TUGboat

Volume 30, Number 2 / 2009
TUG 2009 Conference Proceedings

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group.

Memberships and Subscriptions

2008 dues for individual members are as follows:

- Ordinary members: $85.
- Students/Seniors: $45.

The discounted rate of $45 is also available to citizens of countries with modest economies, as detailed on our web site.

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. Individual membership 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 $100 per year, including air mail delivery.

Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group, as well as providing a discounted group rate and other benefits. For further information, see `http://tug.org/instmem.html` or contact the TUG office.

TEX is a trademark of the American Mathematical Society.

Addresses

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

Telephone

+1 503 223-9994

Fax

+1 206 203-3960

Web

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

Electronic Mail
(Internet)

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

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

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

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

Have a suggestion? Problems not resolved?
The TUG Board wants to hear from you:
Please email `board@tug.org`.

[printing date: October 2009]

Printed in U.S.A.

TUGBOAT

The Communications of the TEX Users Group

Volume 30, Number 2, 2009
TUG 2009 Conference Proceedings

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group.

Memberships and Subscriptions

2008 dues for individual members are as follows:

- Ordinary members: $85.
- Students/Seniors: $45.

The discounted rate of $45 is also available to citizens of countries with modest economies, as detailed on our web site.

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. Individual membership 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 $100 per year, including air mail delivery.

Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group, as well as providing a discounted group rate and other benefits. For further information, see `http://tug.org/instmem.html` or contact the TUG office.

TEX is a trademark of the American Mathematical Society.

Addresses

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

Telephone

+1 503 223-9994

Fax

+1 206 203-3960

Web

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

Electronic Mail
(Internet)

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

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

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

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

Have a suggestion? Problems not resolved?
The TUG Board wants to hear from you:
Please email `board@tug.org`.

[printing date: October 2009]

Printed in U.S.A.

2009 Conference Proceedings

TEX Users Group
Thirtieth Annual Meeting
University of Notre Dame
Indiana, USA
July 29–31, 2009

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
TUGboat Editor Barbara Beeton
Proceedings Editor Karl Berry

Volume 30, Number 2 • 2009
Portland • Oregon • U.S.A.

TUG 2009

University of Notre Dame ■ Indiana, USA

July 28–July 31, 2009

Sponsors

TEX Users Group ■ Notre Dame Journal of Formal Logic ■ DANTE e.V.

A-R Editions ■ Design Science ■ MacKichan Software
O'Reilly Media ■ River Valley Technologies

Thanks to all!

Acknowledgments

Many thanks to all the speakers and teachers, without whom there would be no conference, and also special thanks to:

- Martha Kummerer, for the local research and organization.
- Duane Bibby, for the (as always) excellent and fun drawings.
- University of Notre Dame, for the facilities, with particular thanks to Harriet Baldwin.

Conference committee

Karl Berry ■ Martha Kummerer ■ Robin Laakso ■ Steve Peter

Bursary committee

Sam Rhoads, chair ■ Jana Chlebikova ■ Kaja Christiansen ■ Bogusław Jackowski

Participants

Leyla Akhmadeeva, Bashkir State Medical Univ.
Bob Alps
Tim Arnold, SAS
Kaveh Bazargan, River Valley Technologies
Nelson Beebe, University of Utah
Barbara Beeton, American Mathematical Society
Karl Berry, TEX Users Group
Jin-Hwan Cho, University of Suwon
Dennis Claudio, Richmond, CA
Jennifer Claudio, St. Lawrence Academy
Joseph Corneli, PlanetMath
Paulo Ney de Souza, UC Berkeley
Sue DeMeritt, Center for Communications Research, La Jolla, CA
Dorothy DiFranco, American Physical Society
Terri Fizer, Duke University Press
Steve Grathwohl, Duke University Press
Hans Hagen, Pragma ADE
Michele Hake, American Physical Society
Idris Samawi Hamid, Colorado State University
Jim Hefferon, St. Michael's College
Morten Høgholm, LATEX Project and Technical University of Denmark
Klaus Höppner, DANTE e.V.
Ned Hummel, Indiana University-Purdue University Indianapolis
Mirko Janc, INFORMS
Jonathan Kew, Mozilla Corporation
Dick Koch, University of Oregon
Martha Kummerer, Notre Dame Journal of Formal Logic
Aditya Mahajan, University of Michigan
Andrew Mertz, Eastern Illinois University
Ross Moore, Macquarie University
Bob Neveln, Widener University
Lynn Newton, Interhack Corp.
Cheryl Ponchin, Center for Communications Research, Princeton, NJ
Gianluca Puliti, University of Notre Dame
Arthur Reutenauer, GUTenberg
Chris Rowley, Open University
Herbert Schulz, College of DuPage (retired)
William Slough, Eastern Illinois University
Boris Veytsman, George Mason University
David Walden, E. Sandwich, MA
Alan Wetmore, US Army

LATEX workshop participants

Jason Bray, University of Notre Dame
Dorothy DiFranco, American Physical Society
Huijing Du, University of Notre Dame
Terri Fizer, Duke University Press
Michele Hake, American Physical Society
Shari Hill, Notre Dame Graduate School
Judy Hygema, University of Notre Dame
Brenda Jackson, University of Notre Dame
Matthew Jobbins, University of Notre Dame
Gianluca Puliti, University of Notre Dame
Aaron Segal, University of Notre Dame

TUG 2009 program

Tuesday July 28

8:30 am	*registration*
9 am–5 pm	track 1: *LaTeX workshop,* Sue DeMeritt & Cheryl Ponchin
9 am	track 2: *LaTeX color/graphics workshop,* Boris Veytsman
10:30–10:45 am	*break*
10:45 am	track 1 continues
10:45 am	track 2: *PSTricks workshop,* Boris Veytsman
12:15–1:30 pm	*lunch*
1:30 pm	track 1 continues
1:30 pm	*TeX Live 2009 round table,* Karl Berry
3:00–3:15 pm	*break*
3:15 pm	track 1 continues
3:15 pm	*CTAN round table,* Jim Hefferon
5–7 pm	*registration & reception,* Morris Inn, Irish Courtyard

Wednesday July 29

8 am	*registration*	
8:30 am	Karl Berry, TeX Users Group	*Welcome*
8:45 am	Ross Moore, Macquarie University	*Towards tagged PDF*
9:25 am	Kaveh Bazargan, River Valley Tech.	*TeX as an eBook reader*
10:05 am	*break*	
10:20 am	Andrew Mertz & William Slough, EIU	*A TikZ tutorial: Generating graphics in the spirit of TeX*
11:00 am	Klaus Höppner, DANTE e.V.	*Introduction to MetaPost*
11:40 am	Jim Hefferon, St. Michael's College	*Gyre Fonts for the easily confused*
12:20 pm	*lunch*	
1:30 pm	Aditya Mahajan, Univ. of Michigan	*LuaTeX: A user's perspective*
2:10 pm	Arthur Reutenauer, GUTenberg	*LuaTeX for the LaTeX user*
2:50 pm	*break*	
3:05 pm	Frank Quinn, Virginia Tech	*EduTeX: A source format for self-scoring tests*
3:30 pm	Jin-Hwan Cho, University of Suwon	*Extended DVI formats and DVIasm*
4:10 pm	Nelson Beebe, University of Utah	*BibTeX meets relational databases*
4:50 pm	q&a	

Thursday July 30

8:45 am	Boris Veytsman, George Mason Univ.	*Class writing for wizard apprentices*
9:25 am	Karl Berry & David Walden, TUG	*TeX People: The TUG interviews project and book*
10:05 am	*break*	
10:20 am	David Walden	*Self-publishing: experiences and opinions*
11:00 am	Paulo Ney de Souza, MSP	*The MSP TeX production system: Restoration of TUGboat*
11:40 am	Idris Hamid, Colorado State Univ.	*Arabic typography: Past, present, and TeX*
12:20 pm	*lunch*	
1:30 am	Hans Hagen, Pragma ADE	*The LuaTeX Project: Halfway to version 1*
2:10 pm	Aditya Mahajan	*ConTeXt math: Integrating Unicode and OpenType*
2:50 pm	*break*	
3:05 pm	Hans Hagen	*Plain TeX and OpenType*
3:45 pm	Idris Hamid	*Dynamic Arabic: Towards the philosopher's stone of Arabic-script typography*
4:25 pm	q&a, TUG meeting	
7 pm	*banquet* Morris Inn, Donors Room	

Friday July 31

8:45 am	Bob Neveln, Widener University & Bob Alps	*Writing and checking complete proofs in LaTeX*
9:25 am	Morten Høgholm, LaTeX Project	*Next steps for breqn*
10:05 am	*break*	
10:20 am	Boris Veytsman & Leyla Akhmadeeva, Bashkir State Medical University	*Medical pedigrees: typography and interface*
11:00 am	Tim Arnold, SAS	*Getting started with plasTeX*
11:40 am	Chris Rowley, Open University	*TeX-free LaTeX, an overview*
12:20 pm	*lunch*	
1:30 pm	Chris Rowley	*Standards for LaTeX documents and processors*
2:10 pm	Morten Høgholm	*Consolidation of expl3*
2:50 pm	Hans Hagen	*Why I still use TeX*
3:30 pm	*end*	

Cheryl Ponchin and Sue DeMeritt, at the reception.

Hans Hagen and Jin-Hwan Cho.

Boris Veytsman and Leyla Akhmadeeva.

Gianluca Politi.

Dennis Claudio, Jonathan Kew, Paulo Ney de Souza.

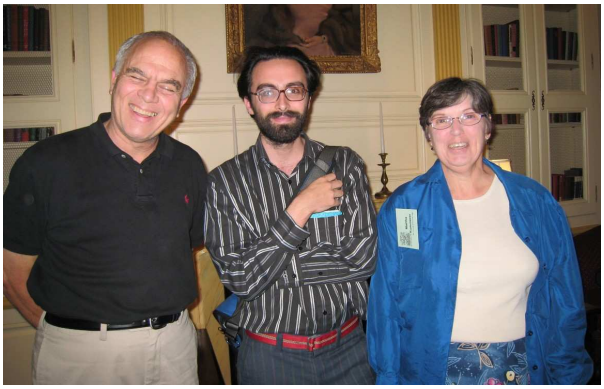
Idris Hamid, Barbara Beeton, Nelson Beebe.

Kavah Bazargan, our estimable banquet MC.

Idris Hamid and Hans Hagen.

Morten Høgholm and family: Evguenia, David, and Abigail.

Our after-dinner entertainment provided by Morten and Evguenia's son David.



Frank Quinn, Joe Corneli, Martha Kummerer.

Steve Grathwohl and Peter Cholak.

MacTEX at work: Jonathan Kew and Dick Koch.

Tim Arnold, Jim Hefferon, Andrew Mertz, and Bill Slough. *Background:* Karl Berry, Bob Neveln, Bob Alps, Leyla Akhmadeeva, Chris Rowley.

Martha Kummerer, our excellent tour guide, with Nelson Beebe.

The TEX pack on tour.

TUG 2009 gathering.

Front row: Dennis Claudio, Terri Fizer, Dorothy DiFranco, Michele Hake, Bram Otten, Martha Kummerer, Ross Moore, Tim Arnold, Kaveh Bazargan, Cheryl Ponchin, Sue DeMeritt, Jin-Hwan Cho.

Second row: Jennifer Claudio, Andrew Mertz, Jonathan Kew, Leyla Akhmadeeva, Mirko Janc, Steve Grathwohl, Bob Alps, Frank Quinn, Barbara Beeton, Dave Walden, Ned Hummel, Bob Neveln, Dick Koch.

Back row: Hans Hagen, Idris Hamid, Klaus Höppner, Gianluca Puliti, Joe Corneli, Boris Veytsman, Nelson Beebe, Arthur Reutenauer, Aditya Mahajan, Karl Berry, William Slough, Herb Schulz, Paulo Ney de Souza, Peter Cholak, Jim Hefferon, Morten Høgholm, Chris Rowley.

Photos courtesy of Jennifer Claudio and Alan Wetmore.

Profile of Eitan Gurari (1947–2009)

Eitan Gurari died unexpectedly on June 22, 2009. The TEX community mourns the loss of this important contributor.

Eitan Gurari was born in March 1947 in Haifa, Israel, and grew up in Tivon, a small town near Haifa. He met his wife, Shaula, who was born and raised in Haifa, at a dance. They had five children: A daughter Inbal, sons Itai and Erez, and twin daughters Netta and Danna. Eitan's son, Itai, says, "My father's sole hobbies were his work and family. He enjoyed both immensely. Also, in recent years he and my mother began traveling a lot." The children had all grown to adulthood at the time of Eitan's death. Eitan's 1989 book, *An Introduction to the Theory of Computation*, was dedicated,

To Shaula, Inbal, Itai, Erez, Netta, and Danna.

Eitan was educated at Technion–Israel Institute of Technology where, in 1971, he received a Bachelor of Science degree in physics. He continued his studies there, but changed his focus to computer science, receiving a Masters degree in 1974. At that point he, his wife, and their then only child, a daughter, moved to the United States where Eitan studied at the University of Minnesota, which granted him a PhD degree in computer science in 1978. Eitan's faculty advisor at the University of Minnesota was Oscar Ibarra.

After graduation from the University of Minnesota, Eitan moved to the University of Wisconsin–Milwaukee, where until 1980 he was an assistant professor in the Department of Electrical Engineering and Computer Science. He next moved to the State University of New York at Buffalo, Department of Computer Science, where he served as an assistant professor until 1982. In 1982, Eitan moved to the Ohio State University, where he was an associate professor teaching computer science and doing research until his death.

At Ohio State, Eitan taught a wide variety of courses, suggesting that he pitched in to teach whatever needed teaching and that he used the need to teach a course in a new area as a way of broadening his own knowledge. His web site at the time of his death listed the following courses that he had taught (we don't know if this is a complete list): Introduction to Computer Graphics, 1993; Introduction to Computer Networks, 1998; Introduction to Data Structures, 1999; Introduction to Automata and Formal Languages, 1999; Introduction to Programming with Java, 2003; Advanced Technologies with Java, 2004; Programming Challenges, 2004; XML Technologies, 2005; Compiler Design and Implementation, 2008; Introduction to Programming with C++, 2008; Data Structures for Information Systems, 2008; Introduction to Database Systems, 2009.

Early in his career, Eitan's interest was in the theory of computation, and between 1978 and 1987 he had 18 papers in this field published in refereed journals, many of them co-authored with Oscar Ibarra. Two typical titles are "Two-Way Counter Machines and Diophantine Equations" and "Some Decision Problems Concerning Sequential Transducers and Checking Automata". During this period Eitan also had six papers published in conference or symposium proceedings. The culmination of this work appears to have been his already mentioned 1989 book on the theory of computation.

Eitan's obituary notice on the website of the Computer Science and Engineering Department of Ohio State University said the following:

> Dr. Gurari started his career as a theoretician. He made fundamental contributions to automata and complexity theory. His textbook, *An Introduction to the Theory of Computation*, was highly praised, and he published frequently in JACM, SIAM Computing, ACM STOC, and IEEE FOCS.

The obituary notice continued,

> After joining OSU, Gurari switched his research focus, starting to build software systems.

We can see this transition from his writings and presentations in the succeeding years.

1991 "A WYSIWYG Literate Programming System" (Preliminary Report), with J. Wu, Nineteenth ACM Computer Science Conference.

1994 *TEX and LATEX: Drawing and Literate Programming*, A manual for DraTEX, AlDraTEX, ProTEX, and AlProTEX, McGraw-Hill
Writing with TEX, McGraw-Hill.

1997 "A Demonstration of TEX4ht", TUG '97
"Drawing with DraTEX", TUG '97

1999 "LaTeX to XML/MathML", with S. Rahtz, TUG '99 workshop and abstract in *TUGboat*

The LaTeX Web Companion, by M. Goossens and S. Rahtz, with contributions by E. Gurari, R. Moore, and R. Sutor, Addison-Wesley

2000 "From LaTeX to MathML and Back with TeX4ht and PassiveTeX", with S. Rahtz, The first MathML International Conference, Urbana–Champaign, Illinois.

2003 "From LaTeX to MathML and Beyond", TUG '03

2004 "TeX4ht: HTML production", Practical TeX 2004

"XML and MathML production through LaTeX", keynote presentation at the Second European Workshop on MathML & Scientific e-Contents, Kuopio, Finland

2005 "SwiExr: Spatial math exercises and worksheets, in Braille and print", TUG '05

"MathML via TeX4ht and other tools", Practical TeX 2005

2007 "LaTeX conversion into normalized forms and speech", TUG '07

Sebastian Rahtz remembers becoming aware of Eitan's work in the world of TeX as follows:

> I first came across Eitan Gurari when I was looking at LaTeX to SGML conversion in the late '90s. I had worked on a system at Elsevier in which we took apart a DVI file decorated with `\special` commands, and I was rather pleased with it. Then I saw a reference to Eitan's TeX4ht system, and realized that he had gone down the same road, but with a much more sophisticated setup. Michel Goossens and I had earlier tried to document Eitan's clever DraTeX drawing macros, so when we were about to start on the *LaTeX Web Companion*, we decided to ask Eitan to contribute a chapter about his system. I like to think that this helped bring even more people's notice to Eitan's remarkably ingenious work in this fascinating side road of TeX.
>
> I did not meet Eitan in person until the TUG 1999 meeting, at which we did a joint session. He turned out as I expected — quiet, humble, and seemingly quite surprised (but pleased) to find out that people appreciated his work. That set the tone for our communication over the following years as the *LaTeX Web Companion* was completed, and he continued to improve TeX4ht. He was always apparently pleased to be told of errors, and quickly fixed them, and was pleased with feedback on his writing. I don't remember a cross word or a disagreement.
>
> I shall remember Eitan with pleasure and gratitude for his careful and innovative contribution to TeX.

Obviously Eitan's TeX4ht had a large impact on the TeX world, as a number of notes since his death to the `comp.text.tex` list and to TUG president Karl Berry have testified.

Karl Berry himself remembers,

> I had two kinds of dealings with Eitan over the years: *TUGboat* papers and TeX4ht software. The *TUGboat* interactions were all straightforward and completely cordial. He got his stuff in on time and was perfectly happy to accept all editing changes. If only all authors were so accommodating!
>
> With TeX4ht, things were a bit more extensive. The TeX Live guide (`http://tug.org/texlive/doc.html`) is written in LaTeX, and translated into several languages. All are converted to HTML using TeX4ht. Sebastian set it up this way back in the first releases of TeX Live, and I stayed with the same procedure when I became the TL editor. Both of us would inevitably find problems with the HTML output at the last minute (the doc is always the last thing to be done), especially given the plethora of translations. Eitan logged in to `tug.org` many times to debug the problems; he invariably found solutions within a day or two, usually sooner. It was amazing to me, given how complex and huge TeX4ht is.
>
> The TeX Live doc has always had this paragraph in the list of thanks:
>
> - Eitan Gurari, whose TeX4ht was used to create the HTML version of this documentation, and who worked tirelessly to improve it at short notice.
>
> The TL 2009 documentation is dedicated to Eitan, in honor of his remarkable contributions.

While many people in the TeX community have used TeX4ht at one time or another to convert a TeX file to HTML (some with considerable regularity as Karl Berry described), TeX4ht was also used by some as a key component in their business workflow. CV Radhakrishnan of River Valley Technologies describes their use as follows:

> We use TeX4ht on a daily basis; in short, the existence of River Valley is wholly dependent on the TeX4ht system. It is one of the brilliant pieces of software written in the TeX language. TeX4ht can digest any LaTeX document and

> output appropriate XML or HTML depending on users' requirements. We use it for generating different kinds of XML from LaTeX documents based on different client DTDs, without human intervention. The main point is that TeX4ht permits command line invocation (in fact, there is no graphical interface) and, therefore, integrates well into our fully automated work flow. Many people see TeX4ht as a monster which defies taming. Our experience is different; TeX4ht is a highly configurable and scalable system which can effectively be used to derive different kinds of formats from LaTeX sources including HTML, XML, OpenOffice documents, braille, etc. with remarkable ease. I have even used TeX4ht to convert an author-macro-ridden LaTeX document into standard LaTeX by degrading all the complex author macros into corresponding TeX/LaTeX primitives!

To this, CVR's partner in River Valley, Kaveh Bazargan, adds:

> Eitan's death is a great loss to our community, greater than most realise. I echo CVR's comments. It is no exaggeration that our company is based around TeX4ht which is an order of magnitude more capable of doing the complex tasks we undertake than any other software.

A common thread among comments about Eitan was his quick responsiveness when problems were found with his software. A quote from Kapil Hari Paranjape, who maintains the "downstream" packaging of TeX4ht for Debian, illustrates the extent to which Eitan went to be responsive:

> I always found [Eitan to be] a responsive upstream who was willing to be patient and explain his way of maintaining the package to the Debian developer community. As soon as I mentioned that it was possible for him to obtain Debian bug reports directly by subscribing to the Package Tracking System, he did so and started responding to such reports with fixes.
>
> The major request which Debian made was that the literate sources of TeX4ht be made available in order to comply with the Debian Free Software Guidelines. This was a big task for him as he described his system as "put together in the basement/garage". However, he took up this task and did it and we are all grateful for it.

Dr. Susan Jolly, a computational scientist retired from Los Alamos National Laboratory, was instrumental in Eitan becoming involved with braille. Susan, who is not blind, has a deep interest and commitment in improving the efficiency of transcription of books and journals, especially math books and journals, into braille (`www.dotlessbraille.org`). She recalls,

> I first wrote to Eitan on July 26, 2001, to ask him about TeX4ht (which I'd found via Google) and to tell him that it should be possible to use TeX4ht as the basis for a project that would mean "that a huge legacy of mathematics could be made available in braille".
>
> Eitan wrote back two hours and eight minutes later to say, "I would love to be involved in such a project". And we were off and running. Of course, at that point, neither of us appreciated how difficult the project would turn out to be nor how long it would take nor how much fun we'd have.
>
> By the end of September 2001, Eitan had drafted a proposal to the NSF. This first proposal was not funded but the similar second one was.

The issue, as described by Susan, is roughly as follows. Braille involves "cells" of six embossed (raised) dots (in a 2-wide 3-high arrangement) allowing 63 different configurations of dots. What these dot configurations mean depends on the "braille code" that is being used: there is a basic code for novice learners, a more advanced code, a literary code, codes for math, music, and different languages, etc. The various codes involve context dependent meanings and contractions, signals for changing among codes, etc.; with the exception of the most basic code for new learners, things are not nearly as simple as just spelling out every printed word, which is impossible in any case for mathematics. All in all, it is far from straightforward to transcribe from print to braille. In many cases the transcription process begins with optical character recognition of a printed page, which must be corrected by transcribers as they also convert the sequence of characters and words into appropriate plain text representations of each of the 63 different cell configurations, typically with the aid of commercial print-to-braille software. Software and hardware can then convert these plain text files into pages of embossed braille. The reverse conversion is also sometimes used as a way of checking the initial transcription. The entire process is time consuming and expensive with not nearly enough people available to do the transcriptions to braille, especially

transcriptions involving math (which is also an area of weakness for print-to-braille software).

The reasons why Eitan's work was valuable are as follows:

- There has not been much math (or science more generally) in braille because of the effort required to convert to braille. There is a particular shortage of transcribers familiar with the Nemeth code used for transcribing technical material.
- There is a large legacy of math, etc., already in journals which accept TEX (by which we also mean LATEX, AMS-LATEX, etc.) input and books written using TEX, and new TEX-based books and journals are being produced all the time. Increasingly these are available in machine readable form, e.g., at `arXiv.org` and an initiative of the American Physical Society to make the source files of its publications available which hopefully will spread to other organizations.
- However, source TEX usually has all sorts of stuff about how to make it look on the printed page which is uninteresting to someone just trying to transcribe it to another format (i.e., Nemeth braille). But TEX4ht already does the job of throwing away all that finicky detail about actual printing and converts the output of a TEX system to a much simpler HTML format. Also, the source TEX code has the math in a form that maintains its meaning (not just combinations of characters on a printed page), and TEX4ht already knows how to convert that format into something else, i.e., MathML.
- Thus, a good place to add braille conversion is as an optional output of TEX4ht.

But, of course, there were lots of unsolved details plus the actual prototyping effort which thus made it a plausible academic project requiring grant funding. For instance, some spatial or planar (2-D) items, i.e., elementary arithmetic, tables and matrices, have prescribed planar renderings in braille. Eitan was especially interested in automating the associated complex formatting problems.

Eitan was scheduled to make a presentation on his work with braille at TUG '09 (held a month after his death) entitled, "SuBrl: A LATEX to braille converter: A first look at a forthcoming system". His pre-conference abstract said,

> SuBrl is a system under development for translating LATEX and XML data into braille. The presentation will demonstrate the translation of LATEX and XML into braille, describe the architecture of the system, discuss issues that require special attention in LATEX and XML sources, and argue the benefit of a LATEX front end for braille production.

Regarding the name of Eitan's system, he wrote to Susan Jolly in an email dated March 19, 2009:

> I was asked to give a talk at the end of July to the annual TEX Users Group meeting. I decided to give a talk about some insight into LATEX translation to braille (with the hope that at that time I'll have some core system nearly ready to release). I'm going to call the system SuBrl. I would like to publicly dedicate the "su" component to your suggesting the project and generously introducing me [to the world of] braille. I hope you'll permit me to do so.

Eitan's children are working with his university to get access to his files for his work with braille with the hope that someone can be found to continue his work in this area.

Also with the encouragement and support of Eitan's children, TEX4ht in general will continue to be maintained and supported. CV Radhakrishnan and Karl Berry have created a public project (see `http://tug.org/tex4ht`) and welcome any involvement. Having this effort continued by the TeX community and others will be one appropriate memorial to Eitan's life and work.

Compiled by David Walden, July 2009

LaTeX class writing for wizard apprentices

Boris Veytsman

Abstract

A number of excellent articles explain LaTeX class writing for beginners (Hefferon, 2005; Flynn, 2007; Pakin, 2008; Mansfield, 2008). Of course, true wizards do not need any instructions. This paper is intended for those TeXnicians who are no longer beginners, but may not (yet) qualify as wizards. It discusses some tips and tricks of the trade: packages you may want to use and packages you must be compatible with; why the first two pages of a book take 60% of your time; what is wrong with the LaTeX sectioning interface but why you better stick with it, and more. It is based on the experience of the author in writing LaTeX classes for various customers (Veytsman, 2008c).

1 Audience and scope

The New Hacker's Dictionary (Raymond, 1996) defines the word "user" in the following way:

> **user:** *n.*
>
> 1. Someone doing "real work" with the computer, using it as a means rather than an end. Someone who pays to use a computer. See *real user.*
> 2. A programmer who will believe anything you tell him. One who asks silly questions. [GLS observes: This is slightly unfair. It is true that users ask questions (of necessity). Sometimes they are thoughtful or deep. Very often they are annoying or downright stupid, apparently because the user failed to think for two seconds or look in the documentation before bothering the maintainer.] See *luser.*
> 3. Someone who uses a program from the outside, however skillfully, without getting into the internals of the program. One who reports bugs instead of just going ahead and fixing them.
>
> The general theory behind this term is that there are two classes of people who work with a program: there are implementors (hackers) and *lusers.* The users are looked down on by hackers to some extent because they don't understand the full ramifications of the system in all its glory. (The few users who do are known as real winners.) The term is a relative one: a skilled hacker may be a user with respect to some program he himself does not hack. A LISP hacker might be one who maintains LISP or one who uses LISP (but with the skill of a hacker). A LISP user is one who uses LISP, whether skillfully or not. Thus there is some overlap between the two terms; the subtle distinctions must be resolved by context.

In our TeX world the distinction between users and implementors is sometimes represented as a distinction between users and TeXnicians. The latter word was introduced by DEK himself (Knuth, 1994). A TeXnician is a person who helps other people to use TeX. This paper is intended for them.

More specifically, here we discuss writing LaTeX classes. I hope some of these issues might be of interest for the macro writers working with other formats, but I personally have been dealing mostly with LaTeX.

The TeXnicians working with macro packages can be divided into three groups. First, there are true wizards (see the definition and discussion in *New Hacker's Dictionary*). They can easily and confidently write pages of TeX code with all manner of `\expandafter` and `\futurelet` uses. Clearly these people do not need to be taught how to write classes (at least by me!).

Next, there are beginners. I do not mean here beginning *users,* just the opposite: this group largely consists of experienced users trying themselves in class writing. There is good and useful literature for this category of TeXnicians; I heartily recommend the papers by Hefferon, 2005; Flynn, 2007; Pakin, 2008; Mansfield, 2008.

The third group consists of people who already authored one or several classes, know how to use the DTX format, have a dog-eared copy of the *Companion* (Mittelbach, Goossens, Braams, Carlisle, and Rowley, 2004) and can recite paragraphs from the *Class Guide* (2006). For the lack of better term we will call such people *wizard apprentices.* I am proud to belong to this group myself. One of my longstanding complaints has been the relative dearth of literature intended for this category of macro writers. The great book by Eijkhout, 2007, is one of the rare exemptions from this rule. This article is also intended to partially fill this need.

I provide here some anecdotes and snippets from my experience as a LaTeX macro writer. If it seems rather subjective and opinionated, you, the reader, are welcome to offer your own point of view. I am certainly open to suggestions and critique from my fellow apprentices, as well as from the wizards and the beginners.

2 LaTeX interfaces

It is well known that the design of the standard LaTeX classes is not optimal from the typographer's point of view. The fact that its user interfaces also leaves much to be desired is probably less appreciated.

Let us consider, for example, the LaTeX sectioning commands. The standard `\chapter` command and its sisters `\section`, `\subsection`, etc. have two arguments: the mandatory title and the optional short title. The latter, if present, is used for the table of contents and headers. This means that the entry in the table of contents must be the same as the headers. This is not what we want in most cases. It is more common for modern books to use the "long" title in the table of contents and the "short" one in the headers.

The starred variants `\chapter*`, `\section*`, etc. have their own problems. First, the star is overloaded: it signifies *both* that this heading is not numbered *and* that it does not produce an entry in the table of contents. This is a wrong idea in most cases. Such unnumbered subdivisions as *Bibliography* and *Index* should be mentioned in the table of contents. The same is true for front matter headings like *Foreword,* which are usually not numbered, but nevertheless belong in the table of contents. One of the frequently asked questions in the Usenet newsgroup `comp.text.tex` is "how to add the bibliography to the TOC?" Of course there are packages which achieve this, but they are correcting a design misfeature which should not be there in the first place. Another problem is the fact that the starred sectioning commands do not have the optional argument (and do not change the headers in the standard classes). Again, in most cases we do want to change the headers for unnumbered subdivisions and thus need a way to set up a short version of the title.

It is reasonable for a class writer to correct these misfeatures and create a more rational interface design. For example, most of the problems discussed above can be easily solved in a new class. However, one must be very careful when changing the interfaces — even if it is tempting to do so. First, the users are accustomed to the LaTeX design, and drastic changes might be too difficult for them. Second, many computer editors are "LaTeX-aware", and offer users ready-made templates. Third, some useful packages like *hyperref* (Rahtz and Oberdiek, 2006) redefine LaTeX commands, and expect certain interfaces to be there.

The following example from my experience illustrates this point. Books published by *Nostarch Press* have so-called *circular art*: a small round picture in the beginning of each chapter. In the preliminary version of the class for this publisher I used the following interface with one optional and two mandatory arguments:

`\chapter[`⟨*Short Title*⟩`]{`⟨*Long Title*⟩`}{`⟨*Artwork*⟩`}`

This was a disaster. It turned out to be difficult for me to remember that `\chapter` now has *two* mandatory arguments, so I got a slew of errors during testing. Moreover, *hyperref* refused to understand this syntax, and I decided that patching it would be too cumbersome. In the end I opted for another design (Veytsman, 2008b):

`\chapter[`⟨*Short Title*⟩`]{`⟨*Long Title*⟩`}`
`\chapterart{`⟨*Artwork*⟩`}`

3 Compatibility issues

In the previous section we briefly touched on the issue of compatibility. Let us discuss it more thoroughly.

