE Frühjahrstagung and 52<sup>nd</sup> meeting, Stralsund, Germany. [www.dante.de/events.html](http://www.dante.de/events.html)
- Apr 29– May 3 Bachtex 2015: 23<sup>rd</sup> Bachtex Conference, Bachotek, Poland. [www.gust.org.pl/bachtex](http://www.gust.org.pl/bachtex)
- Apr 30– May 1 TYPO San Francisco, Yerba Buena Center for the Arts, San Francisco, California. [typotalks.com/sanfrancisco](http://typotalks.com/sanfrancisco)
- May 21–23 TYPO Berlin 2015, “Character”, Berlin, Germany. [typotalks.com/berlin](http://typotalks.com/berlin)
- Jun 29– Jul 3 Digital Humanities 2015, Alliance of Digital Humanities Organizations, “Global Digital Humanities”, Sydney, Australia. [dh2015.org](http://dh2015.org)
- Jul 7–10 SHARP 2015, “The Generation and Regeneration of Books”. Society for the History of Authorship, Reading & Publishing, Longueuil/Montreal, Canada, [www.sharpweb.org](http://www.sharpweb.org)

### TUG 2015 Darmstadt, Germany.

- Jul 20–22 The 36<sup>th</sup> annual meeting of the T<sub>E</sub>X Users Group. [tug.org/tug2015](http://tug.org/tug2015)
- Jul 31 *TUGboat* 36:2, submission deadline (proceedings issue).
- Aug 9–13 SIGGRAPH 2015, “Xroads of Discovery”, Los Angeles, California. [s2015.siggraph.org](http://s2015.siggraph.org)
- Aug 24–28 SHARP 2015, Society for the History of Authorship, Reading & Publishing, Jinan, Shandong Province, China, [www.sharpweb.org](http://www.sharpweb.org)
- Oct 19–20 The Thirteenth International Conference on Books, Publishing, and Libraries, University of British Columbia, Vancouver, Canada. [booksandpublishing.com/the-conference-2015](http://booksandpublishing.com/the-conference-2015)

*Status as of 20 October 2014*

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](mailto:office@tug.org)). For events sponsored by other organizations, please use the contact address provided.

A combined calendar for all user groups is online at [texcalendar.dante.de](http://texcalendar.dante.de).

Other calendars of typographic interest are linked from [tug.org/calendar.html](http://tug.org/calendar.html).

**Introductory**

- 231 *Barbara Beeton* / Editorial comments
- typography and *TUGboat* news
- 244 *Charles Bigelow* / A letter on the persistence of (e)books
- the Kindle, Nook, Sony Reader, and Pierre-Simon Fournier
- 232 *Donald Knuth* / A footnote about ‘Oh, oh, zero’
- notes on early typesetting of computer programs by ACM and Addison-Wesley
- 274 *Gerd Neugebauer* / CTAN goes multi-lingual: Additional language support for the Web portal
- making `ctan.org` available in German, as an experiment
- 230 *Steve Peter* / Ab epistulis
- upcoming election, conferences, *TUGboat*, book reviews
- 276 *Basil Solomykov* / Obyknovennaya Novaya (Ordinary New Face) in METAFONT
- a reworking of a famous Cyrillic typeface, in several sizes and shapes

**Intermediate**

- 315 *Karl Berry* / The treasure chest
- new CTAN packages, March–September 2014
- 256 *Ulrike Fischer* / `biblatex` variations
- `biblatex` as a database: QR codes, PDF attachments, address lists
- 235 Twenty Questions for Donald Knuth
- to celebrate the publication of TAOCP as eBooks
- 284 *Bob Tennent* / Visual editing (in a specialized case): `prerex`
- a useful application for visual editing—charts of course prerequisites
- 245 *Thomas Thurnherr* /  $\LaTeX$  document class options
- options for the standard classes, and packages extending similar functionality
- 269 *Peter Wilson* / Glisterings: Lining up
- ruling off; marginal rules; preventing an awkward page break; not at a page break; line backing; linespacing

**Intermediate Plus**

- 248 *Frank Mittelbach* / How to influence the position of float environments like figure and table in  $\LaTeX$ ?
- explaining and working with the  $\LaTeX$  float placement algorithm
- 287 *Frank Mittelbach, Will Robertson,  $\LaTeX$ 3 team* / `l3build`—A modern Lua test suite for  $\TeX$  programming
- regression testing for  $\LaTeX$ , including typeset output
- 309 *Daniel Stender* / A Citation Style Language (CSL) workshop
- introduction to this XML-based language for programming citation and bibliography styles
- 261 *David Walden* / Every  $\LaTeX$  document brings new programming issues
- practical approaches for ellipses, blank verso pages, and photo album layout

**Advanced**

- 255 *Barbara Beeton* / Placing a full-width insert at the bottom of two columns
- even on the first page of an article
- 277 *Yannis Haralambous* / A simple Arabic typesetting system for mixed Latin/Arabic documents: *dād*
- supporting both transliteration and direct Unicode input of Arabic, using ligatures
- 294 *Taco Hoekwater* / MetaPost path resolution isolated
- new interface in MPlib 1.800 for resolving paths from external programs
- 297 *Udo Wermuth* / Typeset MMIX programs with  $\TeX$
- a  $\TeX$  macro package to typeset (M)MIX(AL) programs

**Contents of other  $\TeX$  journals**

- 322 *Die  $\TeX$ nische Komödie 2–3/2014; Les Cahiers GUTenberg 56 (2012)*

**Reports and notices**

- 317 *William Adams* / Book review: *Practical  $\LaTeX$* , by George Grätzer
- review of this introductory text on document production with  $\LaTeX$
- 318 *Jacques André* / Book review: *Apprendre à programmer en  $\TeX$* , by Christian Tellechea
- review of this book in French on  $\TeX$  as a programming language
- 319 *Michael Berry* / Book review: *The Imitation Game*, by Jim Ottaviani and Leland Purvis
- review of this graphic novel about the life of Alan Turing
- 320 *Nicola Talbot* / Book review: *Let's Learn  $\LaTeX$* , by S. Parthasarathy
- review of this free ebook intended to assist learning  $\LaTeX$  by example
- 323  *$\TeX$  Development Fund committee* /  $\TeX$  Development Fund 2013 report
- 324 *TUG Election committee* / TUG 2015 election
- 325 TUG membership form
- 326 Institutional members
- 326  $\TeX$  consulting and production services
- 327 TUG 2015 announcement
- 328 Calendar