The LaTeX world can be described as a motley collection of packages written by different authors with various philosophies, goals, design ideas and skill levels. Nobody ever guaranteed that these packages would work together. The fact that TeX has no concept of namespace makes the combination of these packages even more daunting.

To tell the truth, though, the situation is better than it could be. Most packages are compatible, and the authors usually take care to patch them if they turn out not to be. Still, sometimes interactions between packages lead to unexpected results.

Some publishers, when accepting LaTeX manuscripts, restrict the authors' choice of packages. Nevertheless there are always situations when an author or an editor really needs a package.

A macro writer should expect the users to load some packages. A good policy is to proactively test the class with the most popular ones and mention such "approved" packages in the documentation. In some cases it makes sense to automatically load these packages from the class.

If an electronic publication is intended (which is almost always the case nowadays), *hyperref* (Rahtz and Oberdiek, 2006) is usually called. This package has many settings changing the appearance of links (the default ones are almost never satisfactory). However, setting them presents the following problem. This package usually should be loaded last, after others. If you load it from your class, the packages called by the users may not work properly.

There are two ways to solve this problem. First, you can write a wrapper package that calls *hyperref*, sets it up and even patches it if necessary. I

adopted this approach for the *nostarch* class (Veytsman, 2008b). Second, you can automatically check at the beginning of the document whether *hyperref* is loaded, and set it up if necessary with a construction like this:

```
\AtBeginDocument{%
   \@ifpackageloaded{hyperref}{%
      \hypersetup{⟨Your customization⟩}}{}}
```

I used this method for the class for an online journal *Philosophers' Imprint* (Veytsman, 2007).

Most people use the graphics bundle (Carlisle, 2005) and sometimes other graphics solutions for illustrations. If your users are mathematicians, they probably are going to load *amsmath* (2002). Programmers usually call *listings* (Heinz and Moses, 2007). Many people working in "natural sciences" need *natbib* (Daly, 2009). You may want to test your class with these packages. In general, a class writer should understand the users' ways of doing things and anticipate their actions.

It is difficult enough to ensure compatibility with the plethora of LaTeX packages, but in fact the problem is even more complex, because these package are not frozen. Sometimes when a user upgrades a package, it breaks many existing documents. This requires constant vigilance from the class author.

4 Some useful tools

Besides creating problems for a class author, the great variety of LaTeX packages has positive value too. Some packages turn out to be great tools for macro writers. In this section I list some tools I often use.

The package *ifpdf* (Oberdiek, 2006) provides the information whether the document is compiled via the DVI route or the direct PDF output is chosen. This is useful, for example, if the class uses a non-standard paper size (this is common for books). We can tell LaTeX the paper size, but to communicate it to e.g. *dvips*, we must instruct the user to put the corresponding option in the program call. If the direct PDF route is chosen, however, we can make the user's life slightly simpler with the commands like these (taken from the *memoir* code (Wilson, 2004)):

```
\ifpdf\relax
  \pdfpageheight=\paperheight
  \pdfpagewidth=\paperwidth
  \pdfvorigin=1in
  \pdfhorigin=1in
\fi
```

The package *geometry* (Umeki, 2008) is quite handy for easy setup of paper size and margins.

The package *fancyhdr* (van Oostrum, 2004) supports defining complex headers and footers, with or without decorative rules.

The package *caption* (Sommerfeldt, 2007) is useful for setting up captions for tables, figures and other floats.

Nowadays many designers prefer ragged layout of the copy. The package *ragged2e* (Schröder, 2003) helps to implement it in a rational way.

In many cases the "house style" includes specific rules for the bibliography. Then the *custom-bib* package (Daly, 2003) can help to create a customized BibTeX style according to the requirements.

There are many more tools that should be in a class writer's toolbox. A constant monitoring of Usenet groups and CTAN announce lists helps to keep abreast of the TeX development. However, there is always something to learn. When presenting this talk at TUG 2009, I complained that the LaTeX `\@addtoreset` command does not have a counterpart that removes some counters from the reset list. Barbara Beeton immediately recalled a small package (Carlisle, 1997) that provides a very useful command `\@removefromreset`.

Another matter related to the reuse of somebody else's code is whether to start a new class from scratch or to load a base class and then to redefine it as necessary. I guess this is a matter of taste. I myself prefer the second approach, but often it turns out in the end that the new class redefines so many macros that there is not much left from the original class.

5 These first pages ...

The cover, title and copyright pages are only a small fraction of the copy. Nevertheless they require a large percentage of a class author's effort. My experience shows that macros for these pages take about 60% of the total time for articles and 80% of the total time for books.

These pages require highly formal typesetting presenting structured information. This is easy to note in books (see the title and copyright pages for the *nostarch* class (Veytsman, 2008b)), but it is also true in reports: see the samples for the *erdc* class written for the reports of US Army Corps of Engineers (Veytsman, 2009).

For these macros we can dispense with the advice of Section 2 and change the standard LaTeX interfaces, since they are woefully inadequate, and most classes completely redesign front matter macros anyway. However there is one idea of front matter macros in standard LaTeX which is worth keeping. It is the idea of the separation of macros for *collecting data,* like `\author`, `\title`, etc. — and the macros

for typesetting like `\maketitle`. This approach allows entering the front matter data in any order, and to start typesetting only when we know which data are present and how large they are.

The commands for collecting data store them in internal macros. There are three kinds of such commands, which we discuss below.

The commands of the first kind are very simple. They have just one argument, and they store it in an internal macro. For example, the date in standard LaTeX can be defined as:

```
\def\date#1{\gdef\@date{#1}}
\date{\today}
```

In a more complex case the command has two arguments: the mandatory one and the optional one, with the usual convention that in the absence of the optional argument the mandatory one is used instead. As an example let us consider the `\title` command. In standard LaTeX it has just one argument, but we might want to have a full title for the title page and a short one for running heads, etc. This is a common enough case for books (Veytsman, 2008b), but can be found in articles if the journal uses article titles in headers (Veytsman, 2007; Veytsman, 2008a). With this command we can say something like

```
\title[Robinson Crusoe]{%
  The Life and strange Surprizing Adventures
  of Robinson Crusoe of York, Mariner:
  Who lived Eight and Twenty Years, all
  alone in an un-inhabited Island on the
  coast of America, near the Mouth of the
  Great River of Oroonoque; Having been cast
  on Shore by Shipwreck, where-in all the Men
  perished but himself. With An Account how
  he was at last as strangely deliver'd by
  Pyrates. Written by Himself}
```

(by the way, the long title is the actual title of Defoe's book!). This effect can be achieved by the `\@ifnextchar[` macro:

```
\def\title{\@ifnextchar[{%
    \title@i}{\title@ii}}
\def\title@i[#1]#2{%
  \gdef\@shorttitle{#1}\gdef\@title{#2}}
\def\title@ii#1{%
  \title@i[#1]{#1}}
```

The result is that the internal macro `\@title` gets the full title, while the macro `\@shorttitle` gets the short title.

The most complex case is *cumulative macros:* each command can be repeated several times, and the consecutive macros add to the stored information. Suppose for example that we have several groups of authors with shared affiliations. Than a natural syntax (following the ideas of American Mathematical Society classes, Downes and Beeton, 2004) to enter the information is the following:

```
\author{A.U.~Thor \and C.O.R.~Respondent
  \and C.O.~Author}
\affiliation{Construction Engineering
  Research Laboratory\\
  U.S. Army Engineer Research and
  Development Center\\
  2902 Newmark Drive\\
  Champaign, IL 61826-9005}
\author{John~M.~Smith}
\affiliation{Coastal and Hydraulics
  Laboratory\\
  U.S. Army Engineer Research and
  Development Center\\
  3909 Halls Ferry Road\\
  Vicksburg, MS 39180-6199}
```

This can be achieved in the following way:

```
\def\author#1{%
  \ifx\@empty\@authors
     \gdef\@authors{#1}%
  \else
    \g@addto@macro{\@authors}{\and#1}%
  \fi
  \ifx\@empty\@addresses
     \gdef\@addresses{\author{#1}}%
  \else
     \g@addto@macro{\@addresses}{%
       \par\author{#1}}%
  \fi}

\def\affiliation#1{%
  \ifx\@empty\@addresses
     \gdef\@addresses{#1\par}%
   \else
       \g@addto@macro{\@addresses}{%
         #1\par\vspace{\baselineskip}}%
   \fi}
```

As the result of these commands we have two internal macros: `\@authors` keeping the authors separated by `\and`, and `\@addresses` keeping the authors and their affiliations. Now we need to substitute `\and` inside them by the proper punctuation. Here the command `\andify` from Downes and Beeton, 2004, is very handy. By default it uses American punctuation (Tom, Dick, and Harry), but it has options for other variants. I used this possibility in packages *ijmart* and *erdc* (Veytsman, 2008a; Veytsman, 2009). The latter package provided an interesting challenge: it required the full list of authors without affiliations separated by commas and *and* on the cover, and separate lists of authors with shared affiliations on

the title page. Having a separate macro with just the authors' names helped to do this properly.

Another interesting thing to do is to automatically set up metadata for PDF output. Many publishers do not specify this in their requirements, but a class writer should know better. In any case this must be done when online publication is intended. As discussed above, we can determine whether *hyperref* is loaded and whether the PDF output is specified. If these conditions are true, then the construction of the title page should include lines like these:

```
\hypersetup{pdfauthor=\@author,%
  pdftitle=\@title,%
  pdfsubject=\@subject,%
  pdfkeywords=\@keywords}
```

6 Final remarks and conclusion

In the previous sections we discussed the computer-related aspects of class writing. There are, however, even more important human aspects.

It is essential to have a good interaction with the typographic designer and the users. A good typographer can explain what is required from the class and convey her or his ideas to the class writer. Some class writers try to double as designers; I always felt that the design requires years of training and apprenticeship. Just the fact that you can code complex things in TEX does not mean these things are beautiful or proper.

Good users can test the code and write sensible bug reports. This is also essential for the success of the class.

If you have good typographic artists and patient users, class writing is fun and rewarding. I personally enjoyed writing the macros for my LATEX classes.

Acknowledgements

The author is grateful to the users and typographic designers of his LATEX classes for their patience and tireless work, to the audience of TUG 2009 for many interesting suggestions.

References

User's Guide for the amsmath Package (Version 2.0). American Mathematical Society, 2002. http://mirror.ctan.org/macros/latex/required/amslatex/math/amsldoc.pdf.

LATEX 2ε for Class and Package Writers. LATEX3 Project, 2006. http://mirror.ctan.org/macros/latex/doc/clsguide.pdf.

Carlisle, David. *remreset Package*, 1997. http://mirror.ctan.org/macros/latex/contrib/carlisle.

Carlisle, D. P. *Packages in the 'Graphics' Bundle*, 2005. http://mirror.ctan.org/macros/latex/required/graphics.

Daly, Patrick W. *Customizing Bibliographic Style Files*, 2003. http://mirror.ctan.org/macros/latex/contrib/custom-bib.

Daly, Patrick W. *Natural Sciences Citations and References (Author-Year and Numerical Schemes)*, 2009. http://mirror.ctan.org/macros/latex/contrib/natbib.

Downes, Michael, and B. Beeton. *The amsart, amsproc, and amsbook document classes*. American Mathematical Society, 2004. http://mirror.ctan.org/macros/latex/required/amslatex/classes.

Eijkhout, Victor. *TEX by Topic*. Lulu, 2007. http://eijkhout.net/texbytopic/texbytopic.html.

Flynn, Peter. "Rolling Your Own Document Class: Using LATEX to Keep Away From the Dark Side". *TUGboat* **28**(1), 110–123, 2007. http://tug.org/TUGboat/Articles/tb28-1/tb88flynn.pdf.

Hefferon, Jim. "Minutes in Less Than Hours: Using LATEX Resources". *TUGboat* **26**(3), 188–192, 2005. http://tug.org/TUGboat/Articles/tb26-3/tb84heff.pdf.

Heinz, Carsten, and B. Moses. *The Listings Package*, 2007. http://mirror.ctan.org/macros/latex/contrib/listings.

Knuth, Donald Ervin. *The TEXbook*. Computers & Typesetting A. Addison-Wesley Publishing Company, Reading, Mass., 1994. Illustrations by Duane Bibby.

Mansfield, Niall. "How to Develop Your Own Document Class — Our Experience". *TUGboat* **29**(3), 356–361, 2008. http://tug.org/TUGboat/Articles/tb29-3/tb93mansfield.pdf.

Mittelbach, Frank, M. Goossens, J. Braams, D. Carlisle, and C. Rowley. *The LATEX Companion, Second Edition*. Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley Professional, Boston, 2004.

Oberdiek, Heiko. *The ifpdf Package*, 2006. http://mirror.ctan.org/macros/latex/contrib/oberdiek.

Pakin, Scott. "Good Things Come in Little Packages: An Introduction to Writing .ins and .dtx Files". *TUGboat* **29**(2), 305–314, 2008. http://tug.org/TUGboat/Articles/tb29-2/tb92pakin.pdf.

Rahtz, Sebastian, and H. Oberdiek. *Hypertext Marks in LaTeX: a Manual for Hyperref*, 2006. http://mirror.ctan.org/macros/latex/contrib/hyperref.

Raymond, Eric S., editor. *New Hacker's Dictionary*. The MIT Press, Boston, MA, third edition, 1996. http://catb.org/jargon.

Schröder, Martin. *The ragged2e Package*, 2003. http://mirror.ctan.org/macros/latex/contrib/ms.

Sommerfeldt, Axel. *Typesetting Captions with the caption Package*, 2007. http://mirror.ctan.org/macros/latex/contrib/caption.

Umeki, Hideo. *The geometry Package*, 2008. http://mirror.ctan.org/macros/latex/contrib/geometry.

van Oostrum, Piet. *Page Layout in LaTeX*, 2004. http://mirror.ctan.org/macros/latex/contrib/fancyhdr.

Veytsman, Boris. *Typesetting Articles for the Online Journal* Philosophers' Imprint, 2007. http://mirror.ctan.org/macros/latex/contrib/philosophersimprint.

Veytsman, Boris. *LaTeX Class for The Israel Journal of Mathematics*, 2008a. http://mirror.ctan.org/macros/latex/contrib/ijmart.

Veytsman, Boris. *LaTeX Style for* No Starch Press, 2008b. http://mirror.ctan.org/macros/latex/contrib/nostarch.

Veytsman, Boris. "Observations of a TeXnician for Hire". *TUGboat* **29**(3), 484, 2008c. http://tug.org/TUGboat/Articles/tb29-3/tb93abstracts.pdf.

Veytsman, Boris. *LaTeX Style for Technical Information Reports of the Engineer Research and Development Center, US Army Corps of Engineers*, 2009. http://mirror.ctan.org/macros/latex/contrib/usace.

Wilson, Peter. *The Memoir Class for Configurable Typesetting*, 2004. http://mirror.ctan.org/macros/latex/contrib/memoir.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net

LuaTeX for the LaTeX user: An introduction

Arthur Reutenauer

Abstract

LuaTeX, the TeX extension that incorporates the Lua scripting language and Ω extensions, has been available for three years already and was added to TeX Live in 2008. Yet, the LaTeX packages that have been developed for it are rather little known and, because the main developers of LuaTeX are deeply involved in ConTeXt, there seems to be a gap between LuaTeX and LaTeX users. This note is a beginning at overcoming this difference.

1 Introduction

LuaTeX is the extension of TeX that incorporates Lua, as well as the capabilities of Ω. It has also been programmed with hooks to interact with the internals of TeX's algorithms (*callbacks* in Lua parlance), that can now be rewritten using Lua code. However, most of the power of LuaTeX resides at a very low level and is available to the user only with the help of macros, that have been written for ConTeXt over the past several years, and only recently begun for LaTeX. I will give a brief descriptions of some of these LaTeX packages.

All the packages mentioned here are available in TeX Live 2009.

2 `\ifluatex`

The package ifluatex provides the conditional command `\ifluatex` to test whether we're running LuaTeX or some other TeX engine.

3 Input encodings

LuaTeX reads source files in Unicode UTF-8 encoding, and it is highly recommended to use that encoding in order to take full advantage of the capabilities of LuaTeX. However, for people who want or need to use 8-bit encodings like Latin 1 or KOI-8, the luainputenc package emulates the behaviour of the standard inputenc in LuaTeX. It recognizes all the encodings that the latter package knows about and can therefore be used for legacy documents, with some limitations.

4 Fonts

One of the most elaborate packages for use with LuaLaTeX is luaotfload.

In this section, I will use the free Linux Libertine fonts (`http://linuxlibertine.sourceforge.net/`) as an example; these fonts have many features that demonstrate OpenType capabilities. I will assume that the font files are available to TeX and that the four faces (roman, italic, bold and bold italic) are respectively called `LinLib-Re.otf`, `LinLib-It.otf`, `LinLib-Bd.otf`, and `LinLib-BI.otf`.

luaotfload implements a XeTeX-like syntax for `\font`, hence a commands such as

```
\font\librm="LinLib-Re"
```

will load the font, and subsequently issuing the `\librm` control sequence will switch to it. This example:

```
\font\librmliga
       ="LinLib-Re:script=latn:+liga"
```

will load the same font while activating the liga feature of the Latin script.

Other useful OpenType feature names include

- smcp, for small capitals;
- onum, for old-style numerals;
- subs, for subscript;
- sups, for superscript.

Here is a complete NFSS declaration of the Linux Libertine family for LaTeX:

```
\DeclareFontFamily{T1}{libertine}{}
\DeclareFontShape{T1}{libertine}{m}{n}
  {<-> "LinLib-Re:+liga"}{}
\DeclareFontShape{T1}{libertine}{m}{sc}
  {<-> "LinLib-Re:+liga:+smcp"}{}
\DeclareFontShape{T1}{libertine}{m}{it}
  {<-> "LinLib-It:+liga"}{}
\DeclareFontShape{T1}{libertine}{bx}{n}
  {<-> "LinLib-Bd:+liga"}{}
\DeclareFontShape{T1}{libertine}{bx}{it}
  {<-> "LinLib-BI:+liga"}{}
```

Then, after defining `\libertine` as follows:

```
\newcommand\libertine
  {\fontfamily{libertine}\selectfont}
```

we can use `\libertine` to switch to the Libertine family.

More examples of using LaTeX with LuaTeX are gathered on `http://code.google.com/p/mingyue`.

◇ Arthur Reutenauer
GUTenberg, France
arthur dot reutenauer (at)
normalesup dot org

Ongoing efforts to generate "tagged PDF" using pdfTeX

Ross Moore

Mathematics Department, Macquarie University, Sydney, Australia
ross@maths.mq.edu.au

Abstract. Recently PDF has been accepted as a standard for production of electronic documents, as ISO 32000-1:2008, with an acronym of PDF/UA (for "Universal Accessibility"). The second draft ISO 32000-2:2009 is to include specifications for including MathML tagging of mathematical environments and expressions. This talk presents a report on work-in-progress, aimed at:

(a) developing the primitive commands for pdfTeX needed to support the production of fully tagged PDF documents;
(b) writing appropriate TeX and LaTeX macros to make effective use of the new primitives;
(c) authoring changes to internal LaTeX structures to use these macros automatically at appropriate places within the existing code-base for LaTeX.

This is work that is being undertaken together with Hàn Thế Thành, author of pdfTeX [2], who has added some new primitive commands to an experimental version of this software tool.

1 Background

In July 2008, Adobe's PDF Reference 1.7 [1] became ISO 32000 [4]. Since 2005, the PDF Reference 1.4 has served as the basis for ISO 19005 [3], as an archival format for technical documents. Both of these standards rely heavily on "Tagged PDF", so that not only is the content displayed at the highest quality, but also its structure is provided, allowing for selective extraction of content and "reflow" on small-screen devices (such as a PDA or modern mobile-phone), and screen-reading perhaps in alternate languages. Work is under way on revision of ISO 19005, called PDF/A-2, to accommodate extra features introduced with PDF 1.5, 1.6 and 1.7. Furthermore, in November 2008 it was agreed that a revised ISO 32000-2 should include tagging of the structure of mathematical expressions and formulae, using MathML tags. It may take as long as 2–3 years before these updated standards are released in their final form.

TeX and LaTeX remain *de facto* standards for technical documents, particularly those having a large amount of mathematical content, though other methods are starting to gain significant usage. Whilst PDF is the main output

Reprinted with permission from Ross Moore: Ongoing Efforts to Generate "Tagged PDF" using pdfTeX. In: Petr Sojka (ed.): DML 2009, Towards a Digital Mathematics Library, Masaryk University Press, pages 125–131, ISBN 978-80-210-4781-5.

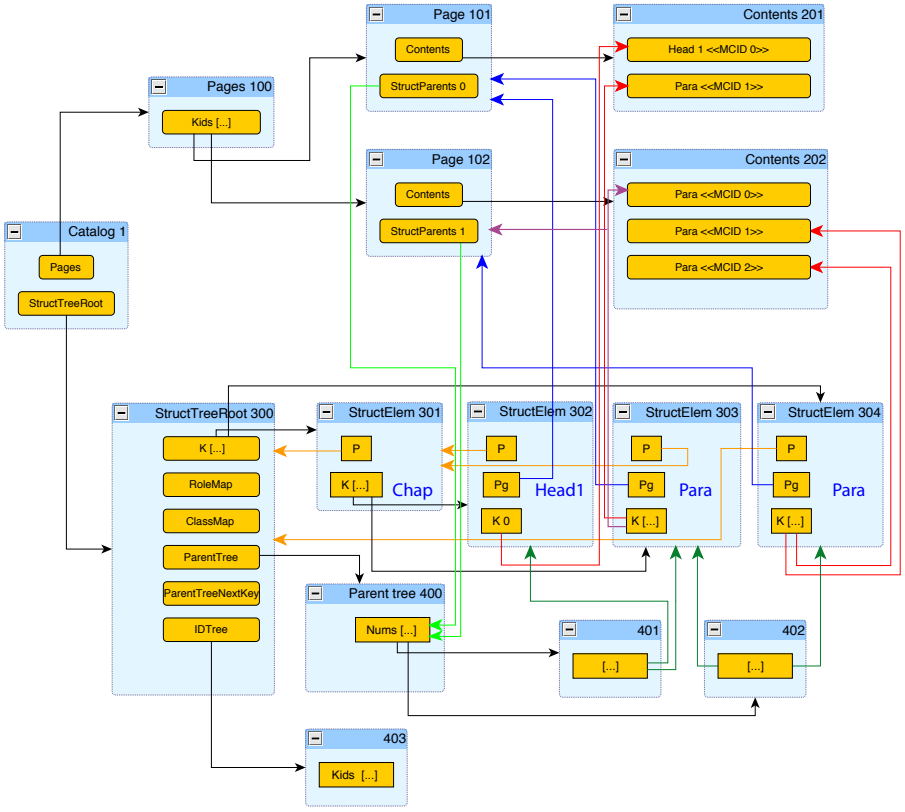


Fig. 1. Interleaving of structure and content tagging within a 2-page PDF document, structured as a heading and two paragraphs. (based on an example in [1])

format for LaTeX, there have been no automated methods to include the structure and content tagging that the above standards require. Work being undertaken here is aimed at providing this missing support when pdfTeX [2] is used as the PDF-producing software. As well as requiring new *primitive* commands to mark content and build the structure trees that are needed for this tagging, a large amount of the LaTeX codebase will need to be revised to take advantage of the new features. In this paper and associated talk, examples are shown of work done so far, towards this aim.

Section 2 gives an idea of how tagging in PDF works, indicating the complexity of the extra structures that PDF-producing software needs to provide; Figure 1 gives a schematic view of these structures, within a document having a quite simple structure. On the other hand, MathML tagging generally requires a much deeper structure tree. In Section 3 an example (see Figure 2) is presented, showing how the MathML tagging is represented, using new TeX primitives and within the PDF, such that it can then be faithfully exported to XML.

2 Tagging PDF documents

Figure 1 indicates the extra structures that need to be created when producing "Tagged PDF". The upper half of the image shows the kinds of object that are needed to display a PDF file as a series of pages. These kinds of objects include:

(i) page content streams, which consist of the low-level commands to select fonts and place text on the page — the blue boxes headed as 'Contents ...';
(ii) an indexing object for each page — the blue boxes headed as 'Page ...';
(iii) an indexing object, headed 'Pages' that acts as a parent for the collection of 'Page ...' objects;
(iv) the previous object is a child of the 'Catalog', which is the root node for the complete document structure.

With tagged PDF there is also a *Structure Tree* whose root node 'StructTreeRoot' is another child of the 'Catalog'. This is itself the root node for a tree of objects headed as 'StructElem ...', which describe the abstract structure of the textual content of the document. Each 'StructElem' node has both references to its children, and a back-pointer to its parent node within the Structure Tree.

To define the content that is encompassed within the structure, there need to be references from the nodes of the Structure Tree to specific locations within the 'Contents' streams. The locations are indicated by the (round) rectangles, with arrows indicating how the structure relates to these. Extra arrows point from structure nodes to 'Page' nodes, which help identify where the content can be seen; that is, on which page does it (mostly) occur.

A second tree is linked-to from the 'StructTreeRoot'; this is called the 'ParentTree', containing a node for each physical page. These nodes are each an array of references to all the 'StructElem' nodes that have content on the corresponding page. There is a link from each 'Page ...' object to the 'ParentTree', which allows the corresponding node to be easily located.

Finally a third 'IDTree' is an optional feature. Each 'StructElem' node can be given a unique name. The 'IDTree' acts as the root node for a tree built up to include arrays of these names, each paired with a pointer to the corresponding 'StructElem' node. This possibility of associating names to structure is for the benefit of Application software that produces or manipulates PDF files. It can use whatever naming scheme it likes to facilitate access to the kinds of structured objects that it needs to work with.

Figure 1 is based on an example in the PDF Reference document [1]. The structure it represents consists of a document section (chapter) having a heading and a paragraph stretching across two pages, together with another paragraph.

3 MathML tagging within a PDF document

Figure 2 shows the effect of having a piece of mathematics tagged (using MathML syntax) within a PDF document. The middle part of the image shows how the page would appear within an Adobe Reader, or (in this case) Acrobat, browser.

This view is partly obscured by Acrobat's 'Order Panel', which displays the tagging of a mathematical expression, with an `<mrow>` selected. The corresponding content is highlighted with rectangles back in the browser view. On the left side we see the result of an 'Export to XML 1.0' action, writing the tagged contents out into a text file. This export has included the mathematical symbols using UTF8 format, so the correct Unicode Plane 1 "Mathematical Alphanumerics" are shown within a text editor that supports the full Unicode range.

This example was produced using an experimental version of pdfTEX. The fonts being used are from the Computer Modern family, which are the standard fonts that have traditionally been used with TEX and LATEX. Mappings to Unicode Plane 1 characters are achieved using the LATEX package mmap.sty, described within a recent *TUGboat* article [5].

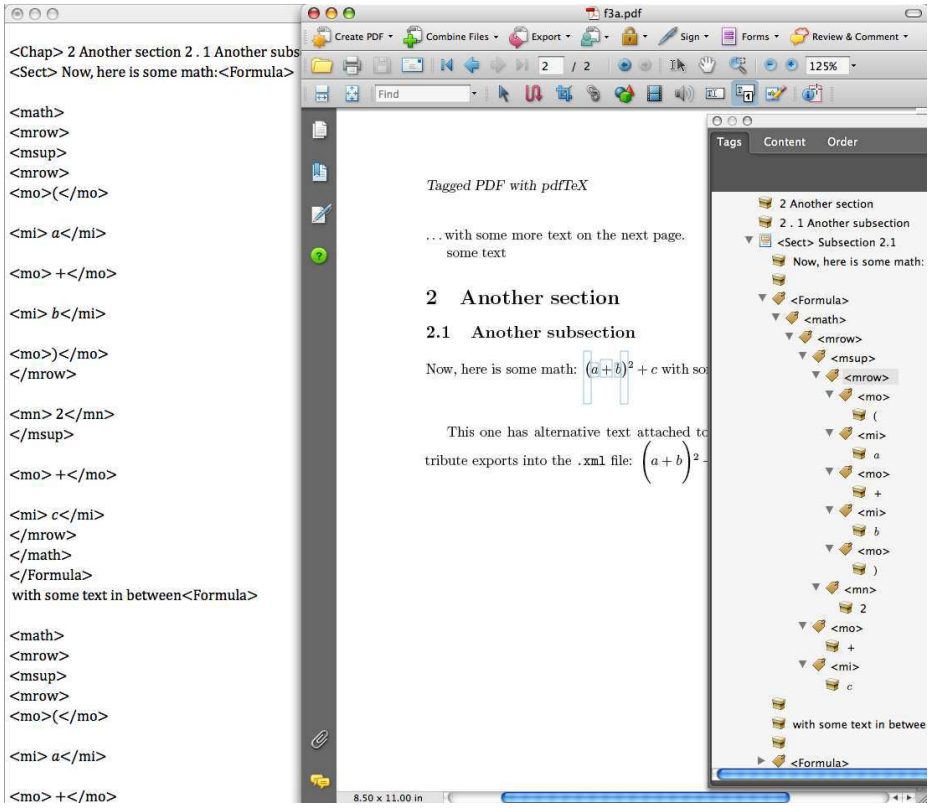


Fig. 2. MathML tagging within a PDF document

The LATEX coding below shows part of what was used to produce the tagging of mathematics shown in Figure 2. It shows how to use new primitive commands `\pdfstructelem`, `\pdfstartmarkedcontent` and `\pdfendmarkedcontent`.

```
\pdfstructelem attr{/S /Formula} 3 27
\pdfstructelem attr{/S /math} 27 28
\pdfstructelem attr{/S /mrow} 28 29
 \pdfstructelem attr{/S /msup} 29 30
  \pdfstructelem attr{/S /mrow} 30 31
   \pdfstructelem attr{/S /mo} 31 32
    \pdfstartmarkedcontent attr{/ActualText(\050)
      /Alt(, open bracket, )} 32 {mo}\biggl( \pdfendmarkedcontent
   \pdfstructelem attr{/S /mi} 31 33
    \pdfstartmarkedcontent attr{/Alt( alpha )}noendtext
       33 {mi}\alpha \pdfendmarkedcontent
   \pdfstructelem attr{/S /mo} 31 34
    \pdfstartmarkedcontent attr{/Alt( plus )}noendtext
       34 {mo}+ \pdfendmarkedcontent
    ...
```

The primitive `\pdfstructelem` requires two numbers specifying a unique identifier for the structure node being created, preceded by the identifier of its parent node, and attributes including the type of tag. Leaf nodes, constructed with `\pdfstartmarkedcontent`, require the identifier of the parent structure node. Their attributes can include `/Alt` text to be read by a screen-reader, and an `/ActualText` alternative for text-extraction. The kind of node for a mathematical symbol agrees with its parent structure node, (e.g., `/mi`, `/mo` or `/mn`). This is followed by the TEX coding to produce a visual representation, terminated by `\pdfendmarkedcontent`. Part of the PDF content stream resulting from this coding is given below, showing how the tagging is interspersed with positioning and font-changing commands, and the font characters themselves.

```
1 0 0 1 70.69 -23.949 cm
/mo <</MCID 15  /ActualText(\050) /Alt(, open bracket, )>>BDC
1 0 0 1 0 17.036 cm BT
/F1 9.9626 Tf/F18 1 Tf( )Tj/F1 9.9626 Tf [(\040)]TJ ET EMC
1 0 0 1 7.887 -17.036 cm
/mi <</MCID 16  /Alt( alpha )>>BDC BT
/F11 9.9626 Tf/F18 1 Tf( )Tj/F11 9.9626 Tf [(\013)]TJ ET EMC
1 0 0 1 6.41 0 cm
/mo <</MCID 17  /Alt( plus )>>BDC
1 0 0 1 2.214 0 cm BT
/F8 9.9626 Tf/F18 1 Tf( )Tj/F8 9.9626 Tf [(+)]TJ ET EMC
```

This kind of coding, directly in pdfTEX primitives, is really only useful for testing and "proof of concept" examples, such as Figure 2. Any mistake in the numerical identifiers can result in a broken PDF that may appear to render properly, but nevertheless crashes Acrobat due to a malformed structure tree.

Handling those numerical identifiers and parent relationships is something better done using an extra layer of LATEX macros, as in the coding example below. A `\taginlinemath` macro sets up an enclosing `/Formula` structure tag. Presentation MathML structure is specified using `\tagmathbranch`. MathML content tags are associated with TEX source using `\tagmathbleaf`, which has

an optional argument for spoken text. A variant `\tagmathaleaf` accommodates `/ActualText` replacements for large delimiters and extended constructions which require more than one glyph to display a single symbol.

```
\taginlinemath{%
  \tagmathbranch{msup}{\storePDFparentID
   \tagmathbranch{mrow}{%
    \tagmathaleaf[, open bracket, ]{mo}{/stretchy /false
      /minsize(1.2em) /maxsize(1.2em)}{\050}{\bigl(}%
    \tagmathbleaf[ alpha ]{mi}{}{\alpha}%
    \tagmathbleaf[ plus ]{mo}{}{+}%
    \tagmathbleaf[ beta ]{mi}{}{\beta}%
    \tagmathaleaf[, close bracket, ]{mo}{/stretchy /false
       /minsize(1.2em) /maxsize(1.2em)}{\051}{\bigr)^{%
      \adjustendcontent \tagmathbleaf[ all squared, ]{mn}{}{2}%
     }%end of ^
    }% </mo>
   }% </mrow>
  }% </msup>
  ...
```

In the above examples, the MathML tagging has been coded by hand to get working LaTeX source. Ultimately such markup, that interweaves MathML tagging with TeX code, needs to be generated automatically. This will require new coding structures called from modified expansions for existing LaTeX internal commands and environments (as used with paragraphs, headings, etc.), as well as with mathematical environments. For math the proposed strategy is to write the LaTeX source of a complete environment to disk, run a 3rd-party MathML converter to generate the tagging, then read the result back into the running job, merging the two coded views of the same piece of mathematics. Any external MathML converter could be used, provided it can be run as a command-line program using TeX's `\write18` facility. Alternatively, if a MathML version is already available for a piece of LaTeX source, then this could be used instead.

References

1. Adobe Systems Inc.; PDF Reference 1.7, November 2006. `http://www.adobe.com/devnet/pdf/pdf_reference.html`
2. Hàn Thế Thành; Thesis — pdfTeX, published as: *TUGboat*, 21:4, (2000). `http://www.tug.org/TUGboat/Contents/contents21-4.html`
3. ISO 19005-1:2005; Document Management — Electronic document file format for long term preservation — Part 1: Use of PDF 1.4 (PDF/A-1). `http://www.iso.org/iso/catalogue_detail?csnumber=38920`
4. ISO/DIS 32000; Document management — Portable document format (PDF 1.7), July 2008. `http://www.iso.org/iso/catalogue_detail?csnumber=51502`
5. Moore, Ross R.; Advanced features for publishing mathematics, in PDF and on the Web. *TUGboat*, 29:3, (2008), pp. 464–473. `http://www.tug.org/TUGboat/Contents/contents29-3.html`
6. PDF/UA Universal Accessibility; websites at `http://pdf.editme.com/pdfua` and `http://www.aiim.org/Standards/article.aspx?ID=27861`.

The EduTEX TUG working group

Frank Quinn

Abstract

This note introduces the EduTEX working group and sketches goals for the startup period.

Introduction

The EduTEX working group is concerned with extended usage of TEX designed to support student learning. We have a wiki at `http://edutex-wiki.tug.org` and a mailing list (see `http://lists.tug.org/edutex`). This note is a brief introduction; see the wiki for more detail and an explanation (if you need one) of why TEX is the best setting for this.

Learning environments

The core objective is a setting for producing learning environments that masquerade as tests. These objects *are* tests in the sense that they are used for assessment, but the main reason for presenting them this way is to provide motivation, and lure students into an environment where they can be given feedback and easy access to resources.

Startup

Initial development of EduTEX is constrained by objectives specified in the grant from the National Science Foundation. So, our concerns for now:

- a focus on mathematics;
- creating educationally effective functionality;
- avoiding exciting but ineffective functionality.

Rigid educational constraints are out of place in a software user group. After startup (i.e., when the funding runs out) we will separate the software and educational threads. We anticipate a continued rich interaction between the two, but the software thread will follow its own interests with more freedom.

Clever new machines, clever new software, same old brains

This is the guiding thought for my approach to education. The point is that human learning uses primitive (or at least undeveloped) facilities and is haphazard and highly variable. Thus:

- Software design driven by needs of human children often seems dull and boring from a technology point of view.
- Software that seems exciting and powerful to experienced users rarely connects with the needs of real-life learners, and is often counterproductive.

Two examples:

Video and animations In the Math Emporium at Virginia Tech (`http://www.emporium.vt.edu`) we have found that video clips and animations almost always put students into spectator mode. They may be interested, amused, even "engaged" while in this mode, but are essentially incapable of real learning.

We surmise this is because they have had much more practice being spectators than learners. Further, the entertainment industry has taught them to suspend critical facilities, and the advertising industry has taught them to resist learning while being entertained.

Distraction Modern students tend to have short attention spans. Distractions such as pictures, cute illustrations, unnecessary links, anything that moves, and vague or unfocused discussion, tend to break attention focus and impede learning. Effective learning environments tend, therefore, to be drab and boring to those who are not actually learning.

For these reasons the startup period will focus on core functionality. Once the core is solidly in place we can be more adventurous.

Contribute?

Comments, suggestions, pointers to useful materials, and any other contributions are very welcome. In particular, significant sub-projects that need fleshing out are on the 'Functionality' page of the wiki.

◇ Frank Quinn
Virgina Tech
`http://edutex-wiki.tug.org`

Becoming a CTAN mirror

Jim Hefferon

Have you downloaded something from CTAN that you found useful? Would you like to be useful in return?

Sites on the Comprehensive TEX Archive Network come in two types. There are three core sites, where every day the archivists place new materials in the file tree. The other sites are mirrors — every night they sync their holdings with a core site and then offer those materials for download by others.

A mirror site, once set up, mostly runs itself.

So, if you are able to run a mirror site, that would be a big help to CTAN and to TEX users near you, without causing too much trouble for you. This note describes the technical requirements for doing this, with the specific steps to become an official CTAN mirror.

1 Requirements

You need a computer with 20–30 gigabytes of available disk space and an Internet connection with a permanent IP address. If you have a firewall then you must make sure external users can get through to your site. And, you also need some experience running a Unix-like system, such as GNU/Linux or Mac OS X (a Windows-based system could work but the directions below don't cover this case).

2 You are only three steps away

To set up a mirror you need to (1) give visitors access to files by running either a web or FTP daemon, or both, (2) set up your system to sync your holdings with a core site every day, and (3) sign up to be an official mirror.

2.1 Give your visitors access

The two most popular way to offer the files to your visitors are over HTTP and over FTP. The examples below assume that you keep the archive materials in your `/var/ftp/pub/tex-archive` directory.

To offer the materials over HTTP you need a web server. We use Apache.[1] Setting up a server is beyond our scope but here are tips.

- One way to make the archive available is to put a link inside your document root. For example, this command (adjust paths as needed) makes `http://`*your site*`/tex-archive` show the top of the archive.

```
ln -s /var/ftp/pub/tex-archive \
      /var/www/html/tex-archive
```

- You don't want files named `index.html` to be served as the index of its directory. See the directives in Figure 1.

To offer materials over FTP, you must have an FTP daemon running. We use ProFTPD[2] but there are many others. Setting up this daemon is also beyond our scope but if your documentation does not cover how to allow anonymous access then we suggest choosing different software.

2.2 Sync with a core site

To get the materials, and to keep them up to date, run `rsync`.[3] This program does the transfers efficently, saving both you and us network usage.

You must mirror from one of the core sites. Pick the one nearest you. Use `rsync://rsync.cam.ctan.org/CTAN` for the one in England, or `rsync://rsync.dante.ctan.org/CTAN` for the German site, or `rsync://rsync.tug.ctan.org/CTAN` for the one in the northeast USA. (The examples below use the first.)

The following command will get everything on CTAN and put it on your hard drive.

```
rsync -av --delete \
      rsync://rsync.cam.ctan.org/CTAN \
      /var/ftp/pub/tex-archive
```

The first time that you run it, this command can take quite a long time (an hour or more, perhaps, depending on the connection speed). Before you commit to it, you can try using the `-n` option, as in `rsync -avn --delete ...`, which will say what would be done without doing it.

You must run the `rsync` command every day. To set this up, run `crontab -e` at the command prompt. In the editor that appears enter a line like this (the line breaks are only for *TUGboat*; it must be one line in the editor):

```
31 2 * * * rsync -a --delete
  rsync://rsync.cam.ctan.org/CTAN
  /var/ftp/pub/tex-archive
```

This makes your system run the `rsync` command at 31 minutes past the hour, of the second hour of the day, on every day of the month, and during every month of the year, and every day of the week, that is, Sunday thru Saturday.

Please change these numbers when you set your system up, so that not everyone in the world hits CTAN at the same instant. Pick a time that is in the middle of the night at the location of the core site that you are mirroring.

[1] `http://httpd.apache.org`

[2] `http://www.proftpd.org/`

[3] `http://rsync.samba.org`

```
<Directory />  # web visitors can't see outside web tree
  Order Deny, Allow
  Deny from all
  </Directory>

<Directory /var/www/html>  # web visitors can see the web tree
  Order Allow, Deny
  Allow from all
  Options +FollowSymLinks
  </Directory>

<Directory /var/www/html/tex-archive> # soft link to CTAN tree
  Order Allow, Deny
  Allow from all
  Options -ExecCGI, +FollowSymLinks, -Includes, -IncludesNOEXEC, +Indexes
  DirectoryIndex      # no value, so 'index.html' is not used
  </Directory>
```

Figure 1: Part of your web configuration file

2.3 Become official

When you have the files, and the cron job, and you are offering public access, visit `http://www.ctan.org/mirror_signup` to sign up to become an official mirror.

After a day or two, you will be added to a mailing list that is very low volume, perhaps a half dozen messages a year. And, you will be put into the CTAN system, so that you will start seeing traffic.

3 Thanks

If you can run a mirror, then we at CTAN would appreciate it. It is a much needed service — we could not continue without our mirrors.

◇ Jim Hefferon
Saint Michael's College
Colchester, Vermont USA
ftpmaint (at) tug dot ctan dot org

TEX Live 2009 news

Karl Berry

Abstract

Notable changes in the TEX Live 2009 release.

Introduction

TEX Live is a cross-platform TEX distribution supported by the TEX user groups worldwide. Our native distribution provides its own installer and management tools; it also serves as the basis for the TEX packages in most GNU/Linux, BSD-related, and other system distributions. It was first released in 1996 by Sebastian Rahtz, based on Thomas Esser's teTEX. By the time this note reaches print, we hope the 2009 release will be in final production.

News for 2009

In TEX Live 2009, the most visible change is that pdf(LA)TEX now *automatically* converts a requested Encapsulated PostScript (EPS) file to PDF, via the `epstopdf` package, when and if the LATEX configuration file `graphics.cfg` is loaded, and PDF is being output. The default options are intended to eliminate any chance of hand-created PDF files being overwritten, but you can also prevent `epstopdf` from being loaded at all by putting `\newcommand {\DoNotLoadEpstopdf}{}` (or `\def...`) before the `\documentclass` declaration. For details, see the package documentation for epstopdf (`http://ctan.org/pkg/epstopdf-pkg`).

A related important change is that execution of a few external commands, via the `\write18` feature, is now enabled by default — for example, `epstopdf`, `makeindex`, and `bibtex`. The exact list of commands is defined in the `texmf.cnf` file. Installations which must disallow all such external commands can deselect this option in the TEX Live installer, or override the value in `texmf.cnf` after installation.

The default output format for Lua(LA)TEX is now PDF, to take advantage of LuaTEX's OpenType support, et al. New executables named `dviluatex` and `dvilualatex` run LuaTEX with DVI output. The LuaTEX home page is `http://luatex.org`.

The original Omega engine and Lambda format have been excised, after discussions with the Omega authors. The updated Aleph and Lamed remain, as do the Omega utilities.

A new release of the `amsfonts` package is included, including updates to the Computer Modern fonts in Type 1: a few shape changes made over the years by Knuth in the Metafont sources have been integrated, and the hinting has been updated. The Euler math fonts have been thoroughly reshaped by Hermann Zapf (see `http://tug.org/TUGboat/Articles/tb29-2/tb92hagen-euler.pdf`). In all cases, the metrics are unchanged. The AMS fonts home page: `http://ams.org/tex/amsfonts.html`.

The new GUI front end TEXworks is included for Windows, and is also in MacTEX. For other platforms, and more information, see its home page, `http://tug.org/texworks`. TEXworks is a cross-platform front end aiming at initial ease-of-use, inspired by the TeXShop editor for Mac OS X.

The graphics program Asymptote is included for most platforms. This implements a text-based graphics description language somewhat akin to MetaPost, but with advanced 3D support and other features. The Asymptote home page, with documentation and more, is `http://asymptote.sourceforge.net`.

The separate `dvipdfm` program (no longer maintained) has been replaced by `dvipdfmx`, which operates in a special compatibility mode under that name. `dvipdfmx` includes CJK support and has accumulated many other fixes over the years since the last `dvipdfm` release. The DVIPDFMx home page is `http://project.ktug.or.kr/dvipdfmx`.

Executables for the `cygwin` and `i386-netbsd` platforms are now included, while the other BSD distributions have been dropped; we were advised that OpenBSD and FreeBSD users get TEX through their package systems, plus there were difficulties in making binaries that had a chance of working on more than one version.

Lastly, a miscellany of smaller changes: we now use `xz` compression, the stable replacement for `lzma` format (`http://tukaani.org/xz/`); a literal `$` is allowed in filenames (when it does not introduce a known variable name); the Kpathsea library is now multi-threaded (made use of in MetaPost); the entire TEX Live build is now based on Automake.

In summary

Thanks to all the many people involved in this and past releases. See the documentation for acknowledgements, as well as installation and usage information — `http://tug.org/texlive/doc.html`.

All releases of TEX Live, along with ancillary material such as CD labels, are available at `ftp://tug.org/historic/systems/texlive`.

TEX Live and MiKTEX account for a substantial portion of the total traffic to CTAN. So more mirrors are always welcome; if you can help, please see `http://ctan.org/mirroring.html`, and thanks.

◇ Karl Berry
`http://tug.org/texlive/`

Getting started with plasTEX

Tim Arnold

Abstract

The software package plasTEX converts LATEX documents to other markup languages. This article describes typical usage with examples of how to create HTML and DocBook XML from LATEX sources, an overview of how to extend the package to handle custom commands and environments, and a demonstration of converting a simple LATEX source file.

1 Introduction

1.1 What plasTEX does

plasTEX is a document-processing framework that converts LATEX sources to HTML, XML, and other markup formats. The author uses plasTEX as part of a publishing workflow to produce statistical documentation at SAS Institute Inc.

The software is released as free and open source, under the MIT license. It is written in Python by Kevin Smith of SAS.

1.2 Math support

plasTEX is designed to use plug-ins for math support. You select the format of the math output by specifying the plug-in that plasTEX should use to render the document.

This plug-in design keeps plasTEX flexible so it can produce a mathematical format that is most appropriate for the current state of browser capabilities. Once browsers fully support MathML and their display is production quality, any software program that produces MathML from LATEX math can be used as the engine.

The default plug-in is `dvipng`, which replaces mathematics with images in the form of PNG formatted bitmap graphics. Optionally, you can use the `dvisvg` plug-in, which produces mathematics in SVG vector graphics.

2 How it works

2.1 Interface

plasTEX provides a command line interface. The interface includes options that enable you to:

- specify themes and navigational elements. A theme is a special template that sets the look and feel of the header, footer, and navigation elements such as breadcrumbs. plasTEX comes with several themes. You can specify a particular theme to match each output format (for example, HTMLHelp, EclipseHelp, and JavaHelp).
- specify input and output encoding. Different input encoding provides options for writers. Different output encoding enables matching the output to the destination; for example, HTMLHelp must be encoded as Windows-1252.
- set specific counters. It is useful to be able to process extremely large books one chapter at a time. Using the command line option, you can set the chapter counter for each chapter.
- set the depth of the table of contents and section numbering.
- specify image generation engine. This allows for PNG or SVG format mathematics and enables flexibility to use any future external process for handling LATEX math.

2.2 Internals

There are two steps in plasTEX processing:

1. plasTEX parses the LATEX source to create a *document object model* (DOM), which is a nested, tree-like data structure that contains the content plus the document data elements for the commands, environments, and their arguments.
2. plasTEX renders the DOM to the output format by combining a set of templates with the document data.

We now describe these steps in more detail.

2.2.1 Create the document object model

For every built-in LATEX command and environment, plasTEX provides a corresponding Python class so it can recognize and digest the tokens as they are encountered in a document.

As an example, plasTEX implements the LATEX `\framebox` command:

```
\framebox[width][position]{text}
```

with the following snippet of Python code, which defines the `framebox` class:

```
class framebox(TextBoxCommand):
    args = '[ width ] [ pos ] self'
```

The `framebox` class inherits from the internal plasTEX base class `TextBoxCommand` and maps the arguments to variables. During parsing, plasTEX reads the `framebox` command according to its definition and saves the resulting named tokens (values of the variables) in the document data structure. These variables are used in the next step, rendering.

2.2.2 Render the document

plasTEX rendering uses templates (a popular methodology in web publishing frameworks). A data structure (DOM) and a set of templates are combined to automatically generate documents.

Suppose you have a `framebox` command in your document:

```
\framebox[15em]{Note: this is an example}
```

LaTeX renders the markup as follows:

Note: this is an example

When plasTeX encounters the markup, it parses the `\framebox` into its constituent data and finally renders that data to HTML with the following template:

```
name: framebox
<span tal:attributes="style
  string:width:${self/attributes/width}"
  tal:content="self"></span>
```

The final output appears as follows:

```
<span style="width:15em">
Note: this is an example</span>
```

2.2.3 Summary of plasTeX processing

plasTeX parses each of the LaTeX commands and text in the source document to create the DOM. Parsing occurs one time for each input document.

The plasTeX parser must recognize every command and environment encountered in the document. plasTeX understands built-in LaTeX and TeX commands in addition to the following packages:

a4wide	alltt	amsart	amsfonts
amsmath	amssymb	amsthm	babel
beamer	changebar	color	comment
fancybox	fancyhdr	fancyvrb	float
fontenc	geometry	graphics	graphicx
hyperref	ifpdf	ifthen	inputenc
keyval	lipsum	longtable	makeidx
minitoc	natbib	pslatex	rotating
shortvrb	subfig	subfigure	textcomp
times	ucs	url	verbatim
wrapfig			

To create the output, plasTeX applies a set of corresponding templates for each document element that it encountered. The rendering step can be done multiple times using different sets of templates to create different output formats from a single DOM.

2.3 Rendering formats

plasTeX is bundled with the following template sets, i.e., it can render documents into these formats:

- HTML
- well-formed XML (the internal representation of a document within plasTeX is well-formed XML, which uses the plasTeX namespace)
- DocBook 4.5 and 5.0 XML
- plain text (useful to "detex" documents since all tagging is removed except for math)
- ePub, an emerging e-book standard format based on XHTML
- S5, a simple standards-based slide show system
- BrlTeX, an open source LaTeX-to-braille translator that is designed to handle math

2.4 Extending plasTeX

plasTeX can be extended in two ways:

- You can add new commands or environments that plasTeX can understand and parse by creating a corresponding Python class. The new class enables the plasTeX parser to tokenize documents that contain the new markup. This can often be a simple subclass of an existing class.
- You can add a new type of output format by adding a set of corresponding templates so that the plasTeX renderer can produce the appropriate output for each element.

3 Examples & demonstrations

The presentation of this paper included a live demonstration of converting LaTeX documents. To view the demonstration, install the plasTeX distribution and give the appropriate command, as follows:

- `plastex sample2e.tex`
  renders the familiar `sample2e.tex` example file into HTML. You can view the contents with any browser; open the file `sample2e/index.html`. See Figure 1.
- `plastex --renderer Text \`
  `--split-level 0 sample2e.tex`
  renders the same source file into a single file of plain text, leaving the math as it was entered in the document.

Also during the presentation, two research articles in LaTeX format were downloaded from the website `ArXiv.org` and rendered to HTML. (Two small bugs were noted by audience members. The bugs have been fixed in the current version of the plasTeX distribution.)

As a larger example, the *Python Library Reference* (a substantial documentation package written in LaTeX) was converted to HTML with no errors in about 45 minutes time.

In short, give plasTeX a try on your favorite documents; it can handle a lot.

4 Summary

In conclusion, plasTeX provides an easy-to-use command line interface with which to convert LaTeX sources to a variety of output formats. The framework can easily be extended by adding Python classes or templates, making it a useful tool for both simple documents as well as production-quality publishing workflows.

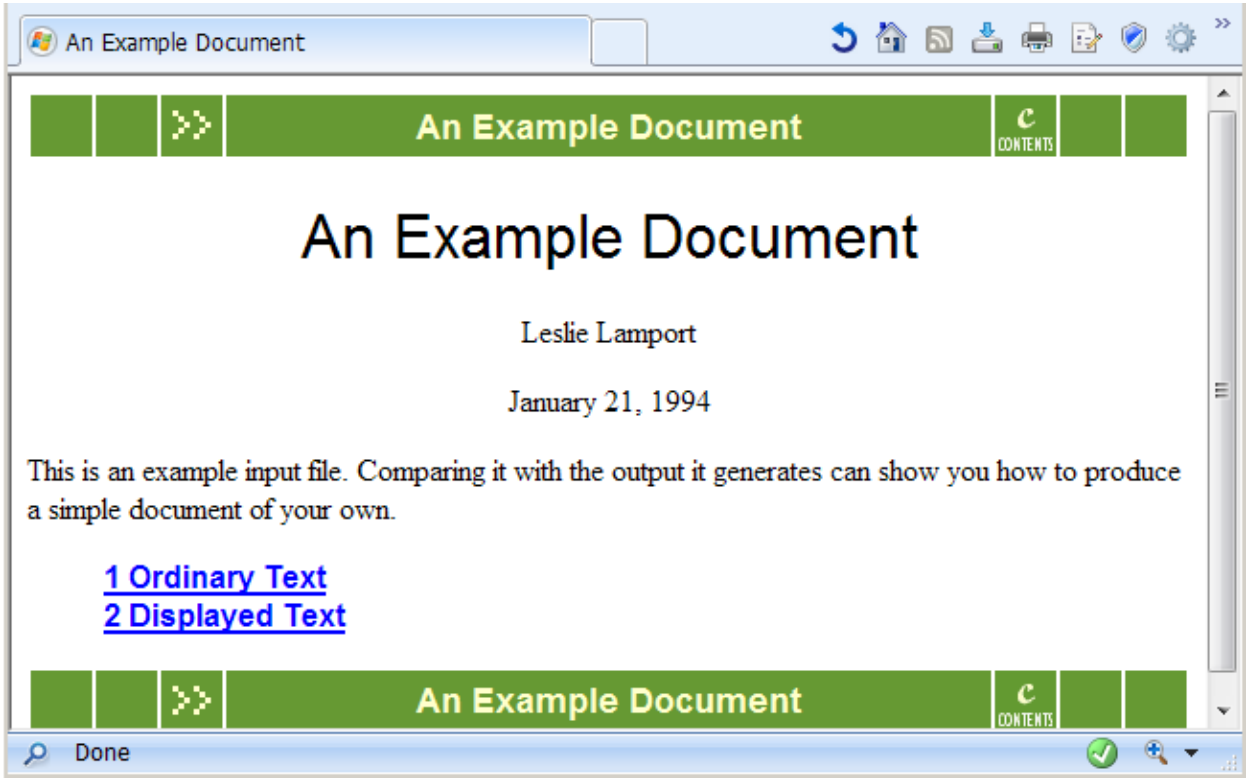


Figure 1: PlasTEX HTML output for `sample2e.tex`.

5 References

ArXiV is an e-print service in the fields of physics, mathematics, nonlinear science, computer science, quantitative biology and statistics.
`http://arxiv.org/`

BrlTEX is an open source LATEX-to-braille translator.
`http://brltex.sourceforge.net/`

DocBook is a schema maintained by the DocBook Technical Committee of OASIS. It is particularly well suited to books and papers about computer hardware and software.
`http://www.docbook.org/`

IDPF (International Digital Publishing Forum) is a trade and standards organization dedicated to the development and promotion of electronic publishing.
`http://www.idpf.org/`

plasTEX is a LATEX document-processing framework written entirely in Python, with the capability to output in many formats.
`http://plastex.sourceforge.net/`

S5 is a slide show format based entirely on XHTML, CSS, and JavaScript.
`http://www.s5.com/`

◇ Tim Arnold
SAS Institute Inc.
Cary, NC USA
`tim dot arnold (at) sas dot com`

LuaTEX: Halfway to version 1

Hans Hagen

Abstract

We're close to releasing version 0.50 of LuaTEX. In this article I give an overview of what has happened so far, what is currently being done and where we expect to end up.

1 Introduction

We are about halfway into the LuaTEX project now. At the time of writing this document we are only a few days away from version 0.40 (the BachoTEX cq. TEX Live version) and around EuroTEX 2009 we will release version 0.50. Starting with version 0.30 (which we released around the conference of the Korean TEX User group meeting) all one-decimal releases are supported and usable for (controlled) production work. We have always stated that all interfaces may change until they are documented to be stable, and we expect to document the first stable parts in version 0.50. Currently we plan to release version 1.00 sometime in 2012, 30 years after TEX82, with 0.60 and 0.70 in 2010, 0.80 and 0.90 in 2011. But of course it might turn out different.

In this update we assume that the reader knows what LuaTEX is and what it does.

2 Design principles

We started this project because we wanted an extensible engine. We chose Lua as the glue language. We do not regret this choice as it permitted us to open up TEX's internals reasonably well. There have been a few extensions to TEX itself, and there will be a few more, but none of them are fundamental in the sense that they influence typesetting. Extending TEX in that area is up to the macro package writer, who can use the Lua language combined with TEX macros.

In a similar fashion we made some decisions about Lua libraries that are included. What we have now is what you will get. Future versions of LuaTEX will have the ability to load additional libraries but these will not be part of the core distribution. There is simply too much choice and we do not want to enter endless discussions about what is best. More flexibility would also add a burden on maintenance that we do not want. Portability has always been a virtue of TEX and we want to keep it that way.

3 Lua scripting

Before 0.40 there could be multiple instances of the Lua interpreter active at the same time, but we have now decided to limit the number of instances to just one. The reason is simple: sharing all functionality among multiple Lua interpreter instances does more bad than good and Lua has enough possibilities to create namespaces anyway. The new limit also simplifies the internal source code, which is a good thing. While the `\directlua` command is now sort of frozen, we might extend the functionality of `\latelua`, especially in relation to what is possible in the backend. Both commands still accept a number but this now refers to an index in a user-definable name table that will be shown when an error occurs.

4 Input and output

The current LuaTEX release permits multiple instances of `kpse` which can be handy if you mix, for instance, a macro package and MPlib, as both have their own 'progname' (and engine) namespace. However, right from the start it has been possible to bring most input under Lua control and one can overload the usual `kpse` mechanisms. This is what we do in ConTEXt (and probably only there).

Logging, etc., is also under Lua control. There is no support for writing to TEX's opened output channels except for the log and the terminal. We are investigating limited write control to numbered channels but this has a very low priority.

Reading from zip files and sockets has been available for a while now.

Among the first things that have been implemented is a mechanism for managing category codes (`\catcode`) although this is not really needed for practical usage as we aim at full compatibility. It just makes printing back to TEX from Lua a bit more comfortable.

5 Interface to TEX

Registers can always be accessed from Lua by number and (when defined at the TEX end) also by name. When writing to a register grouping is honored. Most internal registers can be accessed (mostly read-only). Box registers can be manipulated but users need to be aware of potential memory management issues.

There will be provisions to use the primitives related to setting codes (lowercase codes and such). Some of this functionality will be available in version 0.50.

6 Fonts

The internal font model has been extended to the full Unicode range. There are readers for OpenType, Type 1, and traditional TEX fonts. Users can create virtual fonts on the fly and have complete control over what goes into TEX. Font specific features can

either be mapped onto the traditional ligature and kerning mechanisms or be implemented in Lua.

We use code from FontForge that has been stripped to get a smaller code base. Using the FontForge code has the advantage that we get a similar view on the fonts in LuaTEX as in this editor which makes debugging easier and developing fonts more convenient.

The interface is already rather stable but some of the keys in loaded tables might change. Almost all of the font interface will be stable in version 0.50.

7 Tokens

It is possible to intercept tokenization. Once intercepted, a token table can be manipulated before being piped back into LuaTEX. We still support Omega's translation processes but that might become obsolete at some point.

Future versions of LuaTEX might use Lua's so-called "user data" concept but the interface will mostly be the same. Therefore this subsystem will not be frozen yet in version 0.50.

8 Nodes

Users have access to the node lists in various stages. This interface has already been quite stable for some time but some cleanup might still take place. Currently the node memory maintenance is still explicit, but eventually we will make releasing unused nodes automatic.

We have plans for keeping more extensive information within a paragraph (initial whatsit) so that one can build alternative paragraph builders in Lua. There will be a vertical packer (in addition to the horizontal packer) and we will open up the page builder (inserts etc.). The basic interface will be stable in version 0.50.

9 Attributes

This new kid on the block is now available for most subsystems but we might change some of its default behaviour. As of 0.40 you can also use negative values for attributes. The original idea of using negative values for special purposes has been abandoned as we consider a secondary (faster and more efficient) limited variant. The basic principles will be stable around version 0.50, but we reserve the freedom to change some aspects of attributes until we reach version 1.00.

10 Hyphenation

In LuaTEX we have clearly separated hyphenation, ligature building and kerning. Managing patterns as well as hyphenation is reimplemented from scratch but uses the same principles as traditional TEX. Patterns can be loaded at run time and exceptions are quite efficient now. There are a few extensions, like embedded discretionaries in exceptions and pre- as well as posthyphens.

On the agenda is fixing some 'hyphenchar' related issues and future releases might deal with compound words as well. There are some known limitations that we hope to have solved in version 0.50.

11 Images

Image handling is part of the backend. This part of the pdfTEX code has been rewritten and can now be controlled from Lua. There are already a few more options than in pdfTEX (simple transformations). The image code will also be integrated in the virtual font handler.

12 Paragraph building

The paragraph builder has been rewritten in C code (soon to be converted back to CWEB). There is a callback related to the builder so it is possible to overload the default line breaker by one written in Lua.

There are no further short-term revisions on the agenda, apart from writing an advanced (third order) Arabic routine for the Oriental TEX project.

Future releases may provide a bit more control over `\parshape`s and multiple paragraph shapes.

13 MetaPost

The closely related MPlib project has resulted in a MetaPost library that is included in LuaTEX. There can be multiple instances active at the same time and MetaPost processing is very fast. Conversion to PDF is to be done with Lua.

On the to-do list is a bit more interoperability (pre- and postscript tables) and this will make it into release 0.50 (maybe even in version 0.40 already).

14 Mathematics

Version 0.50 will have a stable version of Unicode math support. Math is backward compatible but provides solutions for dealing with OpenType math fonts. We provide math lists in their intermediate form (noads) so that it is possible to manipulate math in great detail.

The relevant math parameters are reorganized according to what OpenType math provides (we use the Cambria font as our reference). Parameters are grouped by style. Future versions of LuaTEX will build upon this base to provide a simple mechanism for switching style sets and font families in-formula.

There are new primitives for placing accents (top and bottom variants and extensible characters), creating radicals, and making delimiters. Math characters are permitted in text mode.

There will be an additional alignment mechanism analogous to what MathML provides. Expect more.

15 Page building

Not much work has been done on opening up the page builder although we do have access to the intermediate lists. This is unlikely to happen before 0.50.

16 Going CWEB

After releasing version 0.50 around EuroTEX 2009 there will be a period of relative silence. Apart from bug fixes and (private) experiments there will be no release for a while. At the time of the 0.50 release the LuaTEX source code will probably be in plain C completely. After that is done, we will concentrate hard on consolidating and upgrading the code base back into CWEB.

17 Cleanup

Cleanup of code is a continuous process. Cleanup is needed because we deal with a merge of traditional TEX, ε-TEX extensions, pdfTEX functionality and some Omega (Aleph) code.

Compatibility is a prerequisite, with the exception of logging and rather special ligature reconstruction code.

We also use the opportunity to slowly move away from all the global variables that are used in the Pascal version.

18 Alignments

We do have some ideas about opening up alignments, but it has a low priority and it will not happen before the 0.50 release.

19 Error handling

Once all code is converted to CWEB, we will look into error handling and recovery. It has no high priority as it is easier to deal with after the conversion to CWEB.

20 Backend

The backend code will be rewritten stepwise. The image related code has already been redone, and currently everything related to positioning and directions is redesigned and made more consistent. Some bugs in the Aleph code (inherited from Omega) have been removed and we are trying to come up with a consistent way of dealing with directions. Conceptually this is somewhat messy because much directionality is delegated to the backend.

We are experimenting with positioning (preroll) and better literal injection. Currently we still use the somewhat fuzzy pdfTEX methods that evolved over time (direct, page and normal injection) but we will come up with a clearer model.

Accuracy of the output (PDF) will be improved and character extension (hz) will be done more efficiently. Experimental code seems to work okay. This will become available from release 0.40 and onwards and further cleanup will take place when the CWEB code is there, as much of the PDF backend code is already C code.

21 ConTEXt MkIV

When we started with LuaTEX we decided to use a branch of ConTEXt for testing as it involves quite drastic changes, many rewrites, a tight connection with binary versions, etc.

As a result for some time we now have two versions of ConTEXt: MkII and MkIV, where the former targets pdfTEX and XETEX, and the latter exclusively uses LuaTEX. Although the user interface is downward compatible the code base starts to diverge more and more. Therefore at the last ConTEXt meeting it was decided to freeze the current version of MkII and only apply bug fixes and an occasional simple extension.

This policy change opened the road to rather drastic splitting of the code, also because full compatibility between MkII and MkIV is not required. Around LuaTEX version 0.40 the new, currently still experimental, document structure related code will be merged into the regular MkIV version. This might have some impact as it opens up new possibilities.

22 Future

In the future, MkIV will try to create (more) clearly separated layers of functionality so that it will become possible to make subsets of ConTEXt for special purposes. This is done under the name MetaTEX. Think of layering like:

- I/O, catcodes, callback management, helpers
- input regimes, characters, filtering
- nodes, attributes and noads
- user interface
- languages, scripts, fonts and math
- spacing, par building and page construction
- XML, graphics, MetaPost, job management, and structure (huge impact)
- modules, styles, specific features
- tools

23 Fonts: future

At this moment MkIV is already quite capable of dealing with OpenType fonts. The driving force behind this is the Oriental TEX project which brings along some very complex and feature rich Arabic font technology. Much time has gone into reverse engineering the specification and behaviour of how these fonts behave in Uniscribe (which we use as our reference for Arabic).

Dealing with the huge CJK fonts is less a font issue and more a matter of node list processing. Around the annual meeting of the Korean User Group we got much of the machinery working, thanks to discussions on the spot and on the mailing list.

24 Math: future

Between LuaTEX versions 0.30 and 0.40 the math machinery was opened up (stage one). In order to test this new functionality, MkIV's math subsystem (that was then already partially Unicode aware) had to be adapted.

First of all Unicode permits us to use only one math family and so MkIV now does that. The implementation uses Microsoft's Cambria Math font as a benchmark. It creates virtual fonts from the other (old and new) math fonts so they appear to match up to Cambria Math. Because the TEX Gyre math project is not yet up to speed MkIV currently uses virtual variants of these fonts that are created at run time. The missing pieces in for instance Latin Modern and friends are compensated for by means of virtual characters.

Because it is now possible to parse the intermediate noad lists MkIV can do some manipulations before the formula is typeset. This is for instance used for alphabet remapping, forcing sizes, and spacing around punctuation.

Although MkIV already supports most of the math that users expect there is still room for improvement once there is even more control over the machinery. This is possible because MkIV is not bound to downward compatibility.

As with all other LuaTEX related MkIV code, it is expected that we will have to rewrite most of the current code a few times as we proceed, so MkIV math support is not yet stable either. We can take such drastic measures because MkIV is still experimental and because users are willing to do frequent synchronous updating of macros and engine. In the process we hope to get away from all ad-hoc boxing and kerning and whatever solutions for creating constructs, by using the new accent, delimiter, and radical primitives.

25 Tracing and testing

Whenever possible we add tracing and visualization features to ConTEXt because the progress reports and articles need them. Recent extensions concerned tracing math and tracing OpenType processing.

The OpenType tracing options are a great help in stepwise reaching the goals of the Oriental TEX project. This project gave the LuaTEX project its initial boost and aims at high quality right-to-left typesetting. In the process complex (test) fonts are made which, combined with the tracing mentioned, help us to reveal the secrets of OpenType.

◇ Hans Hagen
Pragma ADE
The Netherlands
http://luatex.org

LuaTEX and ConTEXt: Where we stand

Hans Hagen

Abstract

We consider the release of LuaTEX 0.50 to be a very important one, both for LuaTEX and for MkIV, so here I will reflect on the state around this release. I will do this from the perspective of processing documents because usability is an important measure.

1 Where do we stand?

There are several reasons why LuaTEX 0.50 is an important release, both for LuaTEX and for MkIV. Let's start with LuaTEX.

- Apart from a couple of bug fixes, the current version is pretty usable and stable. Details of what we've reached so far have been presented previously.
- The code base has been converted from Pascal to C code, and as a result the source tree has become simpler (being CWEB compliant happens around 0.60). This transition also opens up the possibility to start looking into some of the more tricky internals, like page building.
- Most of the front end has been opened up and the new backend code is getting into shape. As the backend was partly already done in C code the moment has come to do a real cleanup. Keep in mind that we started with pdfTEX and that much of its extra functionality is rather interwoven with traditional TEX code.

If we look at ConTEXt, we've also reached a crucial point in the upgrade.

- The code base is now divided into MkII and MkIV. This permits us not only to reimplement bits and pieces (something that was already in progress) but also to clean up the code (only MkIV).
- If you kept up with the development you already know the kind of tasks we can (and do) delegate to Lua. Just to mention a few: file handling, font loading and OpenType processing, casing and some spacing issues, everything related to graphics and MetaPost, language support, color and other attributes, input regimes, XML, multipass data, etc.
- Recently all backend related code was moved to Lua and the code dealing with hyperlinks, widgets and alike is now mostly moved away from TEX. The related cleanup was possible because we no longer have to deal with a mix of DVI drivers too.
- Everything related to structure (which includes numbering and multi-pass data like tables of contents and registers) is now delegated to Lua. We move around way more information and will extend these mechanisms in the near future.

2 Performance testing

Tracing on Taco's machine has shown that when processing the LuaTEX reference manual the engine spends about 10% of the time on getting tokens, 15% on macro expansion, and some 50% on Lua (callback interfacing included). Especially the time spent by Lua differs per document and garbage collections seems to be a bottleneck here. So, let's wrap up how LuaTEX performs around the time of 0.50. We use three documents for testing (intermediate) LuaTEX binaries: the reference manual, the history document 'mk', and the revised Metafun manual.

The reference manual has a MetaPost graphic on each page which is positioned using the ConTEXt background layering mechanism. This mechanism is active only when backgrounds are defined and has some performance consequences for the page builder. However, most time is spent on constructing the tables (tabulate) and because these can contain paragraphs that can run over multiple pages, constructing a table takes a few analysis passes per table plus some so-called vsplitting. We load some fonts (including narrow variants) but for the rest this document is not that complex. Of course colors are used as well as hyperlinks. The report at the end of these runs is shown in figure 1.

The runtime is influenced by the fact that some startup time and font loading takes place. The more pages your document has, the less the runtime is influenced by this.

More demanding is the 'mk' document (figure 2). Here we have many fonts, including some really huge CJK and Arabic ones (and these are loaded at several sizes and with different features). The reported font load time is large but this is partly due to the fact that on my machine for some reason passing the tables to TEX involved a lot of pagefaults (we think that the cpu cache is the culprit). Older versions of LuaTEX didn't have that performance penalty, so probably half of the reported font loading time is kind of wasted.

The hnode processing time refers mostly to OpenType font processing and attribute processing time has to do with backend issues (like injecting color directives). The more features you enable, the larger these numbers get. The MetaPost font loading refers to the punk font instances.

```
input load time           - 0.109 seconds
stored bytecode data      - 184 modules, 45 tables, 229 chunks
node list callback tasks  - 4 unique tasks, 4 created, 20980 calls
cleaned up reserved nodes - 29 nodes, 10 lists of 1427
node memory usage         - 19 glue_spec, 2 dir
h-node processing time    - 0.312 seconds including kernel
attribute processing time - 1.154 seconds
used backend              - pdf (backend for directly generating pdf output)
loaded patterns           - en:us:pat:exc:2
jobdata time              - 0.078 seconds saving, 0.047 seconds loading
callbacks                 - direct: 86692, indirect: 13364, total: 100056
interactive elements      - 178 references, 356 destinations
v-node processing time    - 0.062 seconds
loaded fonts              - 43 files: ....
fonts load time           - 1.030 seconds
metapost processing time  - 0.281 seconds, loading: 0.016 seconds,
                            execution: 0.156 seconds, n: 161
result saved in file      - luatexref-t.pdf
luatex banner             - this is luatex, version beta-0.42.0
control sequences         - 31880 of 147189
current memory usage      - 106 MB (ctx: 108 MB)
runtime                   - 12.433 seconds, 164 processed pages,
                            164 shipped pages, 13.191 pages/second
```

Figure 1: Timing reports for the LuaTEX reference manual.

```
input load time           - 0.125 seconds
stored bytecode data      - 184 modules, 45 tables, 229 chunks
node list callback tasks  - 4 unique tasks, 4 created, 24295 calls
cleaned up reserved nodes - 116 nodes, 29 lists of 1411
node memory usage         - 21 attribute, 23 glue_spec, 7 attribute_list,
                            7 local_par, 2 dir
h-node processing time    - 1.763 seconds including kernel
attribute processing time - 2.231 seconds
used backend              - pdf (backend for directly generating pdf output)
loaded patterns           - en:us:pat:exc:2 en-gb:gb:pat:exc:3 nl:nl:pat:exc:4
language load time        - 0.094 seconds, n=4
jobdata time              - 0.062 seconds saving, 0.031 seconds loading
callbacks                 - direct: 98199, indirect: 20257, total: 118456
xml load time             - 0.000 seconds, lpath calls: 46, cached calls: 31
v-node processing time    - 0.234 seconds
loaded fonts              - 69 files: ....
fonts load time           - 28.205 seconds
metapost processing time  - 0.421 seconds, loading: 0.016 seconds,
                            execution: 0.203 seconds, n: 65
graphics processing time  - 0.125 seconds including tex, n=7
result saved in file      - mk.pdf
metapost font generation  - 0 glyphs, 0.000 seconds runtime
metapost font loading     - 0.187 seconds, 40 instances,
                            213.904 instances/second
luatex banner             - this is luatex, version beta-0.42.0
control sequences         - 34449 of 147189
current memory usage      - 454 MB (ctx: 465 MB)
runtime                   - 50.326 seconds, 316 processed pages,
                            316 shipped pages, 6.279 pages/second
```

Figure 2: Timing reports for the 'mk' document.

Looking at the Metafun manual one might expect that one needs even more time per page but this is not true. We use OpenType fonts in base mode as we don't use fancy font features (base mode uses traditional TEX methods). Most interesting here is the time involved in processing MetaPost graphics. There are a lot of them (1772) and in addition we have 7 calls to independent ConTEXt runs that take one third of the total runtime. About half of the runtime involves graphics. See figure 3.

By now it will be clear that processing a document takes a bit of time. However, keep in mind

```
input load time           - 0.109 seconds
stored bytecode data      - 184 modules, 45 tables, 229 chunks
node list callback tasks  - 4 unique tasks, 4 created, 33510 calls
cleaned up reserved nodes - 39 nodes, 93 lists of 1432
node memory usage         - 249 attribute, 19 glue_spec, 82 attribute_list,
                            85 local_par, 2 dir
h-node processing time    - 0.562 seconds including kernel
attribute processing time - 2.512 seconds
used backend              - pdf (backend for directly generating pdf output)
loaded patterns           - en:us:pat:exc:2
jobdata time              - 0.094 seconds saving, 0.031 seconds loading
callbacks                 - direct: 143950, indirect: 28492, total: 172442
interactive elements      - 214 references, 371 destinations
v-node processing time    - 0.250 seconds
loaded fonts              - 45 files: l.....
fonts load time           - 1.794 seconds
metapost processing time  - 5.585 seconds, loading: 0.047 seconds,
                            execution: 2.371 seconds, n: 1772,
                            external: 15.475 seconds (7 calls)
mps conversion time       - 0.000 seconds, 1 conversions
graphics processing time  - 0.499 seconds including tex, n=74
result saved in file      - metafun.pdf
luatex banner             - this is luatex, version beta-0.42.0
control sequences         - 32587 of 147189
current memory usage      - 113 MB (ctx: 115 MB)
runtime                   - 43.368 seconds, 362 processed pages,
                            362 shipped pages, 8.347 pages/second
```

Figure 3: Timing reports for the Metafun manual.

that these documents are a bit atypical. Although ... the average ConTEXt document probably uses color (including color spaces that involve resource management), and has multiple layers, which involves some testing of the about 30 areas that make up the page. And there is the user interface that comes with a price.

3 Fonts and performance

It might be good to say a bit more about fonts. In ConTEXt we use symbolic names and often a chain of them, so the abstract `SerifBold` resolves to `MyNiceFontSerif-Bold` which in turn resolves to `mnfs-bold.otf`. As XETEX introduced lookup by internal (or system) fontname instead of filename, MkII also provides that method but MkIV adds some heuristics to it. Users can specify font sizes in traditional TEX units but also relative to the body font. All this involves a bit of expansion (resolving the chain) and parsing (of the specification). At each of the levels of name abstraction we can have associated parameters, like features, fallbacks and more. Although these mechanisms are quite optimized this still comes at a performance price.

Also, in the default MkIV font setup we use a couple more font variants (as they are available in Latin Modern). We've kept definitions sort of dynamic so you can change them and combine them in many ways. Definitions are collected in typescripts which are filtered. We support multiple mixed font sets which takes a bit of time to define but switching is generally fast. Compared to MkII the model lacks the (font) encoding and case handling code (here we gain speed) but it now offers fallback fonts (replaced ranges within fonts) and dynamic OpenType font feature switching. When used we might lose a bit of processing speed although fewer definitions are needed which gets us some back. The font subsystem is anyway a factor in the performance, if only because more complex scripts or font features demand extensive node list parsing.

Processing *The TEXbook* with LuaTEX on Taco's machine takes some 3.5 seconds in pdfTEX and 5.5 seconds in LuaTEX. This is because LuaTEX internally is Unicode and has a larger memory space. The few seconds more runtime are consistent with this. One of the reasons that *The TEXbook* processes fast is that the font system is not that complex and has hardly any overhead, and an efficient output routine is used. The format file is small and the macro set is optimal for the task. The coding is rather low level so to say (no layers of interfacing). Anyway, 100 pages per second is not bad at all and we don't come close with ConTEXt and the kind of documents that we produce there.

4 Engine performance comparisons

This made me curious as to how fast really dumb documents could be processed. It does not make sense to compare plain TEX and ConTEXt because

they do different things. Instead I decided to look at differences in engines and compare runs with different numbers of pages. That way we get an idea of how startup time influences overall performance. We look at pdfTEX, which is basically an 8-bit system, XETEX, which uses external libraries and is Unicode, and LuaTEX which is also Unicode, but stays closer to traditional TEX but has to check for callbacks.

In our measurement we use a really simple test document as we only want to see how the baseline performs. As not much content is processed, we focus on loading (startup), the output routine and page building, and some basic PDF generation. After all, it's often a quick and dirty test that gives users their first impression. When looking at the times you need to keep in mind that XETEX pipes to DVIPDFM*x* and can benefit from multiple cpu cores. All systems have different memory management and garbage collection might influence performance (as demonstrated in an earlier chapter of the 'mk' document we can trace in detail how the runtime is distributed). As terminal output is a significant slowdown for TEX we run in batchmode. The test is as follows:

```
\starttext
    \dorecurse{2000}{test\page}
\stoptext
```

On my laptop (Dell M90 with 2.3Ghz T76000 Core 2 and 4MB memory running Vista) I get the following results. The test script ran each test set 5 times and we show the fastest run so we kind of avoid interference with other processes that take time. In practice runtime differs quite a bit for similar runs, depending on the system load. The time is in seconds and between parentheses the number of pages per seconds is mentioned.

engine	30	300	2000	10000
xetex	1.81 (16)	2.45 (122)	6.97 (286)	29.20 (342)
pdftex	1.28 (23)	2.07 (144)	6.96 (287)	30.94 (323)
luatex	1.48 (20)	2.36 (127)	7.85 (254)	34.34 (291)

The next table shows the same test but this time on a 2.5Ghz E5420 quad core server with 16GB memory running Linux, but with 6 virtual machines idling in the background. All binaries are 64 bit.

engine	30	300	2000	10000
xetex	0.92 (32)	1.89 (158)	8.74 (228)	42.19 (237)
pdftex	0.49 (61)	1.14 (262)	5.23 (382)	24.66 (405)
luatex	1.07 (27)	1.99 (150)	8.32 (240)	38.22 (261)

A test demonstrated that for LuaTEX the 30 and 300 page runs take 70% more runtime with 32 bit binaries (recent binaries for these engines are available on the ConTEXt wiki `contextgarden.net`).

When you compare both tables it will be clear that it is non-trivial to come to conclusions about performances. But one thing is clear: LuaTEX with ConTEXt MkIV is not performing that badly compared to its cousins. The Unicode engines perform about the same and pdfTEX beats them significantly. Okay, I have to admit that in the meantime some cleanup of code in MkIV has happened and the LuaTEX runs benefit from this, but on the other hand, the other engines are not hindered by callbacks. As I expect to use MkII less frequently optimizing the older code makes no sense.

5 Futures

There is not much chance of LuaTEX itself becoming faster, although a few days before writing this Taco managed to speed up font inclusion in the backend code significantly (we're talking about half a second to a second for the three documents used here). On the contrary, when we open up more mechanisms and have upgraded backend code it might actually be a bit slower. On the other hand, I expect to be able to clean up some more ConTEXt code, although we already got rid of some subsystems (like the rather flexible (mixed) font encoding, where each language could have multiple hyphenation patters, etc.). Also, although initial loading of math fonts might take a bit more time (as long as we use virtual Latin Modern math), font switching is more efficient now due to fewer families. But speedups in the ConTEXt code might be compensated for by more advanced mechanisms that call out to Lua. You will be surprised by how much speed can be improved by proper document encoding and proper styles. I can try to gain a couple more pages per second by more efficient code, but a user's style that does an inefficient massive font switch for some 10 words per page easily compensates for that.

When processing the present 10 page document in an editor (Scite) it takes some 2.7 seconds between hitting the processing key and the result showing up in Acrobat. I can live with that, especially when I keep in mind that my next computer will be faster.

This is where we stand now. The three reports shown before give you an impression of the impact of LuaTEX on ConTEXt. To what extent is this reflected in the code base? Eventually most MkII files (with the `mkii` suffix) and MkIV files (with suffix `mkiv`) will differ and the number of files with the `tex` suffix will be fewer. Because they are and will be mostly downward compatible, styles and modules will be shared as much as possible.

◇ Hans Hagen
Pragma ADE
`http://pragma-ade.com`

ProofCheck: Writing and checking complete proofs in LaTeX

Bob Neveln and Bob Alps

Abstract

ProofCheck is a system for writing and checking mathematical proofs. Theorems and proofs are contained in a plain TeX or LaTeX document. Parsing and proof checking are accomplished through Python programs which read the source file. A general explanation of the use and structure of the system and programs is provided and a sample proof is shown in detail. The work done by the authors has been based on standard sentence logic, a non-standard predicate logic and set theory with proper classes. Theorems and proofs based on other foundations may be checked if external data files are modified. Four such data files and their possible modifications are described. In addition, the extent to which the formal language can be shaped to accommodate an author's preferences is discussed.

1 Introduction

The purpose of ProofCheck is to enable mathematicians to write readable proofs that are computer checkable. Such readable, computer-checkable proofs could also be of value in the refereeing process. In a previous article [4] the system now called ProofCheck was described. Since then the system has been extended in several ways. The number of inference rules and common notions has been increased. A web site at `www.proofcheck.org` has been developed. Further, the system now works with LaTeX in addition to plain TeX.

2 Other systems

Two well-known systems to which ProofCheck might be compared are HOL [2] and Mizar [5].

HOL (Higher Order Logic) is a computer-assisted theorem proving system operating within the OCAML environment. OCAML (Objective CAML) is an object-oriented programming language which can be run interactively. Mathematical objects are treated as OCAML objects and mathematical theorems are stated in OCAML. Thus both ontology and syntax are subsumed by OCAML type theory. Theorems are proved by entering commands at the OCAML prompt or by running a script. Thus a proof consists of a record of OCAML commands.

Mizar is also a language for stating proofs intended to be human-readable as well as computer-checkable. Proofs are entered in ASCII text in the Mizar language. The language is extensive and based on a particular axiomatization of mathematics, Tarski-Grothendieck set theory. The system can also produce LaTeX output.

These systems also tend to form closed mathematical systems. Proving a theorem using these systems means showing that the given theorem is provable within that system.

3 Features and goals

TeX and LaTeX convert an author's `.tex` source files into DVI or PDF output. ProofCheck works with these same `.tex` source files. When using ProofCheck the work cycle is typically to run the parser and the checker after a successful run of TeX or LaTeX. Errors encountered in each case throw the author back into the text editor.

ProofCheck is intended to be open with respect to mathematical foundations. When ProofCheck checks the proof of a theorem it shows that the theorem follows from definitions and other theorems which have been stated and parsed but not necessarily checked. An author does not need to commit to a particular axiom system.

In their own mathematical work the authors use standard sentence logic, a non-standard predicate logic, and a set theory which admits proper classes — but none of these choices is required. Of course checkable-proofs are easier to write when there is an accumulated body of accepted propositions available for referral. This does constitute an implicit pressure to use the specific development already on the ProofCheck web-site. But nothing prevents an author from creating another one. We will post any such developments we receive.

4 Mathematical language

For almost a century there has been general agreement that there is no obstacle in principle to writing mathematics in a formal language and therefore to checking proofs mechanically. One of the main obstacles in practice to stating definitions and theorems in a formal language is that the required sacrifice of syntactical freedom may be more than a mathematical author is willing to tolerate. In devising a usable proof-checking system therefore it is important to maximize syntactic freedom.

ProofCheck is not built on any specific mathematical language. Instead, a context-free grammar is generated on the basis of whatever definitions have been made. The rules of grammar are based on syntactical ideas of A.P. Morse [3], and include a variation on Morse's handling of second-order variables. Definitions are presumed to take the form $(p \leftrightarrow q)$ for formulas and $(x \equiv y)$ for terms. The set

of definitions is taken to include a standard default set of infix operators as well as quantifiers. The default symbols may be easily replaced and with a little additional effort, default forms can be replaced. The resulting language is unambiguous and possesses the property that no term or formula can begin another.

ProofCheck minimizes the loss of syntactical freedom by striving to keep the syntactical restrictions as close as possible to the absolute requirements of logic and consistency.

5 Proof language

The following TEX macros suffice to mark up a proof for checking. We are grateful to Karl Berry for his help in streamlining the first two of these. The general proof structure which these macros implement is very similar to that defined in [1].

`\prop` Any proposition, whether a theorem, a definition or an axiom, must be introduced by this macro. Its use requires ProofCheck to look for a proof. Its syntax is `\prop`, followed by the enumeration, followed by the proposition. For example:

```
\prop 14.7 $(1 < 2)$
```

The `\prop` macro must be at the beginning of a line. The enumeration is of the form $n.m$ where n and m are positive integers with n representing the section or chapter number. The proposition must be enclosed within TEX dollar signs. References in a proof to other theorems use the same $n.m$ style and not LATEX labels. When re-arrangement of theorems necessitates renumbering, this is handled by a ProofCheck program called `renum`. This program recognizes LATEX section macros so that unless the section counter is reset manually, consistency with LATEX sectioning is maintained.

`\note` This macro is used to introduce an assertion within the proof which can be referred to later in the proof. The assertion may be the result of telescoping multiple lines of proof, each having an optional justification. The ProofCheck syntax is `\note`, followed by the note enumeration, followed by the (possibly multi-line) assertion. Here the enumeration consists of a positive integer.

`\By` This macro is used at the end of a line to introduce justification for the step. A note with a single line of assertion may be justified by one of the following:

`\By G` (Given) signals an assertion which is to be accepted locally as a hypothesis. It initiates a block of the proof within which this hypothesis is in effect.

`\By S` (Set) indicates that the note is used to locally define a variable for use in the proof.

`\By .n H .m` (Hence) where n is the number of a note established using note m as a Given, ends the block initiated by note m. Such blocks are called Given-Hence blocks. A Hence justification is typically used to establish an assertion such as $(p \to q)$ where p is note m and q is note n.

Notes with either a single line or more than one line of assertion may be justified using other notes and a `.m` enumeration or other propositions and `n.m` enumeration with punctuation as shown in the sample proof. More discussion on this syntax may be found in [4].

`\Bye` in addition to introducing a justification, signals the end of a proof and prints "Q.E.D.".

`\linea, \lineb, ...` These begin a new line with increasing degrees of indentation. They are not proof macros per se but are used for any mathematical expressions that need to go beyond one line and need indentation. At present the parser allows only these.

In section 7 the use of these macros in a sample proof is shown.

6 Sample proof: Reader view

The simple proof in figure 1 is taken from a development of the von Neumann model of the natural numbers, ω, in which each natural number is the set of the preceding natural numbers. The theorem asserts that if y is an element of a natural number x then y is a subset of x.

The first line of the proof defines a set A. This note should be easy to read except for possibly the quantifier notation and the classifier notation. Notational changes are discussed in section 9. The second note translates the definition in note 1 into a bi-conditional which is much more useful deductively. We often refer to steps that turn a definition into one or more implications as "unwrapping" steps. Explicit inclusion of such unwrapping steps is often key in getting a proof to check. The stage is set for an induction proof.

The theorem 4.7 referred to is just the standard induction theorem:

$$(\emptyset \in A \wedge \bigwedge x \in A(x \in A \to \mathrm{scsr}\, x \in A) \to \omega \subset A)$$

where "scsr x" denotes the successor of x. Its two hypotheses are the base case which in this proof is established in note 3 and the universalization of the

Theorem
4.8 $(y \in x \in \omega \to y \subset x)$

Proof: To prove this by induction we begin by letting A be the set of all x such that each element y of x is a subset of x. We set

.1 $(A \equiv \mathsf{E}\, x \bigwedge y \in x(y \subset x))$ ‡S

It will follow from 4.7 that ω is a subset of A. First we unwrap .1.

.2 $(x \in A \leftrightarrow \bigwedge y \in x(y \subset x) \wedge x \in \mathrm{U})$ ‡08.3;.1

Base Case $(\emptyset \in A)$.

.3 $(\emptyset \in A)$ ‡.2;09.19,09.12

Induction Step $(x \in A \to \mathrm{scsr}\, x \in A)$.

Given

.4 $(x \in A)$ ‡G

We note first that

.5 $(x \in \mathrm{U})$ ‡09.20;.4

.6 $\bigwedge y \in x(y \subset x)$ ‡.2;.4

Then we have

.7 $(y \in \mathrm{scsr}\, x \to y \in x \vee y = x$ ‡3.7
$\to y \subset x \vee y = x$ ‡.6
$\to y \subset x$ ‡011.14
$\to y \subset \mathrm{scsr}\, x)$ ‡011.10;(3.5;(09.20;.4))

So we can conclude that

.8 $(\mathrm{scsr}\, x \in A)$ ‡.2;(3.3;.5),(.7 U)

Hence

.9 $(x \in A \to \mathrm{scsr}\, x \in A)$ ‡.8 H .4

This completes the proof that

.10 $(\omega \subset A)$ ‡4.7;.3,(.9 U)

The conclusion now follows quickly.

.11 $(y \in x \wedge x \in \omega \to y \in x \wedge x \in A$ ‡011.7;.10
$\to y \in x \wedge \bigwedge y \in x(y \subset x)$ ‡.2
$\to y \subset x)$
Q.E.D. .11

Figure 1: Sample proof: DVI output

induction step established in note 9. This theorem is invoked in the justification of note 10. In the proof of the induction step, note 7 shows that any member y of scsr x is a subset of scsr x. In note 8, we conclude that scsr x is in A. In note 9 we join the hypothesis from note 4 to the conclusion obtained in note 8.

Note 11 details the step from note 10 to the theorem which is short but cannot be skipped. The "Q.E.D. .11" at the end asserts that the theorem itself follows from note 11.

7 Sample proof: Author view

In Figure 2 we have the LaTeX source code for the sample proof.

The first couple of lines of the sample proof begin explaining the proof. Since they are not noted they do not contribute to the check. But neither do they get in the way of the check. Unchecked text of any sort is admissible so long as it does not interrupt mathematical expressions or interfere with proof specification.

The reader will note that all the macros and note justifications described in section 5 are used in this sample proof. There is a Set statement in note 1, a Given-Hence block in notes 4 through 9, and the proof terminates with a `\Bye` macro.

Note 4 opens a Given-Hence block and establishes $(x \in A)$ as a working hypothesis. Note 4 may be referred to only within this Given-Hence block. A Hence justification may close more than one Given note, but each Given note must be explicitly closed by a Hence justification. This Given-Hence block is closed by note 9.

Note 7 is a multi-line note each line of which has a justification. A reference to note 7 accesses the telescoped result of the note which is

$$(y \in \mathrm{scsr}\, x \to y \subset \mathrm{scsr}\, x)$$

The `\Bye` line is justified by note 11. The telescoped note 11 and the statement of the theorem differ only notationally on the left side of the implication. A supplemental parser produces a canonical version of each of the two left sides. These two versions turn out to be identical. Consequently the rule of inference used here merely allows one to infer p from p.

8 Proof checking

Each assertion within a note is checked individually using end-of-line references to theorems and/or notes. Each of these checks is done by submitting a formula and the formulas referred to in its justification to a rule matcher which conducts a simple linear search of the list of inference rules. The search must find a rule which unifies with the submitted formulas in order for the check of the assertion to succeed.

In Figure 3 the rectangular boxes represent TeX files whereas the unifier and the rule matcher are Python scripts.

9 Working with ProofCheck files

The files `rules.tex` and `common.tex` contain the rules of inference and common notions used as the defaults for optional command line parameters of the checking script. In adapting the common notions to suit a particular mathematical interest, the file `common.tex` may be modified or another file altogether may be written. The same applies to the file `rules.tex` should the use of another logic be required. Both of these files require the inclusion of many redundant forms of whatever principles are

```
\noindent{}Theorem

\prop 4.8 $(y \in x \in \omega \c y \subset x)$
    \lineb Proof:  To prove this by induction we begin by
letting $A$ be the set of all $x$ such that each element $y$ of $x$
is a subset of $x$.  We set
\note 1 $(A \ident \setof x \Each y\in x(y\subset x))$ \By S
\linea It will follow from 4.7 that $\omega$ is a subset of $A$. First we unwrap .1.
\note 2 $(x \in A \Iff \Each y \in x (y \subset x) \And x \in \U)$ \By 08.3;.1
    \lineb Base Case $(\e \in A )$~.
\note 3 $(\e \in A)$ \By .2;09.19,09.12
    \lineb Induction Step $(x \in A \c \scsr x \in A)$~.
\linea Given
\note 4  $(x \in A )$  \By G
\linea We note first that
\note 5 $(x \in \U)$  \By 09.20;.4
\note 6 $\Each y \in x(y \subset  x)$\By .2;.4
\linea Then we have
\note 7 $(y \in \scsr x \c y \in x \Or y = x$  \By 3.7
    \lined $\c y \subset x \Or y = x$  \By .6
    \lined $\c y \subset x $  \By 011.14
    \lined $\c y \subset \scsr x )$ \By 011.10;( 3.5;(09.20;.4))
\linea So we can conclude that
\note 8 $(\scsr x \in A)$ \By .2;(3.3;.5),(.7 U)
\linea Hence
\note 9 $(x \in A \c \scsr x \in A)$ \By .8 H .4
\linea This completes the proof that
\Note 10 $(\omega \subset A)$ \By 4.7;.3,(.9 U)
\linea The conclusion now follows quickly.
\Note 11 $(y \in x \And x \in \omega\c y \in x \And x \in A$ \By 011.7;.10
    \lined $\c y \in x \And \Each y \in x(y \subset x)$ \By .2
    \lined $\c y \subset x)$
    \lineb \Bye .11
```

Figure 2: Sample proof: LaTeX input

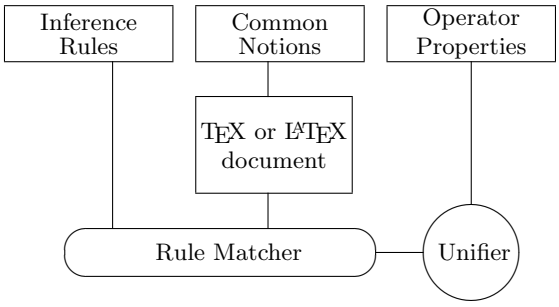


Figure 3: Main proof checking files

included, because in application many difference variations present themselves for unification. Additions contrived with a single application in mind should of course be avoided — making this distinction sometimes requires a non-trivial judgment. Building up these files is time consuming.

On the other hand, changes consisting simply of substituting one symbol for another are easily accommodated.

Common notions Common notions comprise all the definitions and theorems outside of the current working document that are needed to prove the theorems in the document. Two files `common.tex` and `common.ldf` are used to store information about common notions. The file `common.ldf` stores TEX macros to represent various constant symbols used to state the common notions, whereas `common.tex` stores the definitions and theorems themselves. If for example an author wished to use the quantifier '∀' instead of '⋀' the definition of `\Each` in this file could be changed

as follows:

```
\def\Each{\mathop{\forall}}
```

This would change the output but still require the use of the `\Each` macro in the source file.

Rules of inference Rules of inference include basic rules such as modus ponens and universalization. The role played by rules of substitution is subsumed by the unifier. The present authors have supplemented the basic rules with over 1000 additional rules. All the additional rules are consequences of the logic we use. In all but the most elementary settings, such an expansion of the rule set is essential to keeping proofs under a reasonable length. This file may be populated according to the author's accepted logic. Each entry of the rules of inference file begins with the formula to be proved, followed by `<=`, followed by the formulas needed to prove it. The entry for modus ponens has the following form:

$$q <= (p \to q)\ ;\ p$$

Math and Logic Symbols The file `equivmacros.trf` consists of a list of macro replacements made prior to sending a term or formula from the author's document to the parser. For example an author who wanted to use `\forall` as the universal quantifier in the source file could include a line in `equivmacros.trf` such as

```
forall Each
```

We will be happy to post any modifications of files as described above on the ProofCheck website.

10 Conclusion

ProofCheck is a very simple system. As shown in figure 3 it consists mainly of rules of inference, a store of assumed elementary propositions, a slightly enhanced unifier and a rule matching script. The size of a download of the complete system from `www.proofcheck.org` is less than one megabyte. The discussion of the proof language in section 5 approaches a complete tutorial. We believe that the fact that proofs can be checked with such a simple system confirms the basic ideas on which it is based.

At the conclusion of [4] we asserted that complete proofs done using ProofCheck required approximately one order of magnitude more time to write than a conventional proof, and about two or three times as much space. Experience since then does not lead us to revise these estimates significantly, although the length of proofs has diminished slightly due to the growth in the number of rules of inference. We anticipate further progress.

References

[1] W. W. Bledsoe and E.J. Gilbert. Automatic theorem proof-checking in set theory. Technical Report SC-RR-67-525, Sandia Laboratories, July 1967.

[2] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, Cambridge, 1993.

[3] A. P. Morse. *A Theory of Sets*. Academic Press, second edition, 1986.

[4] Bob Neveln and Bob Alps. Writing and checking complete proofs in TEX. *TUGboat* 28(1), 80–83, 2007.

[5] Piotr Rudnicki and Andrzej Trybulec. A collection of TEXed mizar abstracts. Technical Report TR 89-18, University of Alberta, June 1989. `www.mizar.org/project/bibliography.html`.

TEX People: The TUG interviews project and book

Karl Berry and David Walden

Abstract

We present the history and evolution of the TUG interviews project. We discuss the interviewing process as well as our methods for creating web pages and a printed book from the interviews, using m4 as a preprocessor targeting either HTML or LATEX. We also describe some business decisions relating to the book. We don't claim great generality for what we have done, but we hope some of our experience will be educational.

1 Introduction

Dave had two ideas in mind when he suggested (originally to Karl) this interview series in 2004: (a) technology is created by and evolves with use by people, and the points of view and backgrounds of the people influence the technology; (b) there are lots of people (such as himself) who are relatively new to the TEX community, who therefore do not know much about the people who are already significant contributors to the community, and who may be curious to know more about past and current contributors to the TEX community. We also hoped that even long-time members of the TEX community would enjoy reading the interviews and learn things they did not previously know about their fellow TEX people.

Dave got the idea for an interview series from reading the books *Mathematical People* (edited by Donald J. Albers and Gerald L. Alexanderson, Birkhäuser, 1985) and *More Mathematical People* (edited by Albers, Alexanderson, and Constance Reid, Harcourt, Brace, Jovanovich, 1990). The first of these collections of interviews of mathematicians includes a wonderful interview of TEX creator Donald Knuth (posted at `tug.org/interviews/interview-files/birkhauser-knuth`, by permission from Birkhäuser).

Dave's first concern was whether anyone would want to do an interview with him, an unknown in the TEX community. He was gratified that Dan Luecking (whose answers on `comp.text.tex` he admired) and Lance Carnes (with whom he was working on *The PracTEX Journal*, published at `tug.org/pracjourn`) agreed to be the first interview subjects. These interviews provided a model to which he could point the next people he asked to do interviews, while also liberally dropping Karl's name. In time, the interview series seemed to gain advance acceptance with potential interview subjects.

2 The interview method

Interview subjects are[1] chosen based on (a) seeking diversity in many dimensions, (b) recommendations from people about who should be interviewed, and (c) potential interview subjects being willing to be interviewed.

The interviews have almost all been done via exchange of email using plain text; a couple of exceptions were done in LATEX.

For the first several interviews, Dave sent a more or less complete list of questions (once the interviewee agreed to participate), on the theory that this would minimize the burden on the interviewee. However, having a dozen questions to answer at one time proved to be daunting to interviewees. Thus, Dave switched to a process wherein he sent a couple of standard initial questions (namely, "Tell me a bit about yourself and your history outside of TEX," and "How and when did you first get involved with TEX?"). The answers to early questions then guide a couple of follow-on groups of questions.

The interviewees are encouraged to answer spontaneously and at any length they desire, both to ease the burden of answering the questions and to spur spontaneity. Once enough questions have been asked and answered, Dave converts the plain text into HTML, possibly reordering some of the questions and answers to improve the flow, does other little bits of necessary editing (all the while trying to maintain the voice of the interviewee), and submits the near final draft to the interviewee for review and any desired changes.

Karl proofreads every interview before posting. He also frequently suggests additional questions, provides contact information, and generally encourages and supports Dave in his interview efforts. The interviews are then posted on the TUG web site: `tug.org/interviews`.

The average interview takes a few weeks of elapsed time. The shortest interview took only a few days. The longest interview took over a year. In a few cases, Dave has interviewed two people at one time — people who were well known for working closely together. In one case, Dave shared the interviewing duties with another person (the interview of Frank Mittelbach, which was also published in the *Free Software Magazine*). In another case (the interview of Raph Levien) Karl shared interviewing duties with another person. All the rest have been one-on-one interviews.

For quite a while Dave manually reformatted

[1] The interview series remains an ongoing process, so we describe it here in the present tense.

the interviews in HTML from the original plain text questions and answers as part of his editing pass. He is biased toward basic tools (WinEdt, Emacs, FTP, etc.) and does not use a "web publishing system".

For the first interviews, we did not think about photos. Then we decided to include photos, began asking for them from current interviewees and also asked the early interviewees for photos. We slipped up here, in light of our later decision to create a book version of the interviews: we asked for photographs with only enough resolution for web display and not enough for printing in a book. So we later had to go back to most interviewees again for higher-resolution images.

3 The idea for the book

After a while, we began to discuss the possibility of eventually producing a book collecting the interviews, after we had about three years' worth, with a dozen or so interviews done a year. Consequently, in our interview requests, we also started asking the interviewee to agree for his or her interview to be transcribed from HTML to LaTeX and be included in the book. We also confirmed that this was acceptable to earlier interviewees.

A little while later, Dave decided to create a small set of m4 macros (`gnu.org/software/m4`) in terms of which he would transcribe the plain text interviews, and the file with the m4 macros would be compiled into HTML for the interviews. His idea was that later another set of m4 macros could convert the same m4 file to LaTeX for the book. All these definitions were in a file called `htmldefs.m4`, and an interview was compiled from its m4 file with a command such as

```
m4 htmldefs.m4 SomeInterviewee.m4 \
  > SomeInterviewee.html
```

An abridged set of the initial m4-to-HTML definitions is listed in Table 1. Here are a couple of specific examples:[2]

```
define([[_par]],[[<p>]])       ( paragraph)
define([[_it]],[[<i>$1</i>]]) ( italics)
```

To generate LaTeX instead of HTML, alternate m4 definitions output `\par` and `\textit{$1}`.

The individual HTML interviews, both before and after using m4 as a stepping stone to HTML, had links in alphabetical order from the interviews home page which Dave created manually on the TUG server with Emacs. He also manually created the HTML file listing the interviews in chronological order. The result of all was posted at `tug.org/interviews`.

Some readers may wonder why we didn't use a more au courant solution, such as creating the interviews in XML and arranging for that to generate HTML initially and LaTeX later, or doing the interviews in LaTeX (which could be used for the book) and using one of the LaTeX-to-HTML converters for the web site. The answer is simple: Dave and Karl already knew something about m4 and saw nothing to be gained by struggling with anything new (and arguably more complex).

Table 1: Some definitions and their purposes from an early version of our `htmldefs.m4` file.

`_intervieweename`	used in header and first question
`_intervieweeinitials`	used in following questions
`_interviewername`	used in first question
`_interviewerinitials`	used in following questions
`_par`	start paragraph
`_header`	takes photo file and blurb as parameters
`_question1`	formatting of first question
`_answer1`	formatting of first answer
`_question`	formatting of following questions
`_answer`	formatting of following answers
`_footer`	date, etc., at end of interviews
`_link`	link given text to a URL
`_url`	print and link given URL
`_it`	italics
`_emph`	also italics
`_ti`	italics within italics
`_tt`	typewriter
`_anglebrackets`	print argument between ⟨ and ⟩
`_Dash`	em-dash
`_dash`	en-dash
`_amp`	ampersand
`_quote`	double-quote argument
`_verbatimstart`	start verbatim block
`_verbatimstop`	end verbatim block
`_verb`	inline verbatim
`_orderedliststart`	start ordered list
`_orderedliststop`	end ordered list
`_unorderedliststart`	start unordered list
`_unorderedliststop`	end unordered list
`_item`	begin an item in any list
`_TeX`, ...	lots of logos which are mostly plain text in HTML, but something fancier in TeX
`_Schopf`, ...	people's names with diacritics
`_Ecole`, ...	other words with diacritics

4 Going ahead with the book

In late 2008 we made the decision to develop the book. We anticipated self-publishing it using print-on-demand from a company like Lightning Source

[2] Readers can mentally skip over the `[[` and `]]` constructs, which we use as the m4 "quoting" characters, allowing arguments to contain characters special to m4 such as commas and parentheses. More about this later.

(LSI, lightningsource.com). Karl chose 7×10 inches as the page size from the sizes supported by LSI, because it was not so big that it would require a two-column format, but otherwise the biggest available, so minimizing the number of pages and print cost.

Karl developed the m4-to-LaTeX definition file (tug.org/interviews/book/texdefs.m4) for converting the m4 files for each interview into LaTeX files for each interview. Naturally, this evolved as new issues were discovered as the conversion into LaTeX proceeded. As an example, one of the shorter m4 interview files is available at tug.org/interviews/book/gordon.m4 (Peter Gordon's interview) and the resulting LaTeX file at tug.org/interviews/book/gordon.inc.

Some of the interviews had never been in the m4 format, and Karl converted the HTML for these earlier interviews into the m4 format using Emacs, Perl, and other Unix tools. This enabled all the interviews to be handled uniformly for the book and provided the option of later converting all the interviews, as cleaned up for the book, back to HTML to improve the interviews web site. Karl then edited all of these m4 files further to eliminate typos, other broken or awkward text, and generate well-typeset LaTeX, following the same general conventions long-established for *TUGboat*.

In addition to Karl, Barbara Beeton (whose reputation for editing prowess is well known in the TeX community) also edited each interview, reviewed the formatting, and so on. As with *TUGboat*, Barbara and Karl strived to achieve appropriate English for each interview while maintaining as much as possible the interviewee's voice.

Next, Dave circulated the edited and typeset interviews back to the interviewees for approval and possibly for an additional update, usually written as an endnote; most interviewees chose to leave their interview current as of the original interview end date. Dave also began to solicit higher resolution photos as part of these exchanges with the interviewees (since we desired the photos to be about 2 inches wide, we asked for photos that were at least 600 pixels wide); Dave also used Photoshop to convert many of the color photos to grayscale and to adjust the contrast, etc., of all photos to improve how well they printed in a black and white book.

Finally, Dave and Karl drafted the preface, title pages, and other front matter, and integrated them into the Makefile which built the entire book.

5 Technical LaTeX details

In addition to the texdefs.m4 file mentioned above, Karl developed a Makefile (tug.org/interviews/book/GNUmakefile) to automatically convert all of the m4 files into LaTeX files, and then compile the LaTeX files for the interviews and the frontmatter files into one complete PDF. The master LaTeX file which the Makefile calls to compile the book is at tug.org/interviews/book/ivbook.tex.

5.1 Table of contents

We considered several orderings for the interviews: alphabetical was an obvious possibility, but added nothing over a simple list of names; another was grouping by category, such as putting related developers together, but this proved too vague to be meaningful. Finally we settled on chronological order.

We decided to have two tables of contents, one chronological (the normal page number order, in this scheme, written by LaTeX), and one alphabetical. Karl wrote a script to sort the names and generate the alphabetical contents from the chronological one (tug.org/interviews/book/a-lphabetize).

5.2 The book style file

The master file, ivbook.tex, is mostly just a list of the frontmatter and interview files in the order we wanted to print them; the only notable thing is the initial setup:

```
\documentclass{book}
\usepackage{ivbook}
```

That is, we use the basic LaTeX book style, augmented with a style file. We have found that collecting all customizations and settings in such a central style file is a good idea for any significant project.

Here are the main things which our book style file does (tug.org/interviews/book/ivbook.sty):

- It loads some standard packages: geometry to specify the page layout, graphicx for handling the photos, microtype to help with line breaking, url for line breaking on urls, and fancyhdr to specify our desired running header (and an empty running footer).
- It loads babel with support for Vietnamese, for the sake of two so-called "horned" letters in the interview with Philip Taylor. We were grateful that Hàn Thế Thành had created Vietnamese glyphs for our font (among many others) as part of the VnTeX project (vntex.sf.net).
- It specifies the fonts used. Karl wanted to use a freely available font that was included in TeX Live, and looked through the online LaTeX Font Catalogue (tug.dk/FontCatalogue). He then experimented with different choices — the Font

Catalogue helpfully lists the LaTeX commands to use, making such experimentation easy — and settled on Charter, as extended by the Math Design Project:

```
\usepackage[bitstream-charter]{mathdesign}
```

Unfortunately, the last release of the free Charter from Bitstream (used by Math Design) includes a number of unfortunate kerning pairs, especially with punctuation, such as 'P.'. Karl ended up remaking the TFM for the main text font with the kerning from an older release (the `bchr8t.tfm` file which is in TeX Live). He wrote an ugly script to get this job done (`tug.org/interviews/book/kernfix`).

For the typewriter material, we used the Inconsolata font by Raph Levien, one of the interviewees. Karl wrote the LaTeX support macros, and later released them to CTAN (`mirror.ctan.org/fonts/inconsolata`) and TeX Live.

- Because the interviews are each only a few pages long, the style file eliminates blank verso pages by redefining `\cleardoublepage`, as suggested in the TeX FAQ (`www.tex.ac.uk/faq`).
- It defines a macro `\ivchapter` to make each interview into a separate (unnumbered) chapter, and add the name to the table of contents.
- Speaking of the tables of contents, it also makes various (re)definitions for formatting those; one that may be of interest is setting `\@pnumwidth`, the page number width, to `2.5em`; the default of `1.55em` is not enough for three digits in Charter, unlike Computer Modern.
- It defines a macro `\ivblurb` to format the beginning of each interview, including the brief description and photograph. Some of this is pretty intricate; the interested reader may like to study the code (due to Donald Knuth) which makes the "[Interview completed ...]" text be typeset flush right on the last line if it fits, or on its own line if not.
- It defines macros with which to format the questions and answers.
- It defines common macros `\Dash` for formatting dashes (we like thinspaces around them), `\acro` for acronyms (we like to set "words" in all caps one point size smaller than the main text), etc. We adopted these and many other conventions from *TUGboat*.
- It defines a lot of logos, both simple and complex. Because we're not using the Computer Modern fonts, we used the alternative logo definitions for LaTeX and the like developed by Grzegorz Murzynowski, as written up in *TUGboat* 29:1 (EuroTeX 2007 proceedings).
- It changes the percent sign and dollar sign into normal characters (`\catcode 12`), since they come up from time to time in the interviews, and enables `DEL` (aka `^^?`) as a comment character.
- It handles indenting of verbatim blocks. Karl was surprised to learn that `verbatim` mode operates by typesetting a one-item list without any list marker. This meant the usual mechanisms for indentation, such as `\leftskip`, were rendered ineffective. We ultimately resorted to loading `fancyvrb` and using its `xleftmargin` setting.
- Finally, we wanted to specify the vertical space above lists. This turned out to be surprisingly difficult. The LaTeX parameter documented to control this is `\topsep`. However, LaTeX saves a copy of the original list parameters in the macro `\@listI` (source file `classes.dtx`), which is invoked after every font size change. Our document, like most documents, changes font size from time to time (for the heading beginning each interview, etc.). Thus, to affect the spacing above lists once and for all, it is necessary to manually redefine `\@listI`, and not just set `\topsep`. The FAQ entry at `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=complist` discusses this, and gives some alternative packages to use.

6 Technical m4 details

We ended up being reasonably happy with our choice of m4 as the "high-level" language in which to write the interviews. Almost all our definitions are trivial text expansions that took no time to write and served the purpose of making it possible to generate either HTML or LaTeX without much trouble. Below we discuss two of the nontrivial definitions. (Incidentally, although all our m4 commands start with `_`, this is not a requirement of the language.)

6.1 Converting date formats

One nontrivial definition related to handling the interview dates. In the interview sources, we specify the dates like this, for an interview completed on June 25, 2005:

```
define([[_date]],[[2007-06-25]])
```

(As mentioned in a previous footnote, the `[[...]]` construct is our way of quoting, stopping m4 from expanding the quoted text further.) In the printed book, though, we wanted to show the date like this:

Interview completed 25 June 2005.

How to convert from one date format to the other? We didn't want to just change the source since the numeric format is used online, and besides, that would have been extra work. We ended up with these m4 definitions to do the job:

```
define([[_header]],dnl
 [[... completed _ivdate(_date))]])dnl
define([[_ivdate]],dnl
 [[esyscmd(date +'%e %B %Y' -d '$1')]])dnl
```

Explanation:

- `dnl` stands for "discard-to-newline", i.e., starts a comment. We use it here because there's no point in passing the newlines after each definition to the LaTeX output file.
- We use a helper macro `_ivdate` to do the transformation, and pass it the original (numeric) date from the source file.
- `_ivdate` uses the m4 builtin function `esyscmd` ("execute system command") to call the GNU date utility, which recognizes the numeric source date and reformats it in the way we want. The result is inserted in the normal output text; `esyscmd`, and therefore `_ivdate`, expands to the result of running the command, like `` `...` `` in many Unix shells.

Discovering the existence of `esyscmd` was the most time-consuming part of handling this.

6.2 When a comma is not a comma

Our most significant complication related to m4 was handling commas in arguments to macros. For example, we defined a command `_title` to typeset its argument as a title: it produces `\textsl{...}` for TeX, and `<i>...<i>` for HTML. The complication is when the argument contains a comma, as in `_title(The _LaTeX Companion, Second Edition)`.

m4 uses the comma to separate arguments; thus, the input `_title(A, B)` passes two arguments, '`A`' and '`B`' to `_title`. We don't want that. We want to pass one argument: '`A, B`'. The way this is done in m4 is to use quoting to make the argument into one string: `_title([[A, B]])`.

So, we added the `[[...]]` sequence where it was needed, and blithely continued on. Then one day we happened to notice that the '`, Second Edition`' in the above title had disappeared from the output we were proofreading, and turned out to be missing in the generated TeX file. Being human, we had failed to add the m4 quoting and had instead written exactly the example above; m4 happily discarded the unused second argument that it saw.

Clearly we had to do an automated check. Besides titles, we had similar m4 commands to do

Figure 1: Our cover design.

footnotes, quotations, et al., which took lengthy arguments often containing commas. Since these commands could easily be split over input lines, we couldn't use grep or another line-oriented utility.

We implemented the check with the following code in m4 (and found a dozen or so places where arguments containing commas had not been quoted, so it was well worth it):

```
define([[_title]],[[dnl
  \ivtitle{$1}_iv_check_empty([[$2]])]])dnl
...
define([[_iv_check_empty]],[[dnl
ifelse($1,,,[[dnl no-op if arg is empty, else:
\misquote
errprint(__file__:__line__: misquoted $1.
)]])dnl end of ifelse
]])dnl  end of _iv_check_empty
```

Explanation: each command such as `_title` passes its *second* argument to the helper `_iv_check_empty`, which then checks if its argument is empty and complains if not, using the GNU m4 `errprint` builtin command. (It took some time to get the m4 quoting right in the definition.) The `\misquote` command in the body is not defined in TeX, and is there just to make sure we can't run the book without fixing these problems.

All in all, m4 proved sufficiently flexible that we were glad to have chosen it.

7 Our cover

Our cover design (Figure 1) was inspired by the dust cover of the *Mathematical People* mentioned in section 1. That dust cover also included several photographs of interviewees, and the names of the interviewees were subtly listed on the cover. We did not match the subtlety, opting for a simple list on the back cover. We also mimicked that book's title.

The figure here is printed in black and white for *TUGboat*; for the color version that the book was

actually published with, see tug.org/interviews/book/cover.pdf.

Dave did an early mockup of our cover using Illustrator, which he has used to create previous book covers. However, given the simplicity of our cover, we quickly decided to do the final version in TEX, and we copied the process of Yuri Robbers and Annemarie Skjold from issue 2007-1 of *The PracTEX Journal* for creating a cover using PSTricks (tug.org/pracjourn/2007-1/robbers).

Dave didn't want to take the time to figure out how to make the PSTricks gradient capability not crash on the computer he was using, so he still used Illustrator to create our cover's background gradient. He also used Illustrator to place a transparent version of the TUG logo for the front cover and spine on the background gradient rather than trying to figure out how to do transparency with PSTricks. He used Photoshop for various manipulations of the cover photos.

Our PSTricks file for the cover is at tug.org/interviews/book/cover.tex. If you compare it with the PSTricks code in the Robbers-Skjold paper, you will see how closely we followed their template. The only addition we made was for our box of interviewee names on the back cover.

8 Development environment

We did much of the coordination of use of files in this project using Subversion. Although initially not sure that we needed a version control system, we were ultimately glad to have the notion of one "reference" master which we could both synchronize to and be sure we were working from the same point.

Dave works on Windows, and accessed the TUG server using programs Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty) for basic command line work and WinSCP (winscp.net) for GUI work, among others. Karl works on GNU/Linux, and used the ubiquitious ssh and scp from OpenSSH (openssh.org) for server access.

9 Self-publishing

Self-publishing is becoming more widespread (www.walden-family.com/self-publishing/). Desktop publishing technology and Internet sales technology allows anyone who wants to make the effort to be his own publisher. There are two primary advantages: (1) the authors control the rights to the book and all decisions about how the book is published and distributed; (2) the authors don't have to find a "real" publisher and deal with publisher requirements about the design and content of the book. The individual or company which acquires and owns the ISBN (International Standard Book Number) for a book is the publisher of record for the book. The disadvantages of self-publishing are obvious: you don't have any of the guidance and services a publisher can provide and have to do everything yourself (or pay to have it done). We decided to self-publish.

Of the services usually provided by a publisher, we acquired the ISBN (in TUG's name) and we felt confident about doing our own book design, typesetting and photo adjustments, and Barbara Beeton helped with editing. Between us we also had considerable experience producing, self-publishing, and distributing several books and a journal (*TUGboat*) via a variety of paths. Additionally, our plan was to submit the book to Lightning Source Inc. (LSI) for on-demand printing and access (via LSI's parent company Ingram Book Group, a giant book wholesaler and distributor) to large (e.g., Amazon) and small on-line (and any other) book stores.

Here's how self-publishing via LSI usually works. You submit a PDF file of the interior of the book and a PDF file of the cover formatted according to LSI's standards. This costs about $120, and they send back a bound proof to review. You tell LSI the list price and a discount (typically 55% if you want good treatment by Amazon) for the book.

Thus, for instance, Dave's previous book had a list price of $30 and a discount down to a wholesale price of $13.50. When Amazon sells the book, it collects their price (typically discounted from list, say to $25) plus shipping costs from the buyer. Amazon keeps the difference between $25 and $13.50 for itself and sends $13.50 plus the money for shipping to LSI. LSI prints a copy of the book and ships it to the buyer, and deposits to Dave's bank account (a couple of months later) the difference between $13.50 and its print cost (which is about $6.25 for one copy of a 280-page 6×9 inch book).

Dave can also take a book order for $30 plus shipping via his web site and PayPal, place an order to have the book printed and shipped to the customer by LSI, paying LSI (immediately) the print cost plus shipping cost, and end up keeping something like $16 for himself. Given the price difference, and people's desire to deal with a known entity such as Amazon, Dave sells very few copies of his book via his web site and, thus, ordering and shipping the orders that come via his web site is not an undue burden.

While we want the interview book to be available via Amazon and other retail book stores, we also want to sell the book to members at a very modest price (one that does not allow a 55% discount between the list and wholesale prices), and we want to sell the book at essentially cost to interviewees. It's

easy enough to use the LSI–Amazon path for non-discounted sales. It is also easy enough to take orders via the TUG web site for members and interviewees, as long as the book is not so popular with TUG members that the volume of placing the individual orders on LSI's web site for drop shipping to members doesn't become be a burden (although LSI supports automated electronic orders, it isn't available to low-volume publishers). We'll wait and see what our sales volume is to TUG members and what ordering options we can work out for LSI; making the book available via LSI does not preclude us from having the book printed by a second company that might lead to a simpler procedure for processing TUG member orders.

10 Going forward

We will continue to use HTML for the web site rather than the LaTeX-based PDFs because HTML is by far the more effective format on the web. With all interviews in m4, it is now possible to use an expanded set of m4-to-html definitions (`tug.org/interviews/book/htmldefs.m4`, which will undoubtedly continue to evolve over time) to more or less automatically regenerate the HTML-based web site using the interviews improved with Karl and Barbara's editing and the new endnotes from a few of the interviewees.

Now that this book has been published, we will continue to do interviews for the web site, using the various tools and processes we have developed as part of this book project. Maybe there will be a volume 2 of the book in another few years; if so, it will be easier next time.

We continue to appreciate the way the TeX typesetting system can be smoothly integrated into a multi-step workflow process, as part of a typical distributed software development project involving multiple collaborators.

Acknowledgments

We very much appreciate the willingness of the interviewees to participate in the TUG interview series and to share their stories with the TeX community.

Barbara Beeton volunteered to edit every interview, already a great gift to our interviews project. And how often does one have a first rate editor who also knows typesetting with TeX! Our appreciation to Barbara is unbounded. Thank you, bb.

Steve Peter was a kind source of answers to many miscellaneous questions.

Members of the TUG board, to whom we first announce each new interview we post on the web site, often catch typos in interviews.

Gianluca Pignalberi and David Crossland shared interviewing duties in two different cases.

The Computer Science Department at the University of Aarhus in Denmark supports the main TeX Users Group server, where we post the interviews online and where we developed the book. Thanks to Kaja Christiansen, Michael Glad, and everyone there.

◇ Karl Berry and David Walden
`http://tug.org/interviews`

Self-publishing: Experiences and opinions

David Walden

At TUG'09 I gave a presentation on self-publishing. My reason for talking about self-publishing is that after one gets done typesetting something, e.g., a book, one needs to get it printed and distributed and self-publishing is one option. This note is a summary of that presentation.

The presentation and this note are based on my experiences. Other, more experienced, self-publishers may disagree with some of what I say. Also, things are changing very fast in the self-publishing world so what may be true today may be false or irrelevant tomorrow.

1 Types of printing and publishing

I'll start by touching briefly on offset versus digital printing.

Offset printing is the traditional way books are published in volume: typically at least a few thousand copies are printed at a time; plates are created that print many pages on a big sheet of paper; pages on a sheet are folded and cut into signatures; signatures are bound into books.

Digital printing is done commercially with bigger, faster machines that are like printing out a single copy of a book on a home laser printer on two sides of each sheet of paper.

The advantage of digital printing — so-called print-on-demand — is that a short run doesn't cost much, e.g., $3.75 for a single copy of a 200 page book, while a minimal offset run costs, say, $1,000. Also you don't have to inventory all the books from a 1,000 book run, and you can consequently update without obsoleting lots of old books. Digital printing is also good for a publisher wanting to keep a book minimally in print.

The disadvantage of digital printing is that the per-book cost of printing lots of books is not much less than printing a single book. With offset printing, the per-book price drops rapidly as you move from printing 500 to 2,000 copies to perhaps $1.50 for a run of 5,000 books.

Digital printing is still questionable for color books and books requiring very fine production, but it is getting better.

There are three kinds of publishers: traditional, subsidy, and self-publishing.

The traditional publisher is professionally staffed, may be large or small, is connected into traditional distribution channels, typically invests in the development and production of a book and pays the author a small royalty per book, historically mostly has used offset printing, and now can use digital printing to keep books "in print" forever.

Subsidy (or vanity) publishers make their money by the author paying them to develop and print a few copies of a book. Print-on-demand has allowed expansion of the subsidy publishing business because they no longer have to charge thousands of dollars to produce those few books; for instance, Lulu's business model is to publish 100 books for a million authors (rather than a million books for 100 authors).

A self-publisher is someone who is neither of the above and gets the ISBN (International Standard Book Number) and manages development and production of his or her book himself or herself. My impression is that the majority of self-publishing authors who hope to make significant money from a book do an offset run (although they may use print-on-demand for review copies).

By definition, the publisher of a book is the person or entity providing the ISBN. Traditional and subsidy publishers typically provide the ISBN. You as an individual are a self-publisher if you provide the ISBN.

Historically, the conventional publishing industry, book reviewers, book stores, and authors are dismissive of self-publishing. Some reasons for this may be: the quality of the writing or publication can be quite poor; there is confusion with the subsidy publishing industry; there isn't a connection to traditional acquisition or distribution channels; self-publishers don't like returns while the norm in the U.S. publishing industry is that all books are returnable by bookstores for full credit.

Nonetheless, self-publishing is a venerable option. Some very famous books, including a number of classics, by some very famous authors were originally self-published. For a list of examples, google on "famous books that were self-published".

2 Publishing and self-publishing economics

The rough economics for a traditional U.S. publisher might be something like the following. The cost per book for an offset run of a few thousand books might be $2 per book. The retail and wholesale prices might be $30 and $13.50 (a 55 percent discount). Of the $11.50 margin, the publisher has to cover development costs (perhaps $5,000 to $10,000), the print run, marketing, business overhead, and returns of books for full credit. The author gets perhaps $1 or $2 per book.

I have less idea about the economics of traditional publishing elsewhere in the world.

My self-published, 280-page book *Breakthrough*

Management has the following economics. I set the list price at $30 (and roughly equivalent amounts in U.K. pounds and Euros). I paid about $3,800 for editing, illustration, and proofreading. The cost of an ISBN in the U.S. is about $25 each in lots of 10.

There was no setup cost for print at Ames On Demand in Somerville, Massachusetts, and no shipping cost since the company is close to my home and I picked up the printed books with my pickup truck. The per-book print cost from Ames was about $5. I sold these books via my web site (which I already had so there was no additional cost) for $32 for shipment within the United States and $36 for shipment outside the United States. Since shipping in the United States actually cost closer to $5 and shipping to outside the United States actually cost closer to $11, I cleared between $20 and $22 on books printed by Ames On Demand and sold via my web site.

Later I paid Lightning Source Inc. (LSI) — a large print-on-demand company about which I will say more later — about $120 in setup fees to upload my interior and cover PDF files and to send me a proof of the book. LSI charged about $4.60 to print one book with a $1.50 handling charge per order of books drop shipped to the same address. Thus, I still cleared only a little less than $20 to $22 per book for orders I took via my web site and fulfilled via LSI (paying whatever shipping cost LSI quoted).

However, books available from LSI are also available via an electronic catalog to book stores around the world, for which I gave a 55 percent discount to go with my $30 list price. In this case, I received (deposited to my bank account three months in arrears) the $13.50 wholesale price minus the print cost, netting about $8. In this case, the shipping cost has nothing to do with me; it is collected by the bookstore and sent to LSI to pay for drop shipping.

With a mix of sales by Amazon and other on-line book stores and sales via my web site, I recovered my development costs with the sale of about 300 books.

Our *TEX People* book (self-published on behalf of TUG by Karl Berry and me) requires more paper and thus the per-book print cost at LSI is greater by $2 or $3. Barbara Beeton edited every interview for free (thank you, bb); and she, we, and our interviewees did our own proofreading. Books we sell via Amazon and the other on-line book stores will have a good margin (to benefit TUG). Books we sell to members at a big discount from list via the TUG web site will have a modest margin to go to TUG.

Self-publishing makes sense when:

- You want to control the book's pricing and rights, including updating, deal making with other publishers, and long term reuse.
- You can't get a "real" publisher (and don't want to use a subsidy publisher).
- A "real" publisher is willing but insists the (technical) author must buy a large number of books.
- You are not a big-name author and so aren't going to get big publisher distribution anyway
- You don't want the editorial "help" a "real" publisher wants to provide, i.e., you want to write the book you want to write.
- You are willing to forgo the "validation" of having a "real" publisher (and to receive some scorn for self-publishing).
- You have a narrow, highly targeted market.
- You can afford the development costs you find necessary to have done for you.
- You want the margin that would otherwise go to the "real" publisher and think you can make more money selling (most probably) fewer books at a higher margin.
- You have *much* time to spare from writing the book (and everything else you have to do in life).

Essentially you are paying in time and money to retain full control.

I don't know the details of Prof. Gilbert Strang's situation, but his web site (`wellesleycambridge.com`) lists seven books available via self-publishing. He gives some reasons for self-publishing his books on page 174 of *Recountings: Conversations with MIT Mathematicians* (edited by Joel Segel and published by A.K. Peters Ltd., Wellesley, Massachusetts, 2009).

> My first self-published book was [in] 1986 on applied mathematics. Essentially, I wanted the adventure. And you work so hard on the book, it's like your child, and I didn't want to put it out for adoption. Not that I had such terrible experiences, though editors and publishing houses come and go with lightning speed, as I now know ... But I thought I'd have the adventure of being in contact with people who use the book, which is still a pleasure. And also of being able to change it, being able to write conversationally if I wanted to ... And just staying with the book rather than writing it and saying goodbye ...
>
> I do [supervise the typesetting, cover design, and interior design]. Not that I know so much, but I care so much, and maybe that makes up for not knowing what you're doing. I'm still discussing the cover for the new book. I have views about it and the interior design,

and you just have freedom to try things, and innovate. It's interesting to me.

In addition to the motivations above for self-publishing, maybe Professor Strang also makes some money with self-publishing. He has a lot of students each year at MIT, and his books are highly regarded beyond MIT.

[The *Recountings* book just quoted was also inspired by the *Mathematical People* books that inspired our *TEX People* book, and I commend this book of interviews of MIT mathematicians to you.]

3 A few resources for self-publishing

I recommend first the Yahoo Self-Publishing discussion group: `finance.groups.yahoo.com/group/Self-Publishing`.

Dan Poynter's *The Self-Publishing Manual* and web site (`parapublishing.com`) provided me with good overall insight.

Pete Masterson's *Book Design and Production, A Guide for Authors and Publishers* and web site (`zeonix.com`) was where I learned about dealing with printers.

Morris Rosenthal's *Print-on-Demand Book Publishing* and web site (`fonerbooks.com`) was where I first grasped the economics of self-publishing.

Robert Bowie Johnson and Ron Pramschufer's *Publishing Basics* and Pramschufer's web site at `selfpublishing.com` was where I first learned about different types of paper.

Later I read Aaron Shepard's book *Aiming at Amazon*, but it might be a good first place to start.

John Culleton's annotated book list (available at `wexfordpress.com/tex/shortlist.pdf`) is a place to discover other books that can be helpful.

The "Reference Desk" area of Marion Gropen's web site (`www.gropenassoc.com`) deals with the business issues of self-publishing.

John Culleton and Marion Gropen are two of the moderators of the Yahoo Self-Publishing discussion group.

4 Components of self-publishing

The components listed in this section are relevant for both self-publishing and traditional publishing. I limit this discussion to my thoughts about self-publishing.

You can do any of these tasks yourself or hire them done. I do the ones I think I can do well enough or want to learn to do, and I hire someone to do the ones I don't want to spend time learning or which I think must be done by someone other than me.

Writing the book. Before you write the book, you should probably decide what your goal is and whether it makes sense to write the book and if you have a plausible path to getting it published at a cost and effort you can afford and sufficient payback, whether tangible or intangible.

If you are self-publishing, you can specify which typesetting system to use. I use some version of TEX, e.g., LATEX, memoir, or ConTEXt.

I draft in my typesetting system (some version of TEX) rather than drafting in some other editor or word processor and flowing the manuscript into the typesetting system later. It helps my motivation to keep writing if successive drafts of the manuscript, however rough or fragmentary, are nicely typeset on pages; it makes me feel closer to being finished.

Editing; illustration. I believe you must have someone else edit your writing. You simply cannot see everything that needs to be seen in your own writing.

There are various levels of editing: (a) correcting misspellings, punctuation errors, bad grammar, etc., and enforcing a particular style for titles, references, etc.; (b) improving correct but awkwardly written sentences; (c) suggesting substantive changes to the content, order of presentation, etc. You need to decide which levels of editing you want. I want all three.

If you are paying the editor yourself, you should select an editor who provides the levels of editing you want and who works with you in a way you find constructive and supportive rather than bothersome.

My management book has a large number of line drawings. I hired an illustrator to do the original creation of the drawings based on a style sheet (typeface, font size, overall size compared with page size, etc.) on which we agreed. She used Adobe Illustrator. I did corrections to the drawings using Illustrator after my editor reviewed the final draft manuscript.

Permissions. One school of thought in the self-publishing discussion group is that you should always ask permission for material you want to quote or of which you want to include a copy. Another school is that you should always ask a qualified lawyer. I don't agree with either of these ideas.

Of course, you should always give credit. Giving credit is independent of asking permission. You also have to ask permission when there is no question of "fair use". And I would ask permission if there is a question of courtesy involved, for instance, if I know the person and he or she would take offense if I didn't ask.

But, fundamentally, I believe that you need to learn enough about fair use to understand when there is a good case for fair use (which is not precisely

defined in the United States — I don't know about other countries). If you ask a lawyer, he or she will charge you lots of money and will not give a definitive answer. If you ask permission, there is a good chance the entity you ask (e.g., publisher of the book you are quoting) will charge you a permission fee whether or not fair use fits the situation. You need to be able to assess the risk of being sued even though you have a good case for fair use. If I have a good case for fair use and mine is a relatively low profile publication, I feel I am unlikely to be sued.

Interior design; typesetting. Historically book design and typesetting were separate functions. Today it seems that many (perhaps most) typesetters also see themselves as book designers.

In the self-publishing discussion group there is lots of discussion about whether or not a person new to self-publishing can do his or her own typesetting at a sufficiently competent level. To summarize, I think there is general agreement that someone can learn to do typesetting over time and that it is best not to try to do it in Microsoft Word (about which beginning self-publishers ask).

One can successfully use TEX or one of its variations to typeset a book and configure its output to be acceptable to on-demand printers and to publishers. However, the printers and publishers don't provide templates for TEX like they do for the commercial typesetting systems such as Adobe's InDesign (they do provide an EPS or PDF template or at least specifications).

The self-publishing discussion group experts admit that TEX, et al., exists and "does as good a job of typesetting as InDesign", but with few exceptions they are dismissive of the non-GUI command model of TEX and put off by its learning curve. In other words, TEX is far from the mainstream of self-publishing — basically irrelevant except for people who already know and use it. I think this is unfortunate since (a) I believe learning to use TEX well is probably no harder than learning to use InDesign *well*, and (b) I suspect a novice typesetter can do a better job of typesetting with less experience using TEX.

Obtaining the ISBN. In the United States, you get ISBNs from `isbn.org` which is operated by Bowker. As I write this in July 2009, you can buy blocks of 10, 100, or 1,000 ISBNs for about $245, $930, or $1,570 respectively. Bowker is trying to make a profit and offers lots of options (e.g., a bar code) which clutter the process of just trying to buy a block of ISBNs.

I believe that in some other countries an ISBN is free.

Proofreading; indexing. I believe that proofreading must be done by someone other than the author or editor. From my experience, you need to make clear that you only want proof reading; this is not the time you want the page proofs to be cluttered with additional editing suggestions.

I suppose I could learn to do indexing, but I am not interested in doing so. For those of use who are used to indexing commands being embedded in the TEX manuscript (such that index entries automatically get new page numbers if the pagination of the book changes), my impression is that most professional indexers don't like embedded indexing. They prefer to work with their own indexing software package using absolute page numbers from the page proofs.

Cover design. For self-publishing, where likely you will be selling books via on-line book stores rather than from the shelves of physical book stores, you need a cover design which shows well on-line and which meets any relevant standards (e.g., from Amazon and your printer).

In the self-publishing discussion group, there is lots of talk about the importance of having an expert cover designer. I am sure there are professional designers who can create wonderful covers that will somehow sell a lot of books. However, I bet the typical self-publisher cannot afford someone with as good a track record as, say, Paula Scher. I'll further bet that hiring one of the self-styled cover design experts is not worth the money compared to doing it yourself if you have the time and inclination to learn a graphics system (e.g., Illustrator, PSTricks, or Ti*k*Z — I think many self-publishers do their cover designs as well as their interior typesetting with InDesign).

I did the first of the covers in figures 1 and 2 with Illustrator and the second with PSTricks and a couple of manipulations with Illustrator. I also used Adobe Photoshop to twist the background design in example 1 and to adjust the photos in example 2.

Printing. Lightning Source Inc. (LSI) is a big print-on-demand company with a big distribution connection to on-line book stores through its parent company, book distributor Ingram. It has printing plants in the United States and the United Kingdom. It does not have a store front. Many other U.S. and Canadian printers offer print-on-demand (Pete Masterson's web site has a list). Some of them offer store fronts. Some companies that offer print-on-demand, e.g., Lulu, have some of their printing done by LSI. Last year it seemed that print-on-demand was not available much outside of the United States; it is increasingly available as time goes by. Traditional offset printers are increasingly offering print-on-demand

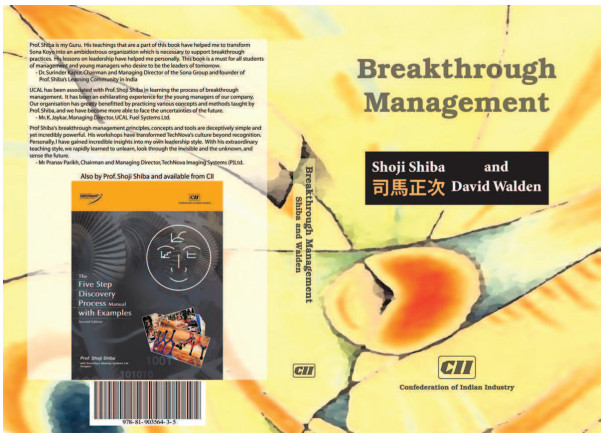


Figure 1: Cover example 1

Figure 2: Cover example 2

as a short-run option.

For my *Breakthrough Management* book, I provided interior and covers PDFs to both print-on-demand and offset printers:

- two offset runs of 2,000 each in India for sale by the organization with which my co-author works;
- LSI in the United States and United Kingdom for sale by bookstores and my web site and drop shipping from LSI;
- Ames On Demand in Somerville, Massachusetts, for inventory at my home and sale via my web site (runs of 250, 100, 50, 50, 25, ...)—I have to handle shipping for these;
- local printers of non-profit organizations in Quebec, Portugal, and Pakistan to which I have given permission to print locally and make a profit from selling my book.

Other self-publishers also use other Amazon options or Lulu, CreateSpace, etc.

Marketing. With self-publishing, you have to do all the promotional work that would be done with a book published via a traditional publisher. Review copies are easy to get using print-on-demand. You will also want to try all the modern communication tools such as blogs, Twitter, podcast, etc.

Distribution; fulfillment/shipping. With self-publishing, the odds are against you having books on the shelves of physical bookstores, except perhaps a few local or specialty stores where you have a particular connection. Nonetheless, you (or on-line bookstores) can take orders via the Internet and in theory books can be shipped around the world.

However, if you want to get books to more than a few friends or family, fulfillment is a hassle. International shipping is expensive and with tremendous variation depending on the size and weight of what is being shipped and the destination. Shipping gets really expensive if you use a trackable method; but, if you don't use a trackable method, you will probably have to absorb the cost of books lost in shipping. Unless you are ordering a large number of books a week, you have to do manual ordering via LSI's web site; electronic data interchange is not available to you. Having books printed in another country results in currency exchange costs and foreign transaction fees if you pay by credit card. If you collect payments through PayPal, they of course charge fees.

All in all, I am happiest when customers order from Amazon (or another on-line bookstore), I am not involved in fulfillment, and the difference between the wholesale price and the print cost is deposited in my bank account by LSI.

Bulk orders (where people ask for a discount) are a particular bother for me. I have to package the books for shipment from the local inventory I maintain, buyers want to be invoiced and pay by bank transfer rather than simply paying by credit card, and in some cases they do accounts-payable manipulations such as delaying payment or subtracting the currency exchange and bank transfer fees from the price I stated was after such fees.

You can hire a company to do fulfillment for you. There is also an Amazon program where you provide Amazon with an inventory of your books and Amazon does the fulfillment.

A key to making fulfillment work is to have enough margin to cover these costs in addition to the printing cost. I did this with my *Breakthrough Management* book. In the case of the TUG interviews book, *TEX People*, we are offering the book with not so much margin to TUG members, which complicates things for us.

Legal and bookkeeping issues. If you are going to self-publish and sell a lot of books, you will want

to think seriously about controlling liability through some form of incorporation. You will also want to be legal by obtaining a state business license (which also may allow you to avoid paying state sales tax to suppliers such as a printer) and by collecting state sales taxes and forwarding them to the state.

My sales in my home state of Massachusetts were not significant enough for me to seek a state business license, and I haven't thought the liability risks were big enough to justify incorporating. I pay income taxes on the difference between book revenue and book expenses (development and cost of sales) in each calendar year. I tell my accountant each year how few books I have sold within Massachusetts (countable on less than all the fingers of one hand), and he has so far told me that any sales tax is so immaterial it is not worth reporting.

5 Reflections

The lines between types of publishing continue to narrow. Traditional publishers outsource many functions and use print-on-demand to keep books in print. Authors have begun making new types of deals with traditional publishers. Every once in a while a self-published book gets picked up by a traditional publisher. The Espresso machine for printing books on demand in bookstores has signed up some traditional publishers, and it is also being used by self-publishers to print a few books for friends and family. E-books are beginning to sell in competition with printed books. All types of publishers, including self-publishers, can sell via the Internet. Traditional barriers of all types are falling.

The barriers that are not falling fast enough are the spread of print-on-demand throughout the world and the price of reliable shipping throughout the world.

My view is to use self-publishing if it fits your situation. Don't worry about traditional discrimination against self-publishing.

Self-publishing has been working for me, and I will continue to use it for books which have a narrow, easy-to-access market and for which it is worth it to me to pay the (relatively modest) development costs in order to have the books in print (and perhaps enough books will be sold to cover my costs and maybe even a little more). The list of books so far:

- Self-publishing my *Breakthrough Management* book has been very satisfactory.
- Our decision to use self-publishing (nominally) by TUG for *TEX People* still seems like a plausibly good decision. For more about that project, see the *TEX People* paper by Karl Berry and me elsewhere in this issue.
- I am currently finishing a book on the computer history of Bolt Beranek and Newman (where I worked for many years), and I will use self-publishing to produce it and make it available to prior and current BBN people (and perhaps a few technology historians will also buy copies).
- I will be reprinting oral histories of my mother and of my mother-in-law which my wife created and which were originally published using Kinko's copyshop technology and a thesis binding company.
- I am turning a photo slide show my son (a professional photographer) and I created a number of years ago into a book (typeset with ConTEXt to take advantage of grid typesetting).

Self-publishing has fit my situation. I've been happier with it than with my books which were published by a traditional publishing company. If it fits your situation, use it.

Acknowledgments

I have learned much from the on-going discussion of the Yahoo Self-publishing group and from answers to specific questions I posed to various members of the group.

Shoji Shiba allowed me to self-publish our *Breakthrough Management* book rather than seek a traditional publisher as we did with our previous books.

Karl Berry was involved every step of the way in self-publishing our *TEX People* book, and Robin Laakso is now involved in fulfillment for the book.

Barbara Beeton edited this paper for *TUGboat*.

◇ David Walden
http://walden-family.com

A short introduction to METAPOST

Klaus Höppner

Abstract

METAPOST is a program strongly related to Knuth's orginal METAFONT. It uses nearly the same graphics language and syntax, but instead of bitmap fonts it produces PostScript output. So it can be used to create high quality graphics. In METAPOST, points and paths may be described by a set of linear equations that are solved by the program. Thus, METAPOST is unique compared to other tools like PSTricks or commercial applications (e. g. CorelDraw). Additionally, the PostScript subset created by METAPOST can be interpreted by pdfTEX. So METAPOST figures can be directly included with e. g. the standard `graphics` package, while normal EPS images have to be converted first to be usable with pdfLATEX.

1 History

When Knuth developed TEX, he also created a set of new fonts, Computer Modern. For this, he created his own font description language and the program METAFONT, which converts a METAFONT source file into a bitmap, usually stored in a file with the extension `.gf` or more often `.pk`. The major feature of METAFONT is that paths may be described with a set of linear equations that determine how the single points of the path are related, and this equation set is solved by the METAFONT program. Additionally, Knuth extensively used parameters within these equations, so different font series (e. g. bold and medium) could be produced from the same equations by changing parameters.

John D. Hobby created METAPOST as a system using (nearly) the same programming language, but with PostScript output. It was presented first in *TUGboat* [2], while the first public versions were released in the early 1990s. Some new features were added to the Meta language, e. g. the ability to include stuff typeset by TEX into a METAPOST drawing (something that wasn't needed in METAFONT for creating glyphs of a font, but is very useful to put text labels into graphics).

At present, METAPOST is maintained by the METAPOST team, with Taco Hoekwater as chief developer. Since then, many improvements have been made. For the future, they plan to release MPlib, a component library that contains the METAPOST engine and can be reused in other applications.

2 Basics

As mentioned before, METAPOST defines its own programming language. It consists of the following elements:

- points,
- pens,
- paths,
- numbers,
- colors (originally RGB only; now CMYK is also supported)

Points are normally named by the letter z, represented by a pair (x, y).

Paths may contain geometrical elements (e. g. `fullcircle`) or may consist of points that are connected by lines or Bézier curves.

Colors are tuples of three (in case of RGB) or four (in case of CMYK) numbers.

For a short example let's have a look at the following example:

Listing 1: First example

```
filenametemplate "%j-%3c.mps";
beginfig(1);
pickup pencircle scaled 1bp;
draw origin--(2cm,1cm)
     ..(1.3cm,0.3cm)..cycle;
endfig;
end
```

Figure 1: Example figure, as defined in listing 1

This shows that each METAPOST figure is put between **`beginfig`** and **`endfig`**, with a number identifying the figure. So, a METAPOST source may contain several figures. Originally, when processing the source (e. g. `ex.mp`) with METAPOST (`mpost ex`), the figure numbers were used as file extensions for the resulting PostScript files. In later releases, the command `filenametemplate` was introduced, that uses a syntax something like the `printf` command in C. In the example above, we would get a PostScript file with the name `ex-001.mps` (and if we add a figure with number 2, the PostScript output would be written into `ex-002.mps`).

Since pdfTEX recognizes files with the extension `.mps` as METAPOST output, the graphic can be used in a LATEX document with a straightforward `\includegraphics{ex-001.mps}` and the document may be processed either by pdfTEX or,

using the original workflow, by compiling to DVI and using dvips.

As in C, all statements may span multiple lines and are finished by the "`;`" character.

The example figure itself shows a straight line (since two dashes were used in the source) from the origin to the point (2cm, 1cm). Then, the path is closed by a Bézier curve (because two dots were used in the path definition) via the point located at (1.3cm, .3cm). For drawing, a round pen with diameter of one PostScript point is used. METAPOST knows the same units as TEX, like bp for PS points, cm, mm or in. The result is shown in fig. 1.

3 Defining points by linear equations

While there is nothing exciting about our first example above, we will now see what makes METAPOST special. Assume you want to draw a simple rectangle. Then you know it consists of four corners (e. g. with the lower left one in the origin), that we will describe by the following equations:

Listing 2: Rectangle

```
path p[];
z0 = origin;
x0 = x3;
x1 = x2;
y1 = y0;
y3 = y2;
x1-x0 = 3cm;
y3-y0 = 2cm;
p0 = z0--z1--z2--z3--cycle;
fill p0 withcolor blue;
draw p0 withpen pencircle scaled 1bp;
```

You see, all corners except for z_0 aren't defined directly as (x, y) pairs but described by their relations. While describing a rectangle with linear equations seems rather like overkill, this METAPOST feature becomes really powerful for the construction of complex paths.

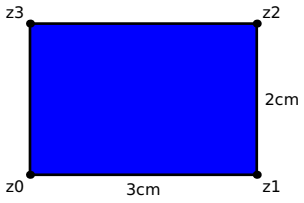


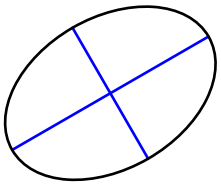
Figure 2: Rectangle, resulting from code in listing 2

4 Transformations of paths

METAPOST supports the following transformations of paths:

- Translations:
  `p0 shifted (x1, x2)`
- Rotation:
  `p0 rotated alpha`
- Scaling (in both directions, or in x or y direction individually):
  `p0 scaled factor`
  `p0 xscaled xfactor`
  `p0 yscaled yfactor`
- Slanting:
  `p0 slanted alpha`

For example, the following ellipse



is the output of the code:

```
draw fullcircle xscaled 3cm
     yscaled 2cm rotated 30;
```

5 Intersection points

Finding the intersection points of paths is another nice METAPOST feature.

Assume you have a triangle. Mathematical theory says that if you draw three lines, each of them from one corner of the triangle to the midpoint of the opposite side, all these lines will intersect at the same point.

The following code shows how this can be demonstrated in a METAPOST drawing:

Listing 3: Triangle 1

```
pickup pencircle scaled 1bp;
path p[];
z0 = origin;
z1 - z0 = 3cm*right;
z2 - z0 = 2.7cm*dir(40);
p0 = z0--z1--z2--cycle;
p1 = .5[z0,z1]--z2;
p2 = .5[z1,z2]--z0;
p3 = .5[z2,z0]--z1;
draw p1 withcolor blue;
draw p2 withcolor blue;
draw p3 withcolor blue;
draw p1 intersectionpoint p2
     withpen pencircle scaled 3bp;
draw p0;
```

This code is more straightforward than it may appear. It consists of three parts.

First, the three points $z_0 \dots z_2$ are defined and path p_0 is defined as the triangle with these points as corners.

Second, the paths $p_1 \dots p_3$ are defined. Each consists of a line from one corner to the midpoint of the opposite side, named a median of the triangle. This may be easily expressed in METAPOST, since e. g. the statement `.5[z1,z2]` is just the point on halfway along the line from z_1 to z_2.

Finally, after drawing all the paths defined above, we mark the intersection point of p_1 and p_2. This is directly given by the command

```
p1 intersectionpoint p2
```

It may be a bit more complicated if two paths have more than one intersection point.

The result of this drawing is shown in fig. 3.

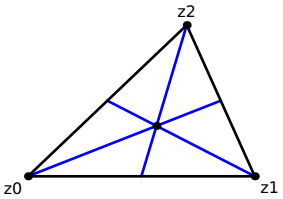


Figure 3: Construction of a triangle

6 Whatever it is ...

Coming back to the triangle in the latest example, another interesting task is the following: draw the altitude of the triangle, that is perpendicular line to the base line through the opposite vertex z_2.

Thus, the altitude line has to fulfill the following conditions:

1. It is orthogonal to the base line (connection of z_0 and z_1).
2. The starting point is in z_2, the end point shall be on the base line.

This may be directly expressed in METAPOST:

Listing 4: Triangle 2

```
z10-z2= whatever*((z1-z0) rotated 90);
z10 = whatever[z0,z1];
```

In the code above the end point of the altitude on the base line is named z_{10}.

Here we see both conditions listed before: first, the distance vector between z_{10} and z_2 is given by the distance vector between z_1 and z_0 (i. e. the base line), rotated by 90 degrees, *scaled by an arbitrary factor*.

Second, z_{10} is located *somewhere* on the line defined by the points z_0 and z_1.

In both cases, I used a numerical value named **whatever**. This may become an arbitrary number. In fact, the value may change from statement to statement, since the variable **whatever** is encapsulated per statement.

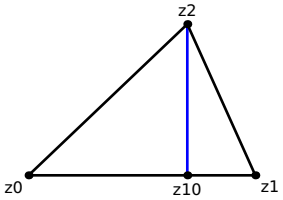


Figure 4: Triangle 2

As shown in fig. 4, METAPOST finds the correct position for z_{10} as starting point of a perpendicular line to the base line, with z_2 as end point.

7 Time variables

A path in METAPOST may be imagined as the travel of a vehicle. Paths are parameterized by a time variable (which might be a bit misleading, since of course the drawing is static). So a path has a start and end time, and any point is correlated to a time in between (and vice versa).

Here is an example where time variables are used:

Listing 5: Time variables and subpaths

```
pickup pencircle scaled 1bp;
path p[];
p0 = origin{up}..(3cm,2cm);
p1 = (-5mm,2cm)--(3cm,5mm);
draw p0 dashed withdots;
draw p1 dashed withdots;
(t0,t1) = p0 intersectiontimes p1;
draw subpath (0,t0) of p0
  -- subpath (t1,length(p1)) of p1;
```

We have two paths, p_0 and p_1: a Bézier curve from lower left to upper right, and a straight line from upper left to lower right, drawn with dotted lines.

To combine the subpath of p_0 before the intersection point with the subpath of p_1 after this point, as drawn with a solid line in fig. 5, it is not sufficient just to know the intersection point ot p_0 and p_1.

In this case, we need the time values of both paths in the intersection point. For this, the statement `p0 intersectiontimes p1` is used. The result of this is a pair (thus a point), with the time value of p_0 in the intersection point as the first part (x-part) and the time value of p_1 as the second part (y-part).

As soon as these time values are known, the desired path is constructed using **subpath**. This is a perfect example showing that METAPOST as a standalone program has full control over the paths, contrary to other tools like PSTricks that let PostScript do the job of drawing the paths.

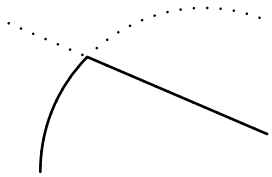


Figure 5: Using time variables and subpaths

8 Text and labels

METAPOST supports placing labels into a figure. In the simplest form, the text may be included directly, without any typesetting done by TEX:

```
defaultfont := "ptmr8r";
defaultscale := 1.2;
label("this is a label",z0);
```

It will just add the text commands to write the label text in 12pt Times Roman (a font scaling factor of one refers to 10pt) into the PostScript code. Please note that `:=` is used in the code above, since new values are assigned to the variables, while `=` is used in linear equations.

The action of placing a label at z_0 in the example is rather straightforward. The **label** command centers the label at the given point. In many cases, a suffix is appended to the **label** command to define how the label is placed in relation to the given point, i. e. **top**, **bot** (bottom), **lft** (left), **rt** (right) or **ulft**, **llft**, **urt**, **lrt** (e. g. **ulft** means upper left and **lrt** means lower right). The **label** command may be replaced by **dotlabel**, that draws a dot at the given point in addition to the label. For example, the code

```
dotlabel.urt("this is a label",z0)
```

draws a dot at z_0 and places the given text in upper right direction from z_0.

Only simple text may be used for labels that are included by METAPOST directly. But for real typesetting, we may use one of the best typesetting programs we know, TEX itself. We may include nearly arbitrary TEX stuff into labels. All TEX snippets that occur in the METAPOST source are extracted and typeset with TEX, before the result is included into the figure by METAPOST.

All the TEX stuff has to be embedded into an environment **btex ... etex**, e. g.

```
label(btex $z_0$ etex, z0)
```

will center the label "z_0" at this point. Normally, the plain TEX compiler is used for typesetting the **btex ... etex** fragments. But the name of the program may be passed on the command line when calling METAPOST, so to use LATEX we can give the command

```
mpost --tex=latex ex1
```

Let's have a look how we can typeset a label with LATEX, using Euler math fonts to typeset a formula:

Listing 6: Typesetting a label with LATEX

```
filenametemplate "%j-%3c.mps";
verbatimtex
\documentclass{article}
\usepackage{euler}
\begin{document}
etex
beginfig(1);
dotlabel.urt(
  btex $\sqrt{\frac{1}{1+x^2}}$ etex,
  origin);
endfig;
```

Since typesetting with LATEX requires a preamble loading a document class and maybe some extra packages, the example contains an environment

```
verbatimtex ... etex
```

which is included as verbatim code before typesetting *all* the labels. In this example, we load the `article` class and the `euler` package. While we had to explicitly write the `\begin{document}`, the closing `\end{document}` is inserted automatically!

The result of this code, when compiled by METAPOST with LATEX used as typesetter — as explained above — is shown in fig. 6.

Figure 6: Using LATEX for typesetting a label in Euler

Including labels causes some difficulties with fonts. Normally, METAPOST doesn't embed fonts but just adds a reference to the used fonts into the PostScript output. This isn't a problem when METAPOST is included in a TEX document, since TEX will resolve all of these font references. But the figures won't be usable standalone, since PostScript interpreters like Ghostscript will complain about unknown fonts.

In recent versions of METAPOST it is possible to run METAPOST in a mode that will produce

standalone PostScript output that contains a "real" EPS with all fonts embedded, that can be displayed in any PS interpreter or may be used in other applications besides TEX documents.

A switch named **prologues** defines whether METAPOST will embed fonts or not. The definition `prologues:=3;` at the start of your METAPOST file will produce a standalone EPS figure. The default value of `prologues` is 0, which means that no fonts will be embedded. The meaning of other values of `prologues` may be looked up in the METAPOST manual; they are relevant only for special cases.

9 Loops

The METAPOST language offers the usual features of programming languages, like macros, loops and conditional expressions. For illustration I present an example where a path is constructed within a loop (the result is shown in fig. 7):

Listing 7: Typesetting a label with LATEX

```
z0 = 2cm*right;
draw origin withpen
    pencircle scaled 2bp;
pickup pencircle scaled 1bp;
draw
   for i:=0 upto 5:
      z0 rotated (i*60) --
   endfor
  cycle;
```

The syntax of the **for** loop is quite easy to understand, it just uses a variable *i* that is incremented stepwise from zero until 5. The loop is expanded *within* the definition of the path to be drawn. Please note that the loop is ended by **endfor** without a semicolon. If a semicolon were present, it would be interpreted as end of the draw statement, leading to a syntax error.

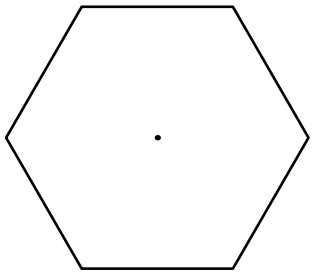


Figure 7: Figure constructed by a loop

10 Conclusion

This article was intended to just give a short introduction to METAPOST. I left out several things, e. g. how to use colors, defining macros, conditional expressions, etc. Since the article is originally based on a talk that was part of a comparison of tools, it is focused on what makes METAPOST unique among other drawing tools: solving linear equations and having direct access on intersection points and time variables of paths.

If you are interested in learning METAPOST, please have a look into the METAPOST manual [3] and/or the *LATEX Graphics Companion* [1], which describes METAPOST (among many other tools).

References

[1] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voss. *The LATEX Graphics Companion, 2nd Edition*. Addison-Wesley Professional, 2007.

[2] John D. Hobby. A METAFONT-like System with PostScript Output. *TUGboat*, 10(4), 1989. `http://www.tug.org/TUGboat/Articles/tb10-4/tb26hobby.pdf`.

[3] John D. Hobby. METAPOST — A User's Manual, 2008. `http://www.tug.org/docs/metapost/mpman.pdf`.

◇ Klaus Höppner
Haardtring 230 a
64295 Darmstadt, Germany
`klaus dot hoeppner (at) gmx dot de`

A Ti*k*Z tutorial: Generating graphics in the spirit of TEX

Andrew Mertz & William Slough

Abstract

Ti*k*Z is a system which can be used to specify graphics of very high quality. For example, accurate placement of picture elements, use of TEX fonts, ability to incorporate mathematical typesetting, and the possibility of introducing macros can be viewed as positive factors of this system. The syntax uses an amalgamation of ideas from METAFONT, METAPOST, PSTricks, and SVG, allowing its users to "program" their desired graphics. The latest revision to Ti*k*Z introduces many new features to an already feature-packed system, as evidenced by its 560-page user manual. Here, we present a tutorial overview of this system, suitable for both beginning and intermediate users of Ti*k*Z.

1 Introduction

PGF, an acronym for "portable graphics format", is a TEX macro package intended for the creation of publication-quality graphics [16]. The use of PGF requires its users to adopt a relatively low-level approach to the specification of a graphical image. To sidestep the associated verboseness of this level, a front-end named Ti*k*Z is also available for use. The syntax of Ti*k*Z borrows ideas from METAFONT, METAPOST, PSTricks, and SVG: its ultimate aim is to simplify the task of specifying graphics.

Users seeking authoritative documentation of Ti*k*Z are well advised to consult its thorough reference manual [15]. Those with an interest in discovering the possibilities of this system may wish to peruse the *TEXample* website [3], a repository with many examples of graphics created with Ti*k*Z accompanied by the associated code. First-time users ready to "take the plunge" with Ti*k*Z may benefit from introductory-level information found in [11] and [18], for example.

Our current purpose is to expand on our earlier treatment of the use of Ti*k*Z. In the intervening years since [11] appeared, a number of developments have taken place. For example, many new capabilities — such as the inclusion of a mathematics engine — are now available within Ti*k*Z. Another interesting development is the appearance of "third-party" packages which extend Ti*k*Z to specialized domains, such as the creation of combinatorial graphs or electrical circuits. A third development is the appearance of other software, such as dynamic geometry systems, which can export in PGF and/or Ti*k*Z formats.

```
\documentclass{article}
...
\usepackage{tikz}
% Optional libraries:
\usetikzlibrary{arrows, automata}
...
\begin{document}
   ...
   \begin{tikzpicture}
   ...
   \end{tikzpicture}
   ...
\end{document}
```

Listing 1: Layout of a document which uses Ti*k*Z.

2 Some Ti*k*Z fundamentals

Ti*k*Z provides support for various input formats, including plain TEX, LATEX, and ConTEXt. The requirements for each of these are fairly similar, so we focus on just one of these, LATEX, for simplicity.

Listing 1 illustrates the layout for a LATEX document containing a number of Ti*k*Z-generated graphics. In the preamble, the `tikz` package is specified, optionally followed by one or more Ti*k*Z libraries. Each graphic to be generated is specified within a `tikzpicture` environment.

Exactly which libraries are needed depend ons the requirements of the images being generated. For the simplest diagrams, no library is required. Other situations which utilize features from the various Ti*k*Z libraries require their explicit mention. For example, in listing 1, the `arrows` and `automata` libraries are referenced to gain access to a variety of arrow tips and obtain the ability to draw finite-state automata.

Specifications for the desired graphic appear within the `tikzpicture` environment. One of the simplest commands available is the `\draw` command which, when coupled with the `--` operator, joins points with straight line segments. This syntax is inspired by METAFONT and METAPOST. Figure 1 shows how a diamond can be drawn with a single `\draw` command, joining the four points on the x and y axes one unit from the origin. Since no dimensional units are provided, the default, one centimeter, is used. The `cycle` keyword is shorthand for the first point on the path. The `\fill` command used here fills the interior of a circle centered at $(0, 0)$ with a one point radius.

In this first example, Cartesian coordinates, illustrated in Figure 2, have been used. An alternate approach is to use polar coordinates, as shown in Figure 3. Angles are specified in degrees, although the inclusion of an `r` suffix can be used to indicate radian

```
\begin{tikzpicture}
  \draw (1,0) -- (0,1)
        -- (-1,0) -- (0,-1) -- cycle;
 \fill (0,0) circle (1pt);
\end{tikzpicture}
```

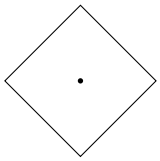


Figure 1: Drawing a diamond with a closed path, using points specified with the usual Cartesian coordinate system. The origin is shown with a filled circle.

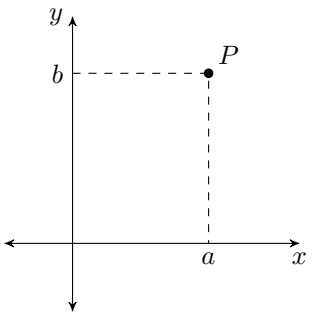


Figure 2: Using Cartesian coordinates, the point P is denoted (a, b).

measure. Thus, the point with Cartesian coordinates $(0, -1)$ can be denoted in Ti*k*Z in a number of ways: `(0,-1)`, `(270:1)`, and `(3/2 * pi r:1)`. In the last case, in addition to specifying radian measure we are making use of the arithmetic expression capabilities within Ti*k*Z to compute the value of $\frac{3}{2}\pi$.

A variety of options can influence the outcome of the `\draw` command. These options control such things as the pen color and width, whether or not to fill or shade the interior, and what line style is to be used — solid or dashed, for instance. These options are enclosed within square brackets and serve to modify the `\draw` command. Figure 4 provides an example of three `\draw` commands with a few options in effect. Multiple options, separated by commas, may appear.

So far, we have seen how the Ti*k*Z `--` operator can be used to draw line segments. There are other operators, including `grid`, `circle`, `rectangle`, and `arc`, which can be used to draw other shapes. For `grid` and `rectangle`, two opposing points of the desired shape are given. A circle is obtained with `circle`, which takes a center point and the radius; an ellipse requires a center point and two radii. A circular arc is specified by giving a starting point and

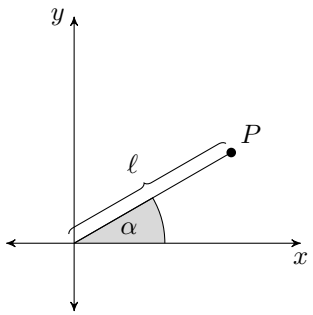


Figure 3: Using polar coordinates, the point P is denoted $(\alpha : \ell)$.

three values: two angles and a radius. Starting from the given point, an arc with the specified radius which sweeps between the two angles is drawn. Figure 5 shows a few examples of these operators.

The `\coordinate` command provides a handy mechanism to name a point. This is especially useful if the point is to be referenced more than once, since its definition is only needed once and referred to by name thereafter. Even in cases where only a single reference is needed, readability can be improved with the introduction of names. The Cartesian point (a, b) can be given the name `P` using the command

```
\coordinate (P) at (a,b);
```

Similarly,

```
\coordinate (P) at (α:ℓ);
```

names a point with polar coordinates. Once the coordinate P has been defined, it can appear in a subsequent `\draw` command as `(P)`, whereupon its defined value is used. For example, the diamond of Figure 1 can also be obtained with the code shown in Listing 2:

```
\begin{tikzpicture}
   % Define four points
   \coordinate (P0) at (1,0);
   \coordinate (P1) at (0,1);
   \coordinate (P2) at (-1,0);
   \coordinate (P3) at (0,-1);

   % Draw the diamond
   \draw (P0)--(P1)--(P2)--(P3)--cycle;
\end{tikzpicture}
```

Listing 2: Drawing a diamond using named points.

The `\node` command extends the idea of a coordinate by associating shapes, such as circles and rectangles, and labels with a specified location. For example,

```
\node (N) at
   (0,0) [draw, shape=circle] {$v_0$};
```

```
\begin{tikzpicture}
  \draw [thick, dotted]
     (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
  \draw [ultra thick]
     (0:1.5) -- (90:1.5) -- (180:1.5) -- (270:1.5) -- cycle;
  \draw [dashed, thick, color=gray]
     (0 r:2) -- (pi/2 r:2) -- (pi r:2) -- (3/2 * pi r:2) -- cycle;
\end{tikzpicture}
```

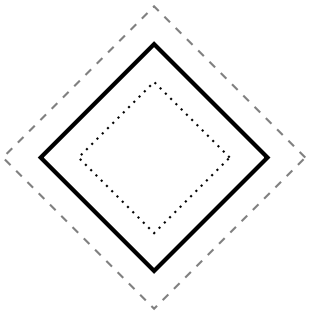


Figure 4: Drawing diamonds using Cartesian and polar coordinates, using angles specified with degrees and radians. Options to `\draw` have been introduced to change the style of the line segments.

```
\begin{tikzpicture}[scale=2/3]
   \draw (0,0) grid (4,4);
   \draw (2,2) circle (2);
   \draw (2,2) circle (1 and 2);
   \draw (0,0) rectangle (4,-1);
   \draw (0,4) arc (90:270:2);
   \fill (0,0) circle (3pt);
\end{tikzpicture}
```

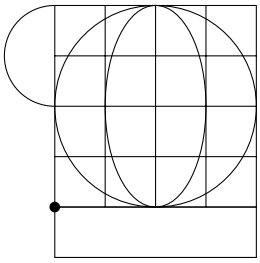


Figure 5: A sampling of Ti*k*Z path operators. As before, the origin is shown with a small filled circle. The `scale` option applied to the entire environment is used to resize the image.

defines a node named `N` which is to be placed at the origin along with a surrounding circular shape and an interior label of v_0. Like a coordinate, its name (in this case, `N`) can appear in subsequent commands. Since nodes have associated shape information, lines drawn between them don't extend to the center point, but stop at the perimeter of the shape.

If multiple nodes are defined within a graphic, it is convenient to use the `\tikzstyle` command to provide options which apply to all defined nodes. For example,

```
\tikzstyle{every node}=
   [draw,shape=circle]
```

indicates all subsequent nodes are to be drawn with a circular shape. This would allow our previous command to be abbreviated as:

```
\node (N) at (0,0) {$v_0$};
```

A complete example with nodes is shown in Figure 6.

Another capability of Ti*k*Z is the `\foreach` command, which provides a way to introduce looping actions. Listing 3 shows yet another way to obtain the diamond figure. A loop with four iterations, one for each edge of the diamond, is established. A subtlety with the parsing involved in this example requires curly braces to be used in order to group the expression corresponding to the mathematical entity $(i+1)\frac{\pi}{2}$.

```
\begin{tikzpicture}
   \foreach \i in {0,...,3}
   {
      \draw (\i * pi/2 r:1) --
            ({(\i + 1) * pi/2} r:1);
   }
\end{tikzpicture}
```

Listing 3: Drawing a diamond with a `\foreach` loop. Each iteration draws one edge of the diamond.

3 The mathematical engine

Ti*k*Z has access to a mathematical engine which provides arithmetic and relational operators, as well as a host of mathematical functions one typically encounters in a traditional programming language. The math engine can also be used independently of Ti*k*Z.

The arithmetic and relational operators are `+`, `-`, `*`, `/`, `^`, `<`, `==`, and `>`, which may appear within infix expressions in the usual manner. Here are the functions the Ti*k*Z mathematical engine supports, as of version 2.0:

```
mod   max  min   abs  round  floor
ceil  exp  ln    pow  sqrt   veclen
pi    r    rad   deg  sin    cos
tan   sec  cosec cot  asin   acos
atan  rnd  rand
```

```
\begin{tikzpicture}
   % Default actions for each node
   \tikzstyle{every node}=[draw, shape=circle];

   % Define and draw five nodes
   \node (v0) at   (0:0) {$v_0$};
   \node (v1) at   (0:2) {$v_1$};
   \node (v2) at  (90:2) {$v_2$};
   \node (v3) at (180:2) {$v_3$};
   \node (v4) at (270:2) {$v_4$};

   % Draw radial edges
   \draw (v0) -- (v1)  (v0) -- (v2)
         (v0) -- (v3)  (v0) -- (v4);
\end{tikzpicture}
```

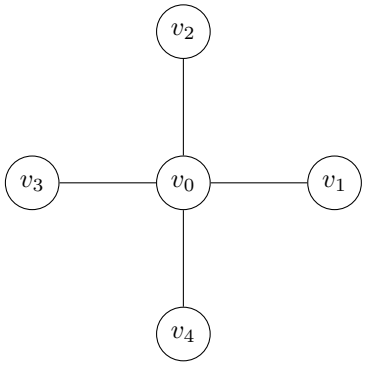


Figure 6: Using nodes.

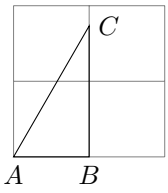


Figure 7: A 30–60–90 triangle.

As shown earlier, coordinates can be specified using arithmetic. For example:

```
\draw (0,0) -- (360.0 / 7.0 * 3.0 : 1);
```

Coordinates can also be defined in terms of mathematical functions. For example, Figure 7 shows a 30–60–90 triangle where one of the coordinates has been defined as follows:

```
\coordinate [label=right:$C$] (C) at
   (1, {sqrt(3)});
```

Note that curly braces distinguish the case where parentheses are used mathematically and the case where they denote a named coordinate.

Points can also be computed in terms of other points. Basic calculations involving coordinates such as addition, subtraction, and scaling can be performed. For such coordinate calculations, the `calc` library is required:

```
\usetikzlibrary{calc}
```

The desired coordinate calculations are then enclosed within `$` symbols. Examples of these types of calculations are illustrated in Figure 8.

Coordinate calculations can also be used to compute points that are partway between two points. The coordinate calculation

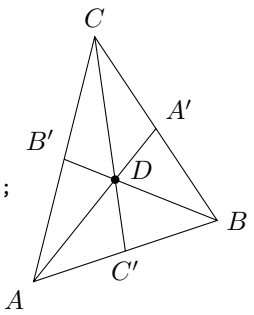
```
($(A)!0.25!(B)$)
```

becomes the point that is 25% of the way along the segment from `A` to `B`. Figure 9 shows examples of coordinate calculations being used to compute the medians of a triangle.

It is sometimes useful to be able to compute the distance between two points. Although Ti*k*Z does not currently have a direct way to do this, it is possible with the `let` operation and `veclen` function. The `let` operation allows coordinates to be defined that are available for just one path. Figure 10 demonstrates a simple use of the `let` operation; note that the macros used to name points must begin with a `p`. The `let` operation also allows extraction of the x and y components of a point. An example of this feature is given in Figure 11, which also uses the function `veclen` to compute the distance between two points.

For example, the centroid of the triangle of Figure 9 can be determined by finding the point of intersection of any two of its medians. This point, labeled D below, can be obtained with the following Ti*k*Z statement:

```
\coordinate (D) at
   (intersection of
    A--Aprime and C--Cprime);
```



4 A few Ti*k*Z libraries

Libraries, optionally loaded in the preamble section of a LaTeX document, extend the capabilities of Ti*k*Z and simplify some kinds of tasks. There are currently more than a dozen different libraries available, providing users with tools to create specific types of diagrams, such as finite-state automata, calendars,

```
\begin{tikzpicture}
   \draw[help lines] (0,0) grid (4,2);
   \coordinate[label=below:$A$] (A) at (2, 1);

   % Three points determined from a reference point
   \coordinate[label=above:$B$] (B) at ($2*(A)$);
   \coordinate[label=above:$C$] (C) at ($(A) + (-1,1)$);
   \coordinate[label=below:$D$] (D) at ($(A) - (-1,1)$);

   % Mark each point with a filled circle
   \fill (A) circle (2pt)  (B) circle (2pt)
         (C) circle (2pt)  (D) circle (2pt);
\end{tikzpicture}
```

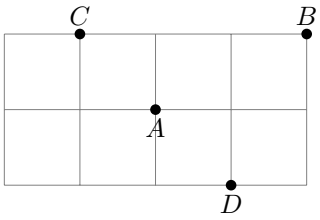


Figure 8: Using coordinate calculations.

```
\begin{tikzpicture}
   % Three vertices of a triangle
   \coordinate[label=below left:$A$] (A) at (0,0);
   \coordinate[label=right:$B$]      (B) at (3,1);
   \coordinate[label=above:$C$]      (C) at (1,4);

   % Find the midpoints
   \coordinate[label=above right:$A'$] (Aprime) at ($(B)!0.5!(C)$);
   \coordinate[label=above left:$B'$]  (Bprime) at ($(A)!0.5!(C)$);
   \coordinate[label=below:$C'$]       (Cprime) at ($(A)!0.5!(B)$);

   % Draw the triangle and its three medians
   \draw (A) -- (B) -- (C) -- cycle
         (A) -- (Aprime) (B) -- (Bprime) (C) -- (Cprime);
\end{tikzpicture}
```

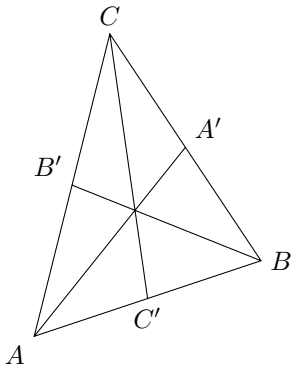


Figure 9: Using coordinate calculations to determine the medians of a triangle.

```
\begin{tikzpicture}
   \draw[help lines] (0,0) grid (2,2);
   \draw let \p1=(0,0), \p2=(1,2), \p3=(2,0) in
      (\p1) -- (\p2) -- (\p3);
\end{tikzpicture}
```

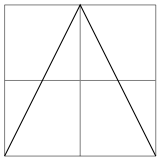


Figure 10: A simple use of the `let` operation.

```
\begin{tikzpicture}
   \coordinate [label=below:$A$] (A) at (0.5,0.75);
   \coordinate [label=above:$B$] (B) at (1,1.85);
   \draw (A) -- (B);
   \draw (A) let \p1 = ($(B) - (A)$) in circle ({veclen(\x1,\y1)});
\end{tikzpicture}
```

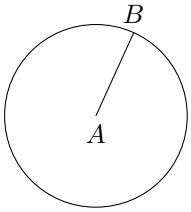


Figure 11: Using `let` to compute the distance between two coordinates.

mind maps, Petri nets, and entity-relationship diagrams. There is even one a bit more whimsical in nature, which typesets a do-it-yourself pattern for a dodecahedron, where the twelve faces can be given arbitrary content. In this section, we examine two of these libraries: `automata` and `mindmap`.

The `automata` library is used to create diagrams of finite-state automata and Turing machines. The details of one such automaton is given in Figure 12.

For each state of an automaton, the following attributes need to be specified: where on the page it should appear, what text is needed for the state name, and whether or not it is an initial and/or an accepting state. For the current example, this infor-

```
\usetikzlibrary{automata, positioning}
...
\begin{tikzpicture}[>=latex, shorten >=1pt,
                    node distance=0.75in, on grid, auto]
   % Vertices of automaton
   \node[state, initial]   (q0)              {$q_0$};
   \node[state]            (q1) [right=of q0] {$q_1$};
   \node[state, accepting] (q2) [right=of q1] {$q_2$};
   \node[state]            (q3) [above=of q1] {$q_3$};

   % Edges of automaton
   \path[->] (q0) edge [loop below] node {0} (q0)
             (q0) edge              node {1} (q1)
             (q1) edge [loop above] node {1} (q1)
             (q1) edge [bend left]  node {0} (q2)
             (q2) edge [bend left]  node {0} (q1)
             (q2) edge [bend right] node[swap] {1} (q3)
             (q3) edge [bend right] node[swap] {1} (q0)
             (q3) edge [loop above] node {0} (q3);
\end{tikzpicture}
```

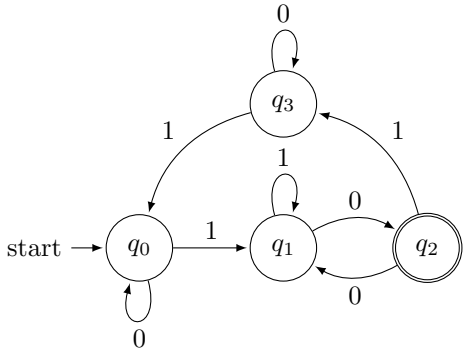


Figure 12: A finite automaton drawn with the Ti*k*Z `automata` library.

mation is provided as a sequence of `\node` commands. Information about initial and accepting states is provided as options to this command; the text appears as an argument. Layout on the page is grid based: the positions of states of the automaton are indicated relative to the location of others.

Each edge of the automaton has a source and destination node, but also requires additional information about typesetting the figure. In particular, edges can be drawn as a straight edge or with a bend; also, the edge label can appear on either "side" of its associated edge. (Imagine "driving" along an edge in the indicated direction. The edge label is either on your left or your right.)

Referring again to Figure 12, note that edge information is given in one extended `\path` command. By default, edges are drawn as straight edges, but this behavior can be modified with `bend` options. The `swap` option changes the default position of the edge label, from left to right.

Options provided to the `tikzpicture` environment specify such things as the type of arrowhead desired, the spacing of nodes on the grid, and the amount of space to leave between an arrowhead and a circular state.

The high-quality output made possible by the `automata` library can be used in conjunction with other software concerned with formal languages and automata theory. One such system is JFLAP [13], which allows users to draw and simulate automata. The creation of state diagrams with JFLAP is easily accomplished with mouse and keyboard interactions. We have implemented a translation utility [9] which converts automata stored in the JFLAP file format to Ti*k*Z format, providing an avenue for typeset output while freeing the user from knowing the underlying Ti*k*Z language details.

The `mindmap` library provides support for drawing hierarchical structures consisting of multicolored, filled shapes with text and annotations. Figure 13 shows an example of a diagram created with this library.

In our example, there is one root node, *bicycle*, with three children: *tandem bicycle*, *mountain bicycle*, and *road bicycle*. The latter child, in turn, has two children of its own. Observe how this hierarchical information is conveyed within the Ti*k*Z code: one root, with three children, and two further child nodes. The `grow` option provides an angle indicating where the child node should appear relative to its parent. For this example, we scaled the entire diagram by 65% to adjust the size of the circles. For this to be done correctly, the `transform shape` option is a crucial requirement.

5 CircuiTi*k*Z

CircuiTi*k*Z [12] is intended for drawing electrical networks and is based on Ti*k*Z, as the name implies. It is inspired by `circuit-macros` [1], a system based on the m4 macro language.* However, unlike that system, all processing is performed within the context of Ti*k*Z, so circuits can be edited directly in the TEX source document.

* As an aside, `circuit-macros` is capable of producing PGF output.

```
\usetikzlibrary{mindmap}
...
\begin{tikzpicture}[scale=0.65]
\path[mindmap, concept color=black, text=white, transform shape]
   node[concept] {bicycle}
      child[grow=230, concept color=blue!80!black] {
         node[concept]{road bicycle}
            child[grow=-120] {
               node[concept]{time trial bicycle}
            }
            child[grow=-60] {
               node[concept]{road racing bicycle}
            }
      }
      child[grow=180, concept color=red!80!black] {
         node[concept]{mountain bicycle}
      }
      child[grow=120, concept color=red!80!black] {
         node[concept]{tandem bicycle}
      };
\end{tikzpicture}
```

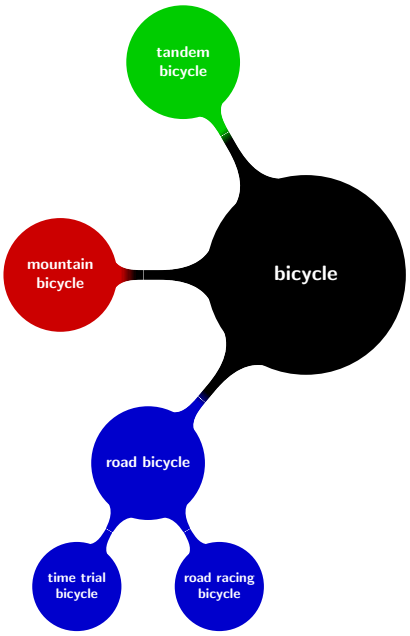


Figure 13: Example output of the `mindmap` library.

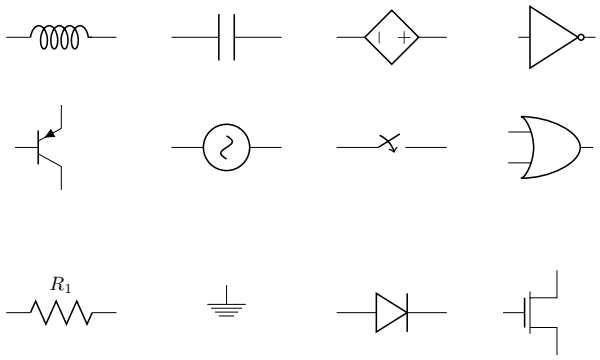


Figure 14: A sampling of circuit symbols available from CircuiTi*k*Z.

Figure 14 shows some of the symbols which are available within CircuiTi*k*Z. Symbols for both American and European electrical conventions are available.

Figure 15 shows how one might draw a diagram for an RLC circuit. As can be seen, its specification uses Cartesian coordinates and a sequence of `\draw` commands. Most of these commands specify a starting and ending point, along with a symbol (such as a resistor) to draw midway between these two points. Observe how electrical connections are not explicitly described as filled circles, but instead use the option `*-*` to indicate the connections at both ends. The grid was included as an aid to understand how the diagram was constructed. The origin for this diagram appears at the lower left corner.

6 Combinatorial graphs via tkz-graph

Combinatorial graphs, as opposed to graphs of functions, are the structures studied in the branch of mathematics known as graph theory. Drawing such graphs is an application made-to-order for Ti*k*Z. However, several other packages, `tkz-graph` and `tkz-berge` [8], are specialized for this task and provide simplifications.

Figure 16 provides a glimpse of the possibilities with the `tkz-berge` package. This package, named in honor of the mathematician Claude Berge, is primarily intended for drawing the well-known graphs in the field. Rather than explore the details of this package, we instead turn our attention to `tkz-graph` which can be used to draw arbitrary graphs.

Figure 17 illustrates a five-vertex graph drawn in three different styles. One attractive feature of this package is that it is very easy to switch from one style to another, primarily by stating the desired style as an option. As a minor complication, when vertex labels appear outside their respective vertices, as in the middle graph of Figure 17, additional information about relative location must be supplied.

Figure 18 provides the details needed to draw this graph in the "normal" style, information conveyed in the `\GraphInit` command. Mathematically, it is unimportant where each vertex appears on the page. However, in order to draw the graph, a location is needed for each of the vertices. Of the various ways allowed to specify these locations, we have chosen to use a Cartesian coordinate system. Each `\Vertex`

```
\usepackage[american]{circuitikz}
...
\begin{circuitikz}
   \draw [help lines] (0,0) grid (6,4);
   \draw (0,0) to [V=V] (0,4);
   \draw (0,4) to (6,4);
   \draw (1,4) node[above] {$I \rightarrow$};
   \draw (6,4) to [C=C] (6,0);
   \draw (4,4) to [L=L, *-*] (4,0);
   \draw (2,4) to [R=R, *-*] (2,0);
   \draw (6,0) to (0,0);
\end{circuitikz}
```

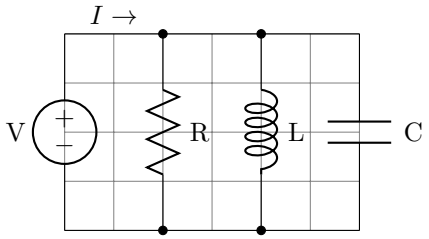


Figure 15: An RLC circuit drawn with CircuiTi*k*Z.

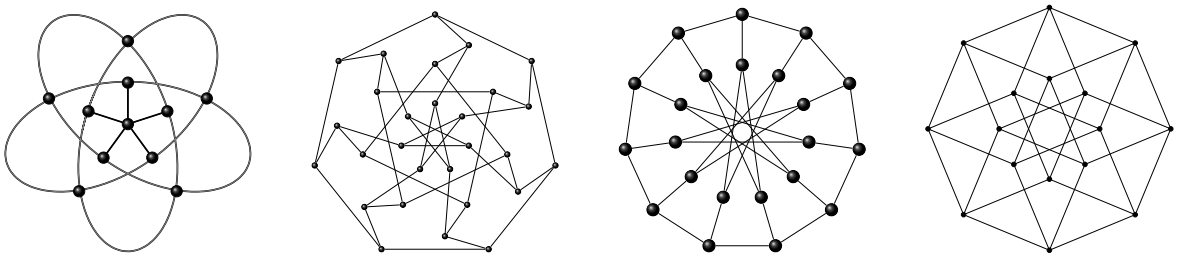


Figure 16: A brief gallery of graphs drawn by `tkz-berge`. Images created by Alain Matthes.

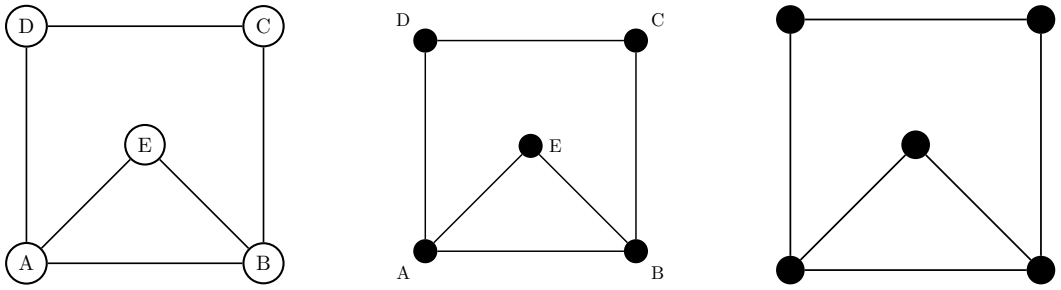


Figure 17: An undirected graph drawn with `tkz-graph` in three different styles: normal, classic, and simple.

command introduces a desired location and a vertex name. Each edge of the graph is specified with an `\Edge` command.

To change the drawing so vertex labels are omitted simply requires a modification to the `GraphInit` command:

```
\GraphInit{vstyle=Simple}
```

To change the appearance of the graph so vertex labels appear next to their respective vertices, two changes are required. First, the vertex style is changed:

```
\GraphInit{vstyle=Classic}
```

To specify each label position, another option is included in each `Vertex` command. So, for example, since vertex C is in the northeast corner relative to its vertex, we use a specification of 45°:

```
\Vertex[x=4, y=4, Lpos=45] {C}
```

A small syntactic detail is worth noting here: in Ti*k*Z, commands are terminated with a semicolon, but no semicolons are required with `tkz-graph`.

The addition of just one line to our example,

```
\tikzset{EdgeStyle/.style={post}}
```

yields a directed graph, as shown in Figure 19. Including the `label` option to each `Edge` command provides a way to describe a weighted, directed graph, as illustrated in Figure 20.

7 Two-dimensional constructions via tkz-2d

The `tkz-2d` package [7] is a collection of macros intended to simplify the construction of diagrams in a two dimensional Cartesian coordinate system. It has particular strengths in the realm of geometric constructions, as it provides higher level abstractions

```
\usepackage{tkz-graph}
...
\begin{tikzpicture}
   % Initialize tkz-graph
   \GraphInit[vstyle=Normal]

   % Vertices
   \Vertex[x=0, y=0] {A}
   \Vertex[x=4, y=0] {B}
   \Vertex[x=4, y=4] {C}
   \Vertex[x=0, y=4] {D}
   \Vertex[x=2, y=2] {E}

   % Edges
   \Edge(A)(B)  \Edge(B)(C)
   \Edge(C)(D)  \Edge(D)(A)
   \Edge(A)(E)  \Edge(E)(B)
\end{tikzpicture}
```

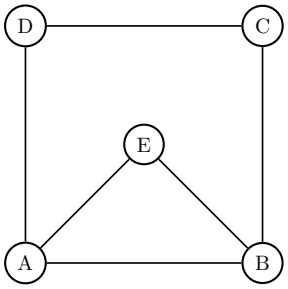


Figure 18: Undirected graph drawn with `tkz-graph`.

```
\usepackage{tkz-graph}
\usetikzlibrary{arrows}
...
\begin{tikzpicture}
   % Initialize tkz-graph
   \GraphInit[vstyle=Normal]
   \tikzset{EdgeStyle/.style={post}}

   % Vertices
   ... same as above ...
\end{tikzpicture}
```

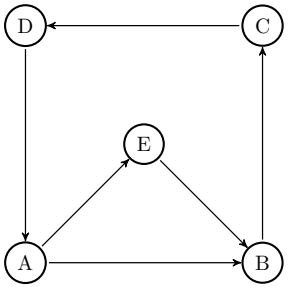


Figure 19: A directed graph drawn with `tkz-graph`.

```
\usepackage{tkz-graph}
\usetikzlibrary{arrows}
...
\begin{tikzpicture}
   % Initialize tkz-graph
   \GraphInit[vstyle=Normal]
   \tikzset{EdgeStyle/.style={post}}

   % Vertices
   \Vertex[x=0, y=0] {A}
   \Vertex[x=4, y=0] {B}
   \Vertex[x=4, y=4] {C}
   \Vertex[x=0, y=4] {D}
   \Vertex[x=2, y=2] {E}

   % Edges
   \Edge[label=$10$](A)(B)   \Edge[label=$5$](B)(C)
   \Edge[label=$20$](C)(D)   \Edge[label=$8$](D)(A)
   \Edge[label=$30$](A)(E)   \Edge[label=$16$](E)(B)
\end{tikzpicture}
```

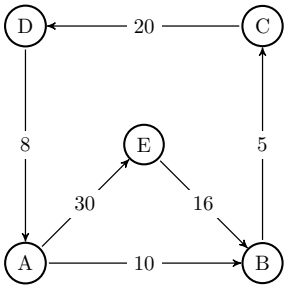


Figure 20: A directed graph drawn with `tkz-graph`.

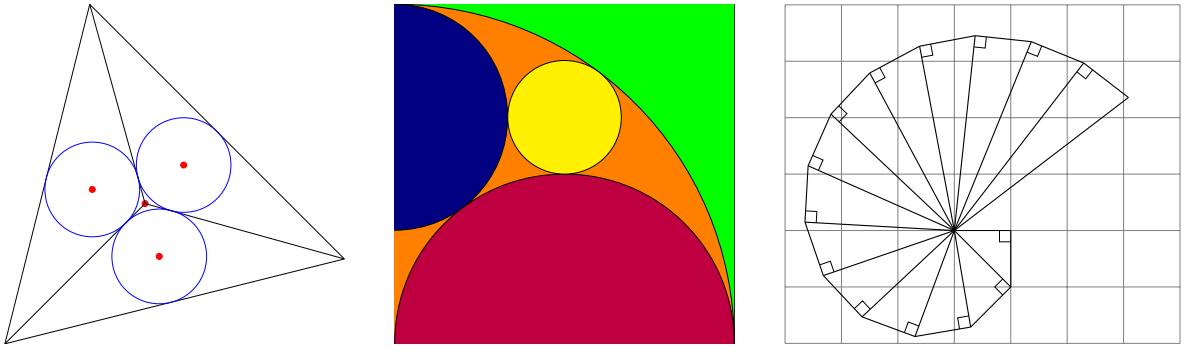


Figure 21: A brief gallery of pictures drawn by `tkz-2d`. Images created by Alain Matthes.

```
\begin{tikzpicture}
   % Initialize tkz-2d
   \tkzInit

   % Define and label two points, A and B, and a segment joining them
   \tkzPoint[pos=left](1,1){A}
   \tkzPoint[pos=right](4,2){B}
   \tkzSegment[style=thick](A/B)

   % Construct two circles, each with radius AB
   \tkzCircle(A,B)
   \tkzCircle(B,A)

   % Find and label the intersection points
   % C and D of the two circles
   \tkzInterCC(A,A,B)(B,B,A){C}{D}
   \tkzDrawPoint[pos=above left](C)
   \tkzDrawPoint(D)

   % Draw the remaining sides of the equilateral triangle
   \tkzPolySeg(A,C,B)

   % Draw a perpendicular bisector
   \tkzMidPoint(A,B){E}
   \tkzSegment[style=dashed](C/D)

   % Mark the 90 degree angle
   \tkzRightAngle(A/E/C)
\end{tikzpicture}
```

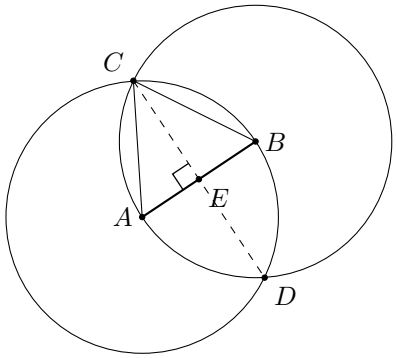


Figure 22: A construction due to Euclid, expressed with `tkz-2d`.

compared to those available to the Ti*k*Z user. The illustrations shown in Figure 21 provide a glimpse of the possibilities afforded by this package.

For an example of `tkz-2d`, we consider a geometric construction due to Euclid. In this construction, a line segment is given and the goal is to construct an equilateral triangle, one side of which is the given segment. A similar construction appears as a tutorial in the Ti*k*Z manual, although we feel the approach made possible by `tkz-2d` is more natural.

Figure 22 provides the full details of the construction. We begin by introducing two points, specifying the position and label information, using the `\tkzPoint` command. The two circles are drawn with the `\tkzCircle` command, given the center and implied radius. The most complicated statement shown here is `\tkzInterCC` which computes the intersection points of the two circles, storing their coordinates in `C` and `D`. The remainder of the construction follows easily from `tkz-2d` primitives.

A second example of the `tkz-2d` package is shown in Figure 23. Observe that the starred form

```
\begin{tikzpicture}[scale=1/2] % scaled to half-size
   % Initialize tkz-2d
   \tkzInit

   % Define three points of a 3-4-5 right triangle
   \tkzPoint*(0,0){C}
   \tkzPoint*(4,0){A}
   \tkzPoint*(0,3){B}

   % Draw the three sides of the triangle
   \tkzPolygon(C,A,B)

   % Create a 4 by 4 grid and move it down
   \begin{scope}[yshift=-4cm]
      \tkzGrid(0,0)(4,4)
   \end{scope}

   % Create a 3 by 3 grid and move it left
   \begin{scope}[xshift=-3cm]
      \tkzGrid(0,0)(3,3)
   \end{scope}

   % Create a 5 by 5 grid for the hypotenuse
   \begin{scope}[yshift=3cm, rotate=-atan(3/4)]
      \tkzGrid(0,0)(5,5)
   \end{scope}
\end{tikzpicture}
```

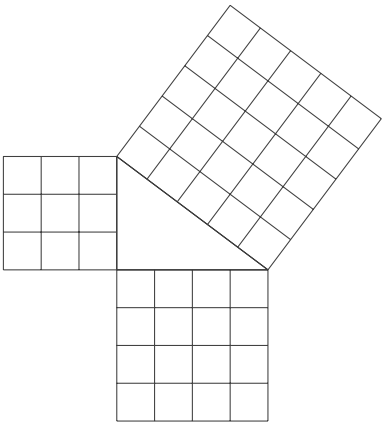


Figure 23: An illustration of the Pythagorean theorem drawn with `tkz-2d`.

of the `\tkzPoint` command used here causes a point to be defined, but no corresponding label is drawn. Another interesting aspect of this example is the use of the Ti*k*Z `scope` environment to limit the effect of the shift and rotation operations applied to each of the three grids.

A few comments about these examples. Like `tkz-graph`, semicolons are not needed to terminate commands. Unlike Ti*k*Z, the use of spaces to separate arguments within `tkz-2d` commands is *not* allowed, an unfortunate requirement in our opinion. Finally, it is possible to blend "pure" Ti*k*Z commands with `tkz-2d` commands.

8 GeoGebra and Ti*k*Z

GeoGebra [4] is a software system intended for mathematics education. Although not limited to geometry, GeoGebra is an example of an interactive geometry system. In these systems, geometric constructions can be performed using fundamental objects such as points, lines and circles. After the construction is complete, it can be modified by dragging points or moving sliders, while preserving the key geometric relationships that defined the construction.

Figure 24 shows a GeoGebra session involving the geometric construction considered in Section 7. In contrast to specifying such a construction with

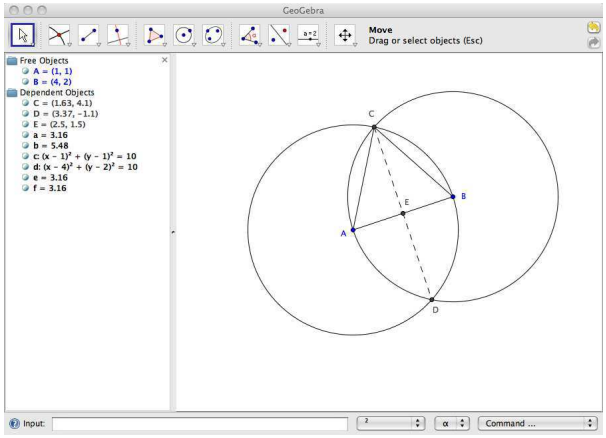


Figure 24: Screen image of a GeoGebra session.

Ti*k*Z commands, this construction was driven by the menus provided by the GUI presented by GeoGebra, involving construction choices such as "circle with center through point", "intersect two objects", and "segment between two points". In brief, GeoGebra provides for geometric constructions in a menu-driven approach, unlike the language-based approaches of Ti*k*Z and `tkz-2d`.

A relatively new feature of GeoGebra is its ability to export to the PGF/Ti*k*Z format. Once a con-

struction is complete, it can be exported to Ti*k*Z format with a simple menu choice. The resulting `tikzpicture` environment can be placed in a TEX document. This feature provides an avenue for producing high quality graphics output without the overhead of learning Ti*k*Z language details. There are currently four different output formats supported: LATEX, plain TEX, ConTEXt, and the LATEX `beamer` class. (See [10] and [17], for example, for information about Beamer.) With the `beamer` format, the geometric construction is formatted for a "play-by-play" presentation, since pauses are inserted after each key construction step.

The GeoGebra software is free — it is open source and licensed under the GNU General Public License. Since it is based on Java, it runs on many different computer systems.

9 PDF and SVG output

For many users of Ti*k*Z, the overall aim is to generate several desired graphics, merging them with text to produce one document. However, there are situations where one simply wants to generate a collection of graphics images in PDF format, one per file.

Stand-alone PDF can be generated through the use of the `preview` package [5] and `pdftk`, the PDF toolkit [14], free software licensed under the GNU GPL and available for many computer systems. A schematic of the work flow needed to produce stand-alone PDF is shown in Figure 25.

With an appropriately constructed LATEX file, `pdflatex` will generate a PDF file, where each page consists of a tightly cropped image obtained from a `tikzpicture` environment. The PDF toolkit has a "burst" mode which can then be used to obtain the desired files, one image per file.

Listing 4 illustrates how to use the `preview` package for this purpose. For some situations, producing tightly cropped graphics is a bit too aggressive. However, the `preview` package conveniently allows the amount of cropping to be specified. In this example, we have specified a two point margin around the edges of the graphic by setting the length `\PreviewBorder`.

SVG [6], *scalable vector graphics*, is a format intended for use on the World Wide Web, in large measure due to its ability to obtain scalable graphical rendering within a browser. One of the supported output formats of PGF, and therefore Ti*k*Z, is SVG. Unfortunately, as is explained in the Ti*k*Z manual, there are some serious restrictions on the types of Ti*k*Z pictures which can be converted.

Happily, there are alternate routes for producing SVG output which do not suffer from these restrictions. In fact the technique, shown in Figure 26, is very similar to that for the production of stand-alone PDF. The key difference lies in the conversion of PDF to SVG, which can be accomplished with `pdf2svg` [2], a free utility.

```
\documentclass{article}
% Use TikZ and any associated libraries
\usepackage{tikz}
\usetikzlibrary{arrows, automata}

\usepackage[tightpage, active]{preview}
\setlength{\PreviewBorder}{2pt}
\PreviewEnvironment{tikzpicture}

\begin{document}

   \begin{tikzpicture}  % First picture
   ...
   \end{tikzpicture}

   \begin{tikzpicture}  % Second picture
   ...
   \end{tikzpicture}
   ...                  % Other pictures
\end{document}
```

Listing 4: Using the `preview` package to generate tightly-cropped graphics images, one per page.

10 Summary

Ti*k*Z is a very capable system which integrates vector graphics with TEX. Since its inception roughly four years ago, it has continued to evolve, gaining new capabilities and features. Moreover, a variety of other programs which can export to Ti*k*Z and/or PGF formats provides some evidence of its acceptance and popularity in the TEX world. Although not shown in our examples, Ti*k*Z's support for color allows for very compelling graphics, especially in conjunction with Beamer documents.

For some specialized domains, such as graph theory and theory of computing, there are relatively simple techniques which utilize Ti*k*Z to produce graphics that meet or exceed the quality of figures found in textbooks and journals in those areas.

References

[1] Dwight Aplevich. M4 macros for electric circuit diagams in LATEX documents. `http://mirror.ctan.org/graphics/circuit_macros/doc/CMman.pdf`.

[2] David Barton. `pdf2svg`. `http://www.cityinthesky.co.uk/pdf2svg.html`.

[3] Kjell Magne Fauske. TEXample.net: Ample resources for TEX users. `http://www.texample.net/`.

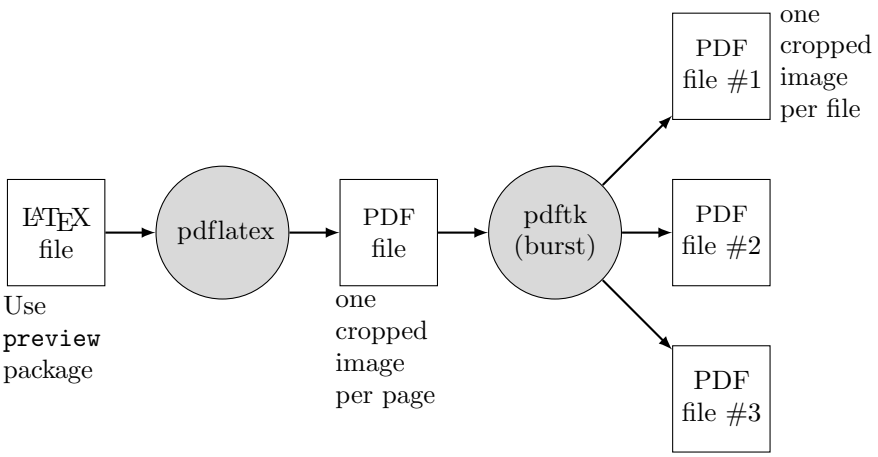


Figure 25: Processing a LaTeX file to obtain stand-alone PDF output.

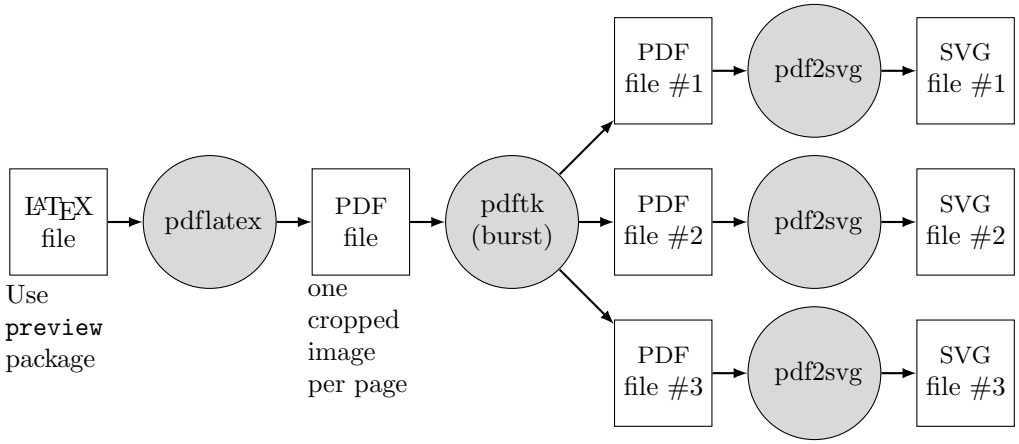


Figure 26: Processing a LaTeX file to obtain stand-alone SVG output.

[4] Markus Hohenwarter. GeoGebra. http://www.geogebra.org/cms/.

[5] David Kastrup. The preview package for LaTeX. http://mirror.ctan.org/macros/latex/contrib/preview/.

[6] Chris Lilley and Doug Schepers. W3C for the SVG working group. http://www.w3.org/Graphics/SVG/.

[7] Alain Matthes. tkz-2d. http://altermundus.fr/pages/download.html.

[8] Alain Matthes. tkz-graph and tkz-berge. http://altermundus.com/pages/tikz.html.

[9] Andrew Mertz and William Slough. jflap2tikz. http://mirror.ctan.org/graphics/jflap2tikz.

[10] Andrew Mertz and William Slough. Beamer by example. *TUGboat*, 26:68–73, 2005.

[11] Andrew Mertz and William Slough. Graphics with PGF and Ti*k*Z. *TUGboat*, 28:91–99, 2007.

[12] Massimo Redaelli. CircuiTi*k*Z. http://home.dei.polimi.it/mredaelli/circuitikz/.

[13] Susan H. Rodger. JFLAP. http://www.jflap.org/.

[14] Sid Stewart. pdftk. http://www.accesspdf.com/pdftk/.

[15] Till Tantau. PGF & Ti*k*Z. http://mirror.ctan.org/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf.

[16] Till Tantau. PGF and Ti*k*Z — graphic systems for TeX. http://sourceforge.net/projects/pgf/.

[17] Till Tantau. User's guide to the beamer class. http://latex-beamer.sourceforge.net.

[18] Zofia Walczac. Graphics in LaTeX using Ti*k*Z. *TUGboat*, 20:176–179, 2008.

◇ Andrew Mertz & William Slough
Department of Mathematics and Computer Science
Eastern Illinois University
Charleston, IL 61920
aemertz (at) eiu dot edu,
waslough (at) eiu dot edu

Medical pedigrees: Typography and interfaces

Boris Veytsman and Leila Akhmadeeva

Abstract

We discuss the history of medical pedigrees and its importance for the design of the modern pedigree drawing software.

1 Writing on the wall

The participants of TUG 2009 could see on the wall of Room 126, DeBartolo Hall, University of Notre Dame, a poster 23 feet wide. This poster depicted a pedigree of one of the authors of this paper (Leila). The pedigree included 13 generations (since the beginning of the 17$^{\text{th}}$ century), about three hundred persons in total. It was created in the following way: Leila did research in the National Archives of Bashkir Republic, interviewed her aunt and other relatives and deciphered the notes about family history dictated by her grandfather at mid-1980s. She put the data into an Excel file and periodically e-mailed the updated file to Boris, who then run it through a custom-made Makefile, getting as the result a PDF file. This file was e-mailed back. The final version was printed on two 16 foot sheets, which were taped together by Leila and her cousin in Ufa.

This project became possible due to the suite of programs for automatic drawing of medical pedigrees we have been developing since 2005 (Veytsman and Akhmadeeva, 2007a; Veytsman and Akhmadeeva, 2007c). Their previous versions were discussed at TEX meetings (Veytsman and Akhmadeeva, 2006; Veytsman and Akhmadeeva, 2007b; Veytsman and Akhmadeeva, 2008) as well as conferences on genetics (Akhmadeeva, 2007). Now that the programs are no longer at the proof of concept level, we became more interested in improving them and making them of practical use for physicians, other health care specialists, geneticists, researchers, educators, and students.

There are basically two ways to improve a typesetting program. First, we can improve its output, making the result more beautiful and clear. Second, we can improve its interface, making it more convenient, user-friendly and adding new features. In this paper we discuss both these approaches.

2 A (very) short history of pedigree typography

A usual approach to beautiful typography is the study of history. The aesthetics of a printed page is based on the tradition. Thus it was natural for us to turn to the history of pedigrees in our quest to make them beautiful. We were fortunate to find the seminal paper (Resta, 1993) discussing just that topic. Resta unearthed many early pedigrees, and this section is based primarily on his research.

One of the biggest surprises for us was the fact that pedigree drawing is a relatively modern art, started as late as the middle of the 19$^{\text{th}}$ century. One would think that people drew genealogical trees for ages, and thus pedigrees should have long history with venerable traditions. Still, this is not the case. While both genealogical trees and pedigrees are a graphical depiction of a family history, the goals of these two are completely different. A genealogical tree is intended to show the lineage of an individual, while a pedigree is used to show genetic traits. This difference in goals leads to a considerable difference in results. As an illustration of this difference, we show in Figure 1 a very nice genealogical tree created with PSTricks (Coustillas and Girou, 2004). The corresponding pedigree is shown in Figure 2. Note that the data of Figure 1 are actually not sufficient for the medical analysis, so we guessed that the marriage of Paul Joannon and Laure de Mortillet was childless. We also could not guess the relationship between the first and the second husbands of Élise Vidal; were they brothers? an uncle and a nephew? — so we decided against putting them on the chart.

Even a casual glance at these two pictures shows the difference between the presentations. Clearly, the typographical traditions of genealogical trees is not very helpful for pedigrees.

One of the first known medical pedigrees was published by Earle in the middle of the 19$^{\text{th}}$ century. He studied cases of color blindness in his own family. The pedigree included only females. Earle found out that his printer used to typeset musical sheets, so taking musical notes for symbols was a natural decision. Unaffected females were represented by half note symbols, while affected ones were represented by quarter-notes. Figure 3 shows the result.

This nomenclature was apparently not used by anybody but Earle himself. Other scientists chose other ways to represent pedigrees. An influential researcher of the latter part of the century, Francis Galton, in his books and papers published in 1869–1904 dispensed with drawing pedigrees, and used intricate tables instead (Resta, 1993).

The rediscovery of Mendel's seminal works on plant genetics might be the cause of the adoption of the symbols of Mars (♂) and Venus (♀) for males and females. A typical (hand-drawn) pedigree using this nomenclature is shown in Figure 4. It was

Affinité d'Élise Vidal avec Sabine Vigière d'Anval, sœur de Paul
et avec Laure de Mortillet, sœur de la fiancée de Paul

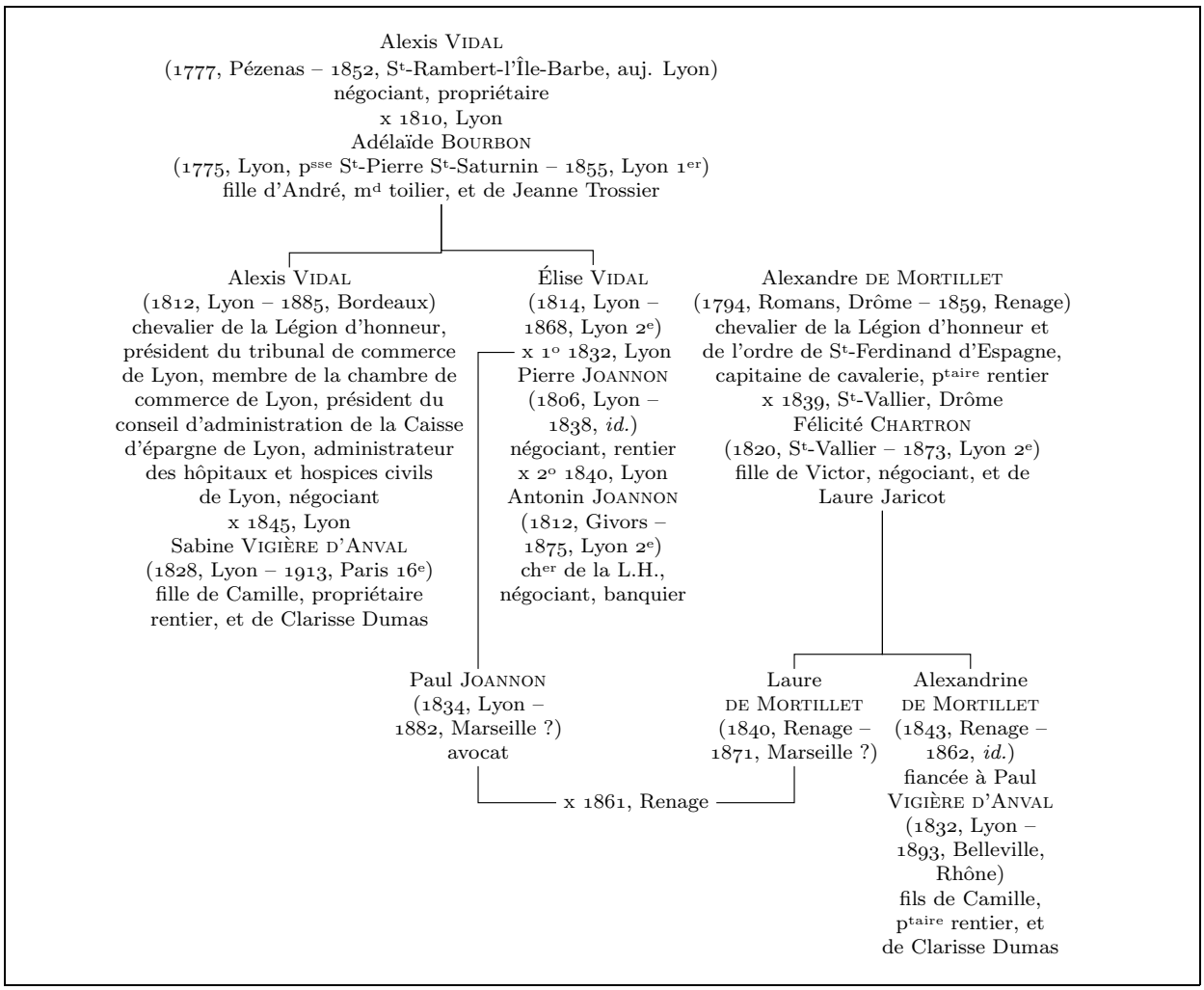


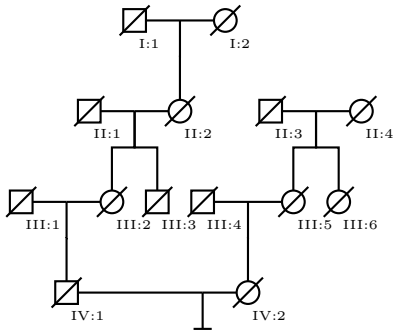
Figure 1: A genealogical tree from Coustillas and Girou, 2004

proposed by Pearson based on Galton's ideas. Karl Pearson, widely known now for his works in mathematical statistics, was a student of Galton, and later took his position as the director of the Laboratory for National Eugenics, UK.

It should be said that the relationship between eugenics and modern genetics resembles the relationship between alchemy and chemistry or astrology and astronomy. We no longer believe that we can breed a better human like we breed better dogs or chickens: people are more complex beings. We also do not believe that stars can be used to predict one's fate or that base metals can be transmuted to gold without a nuclear reactor. Nevertheless the methods and nomenclature developed by astrologers and alchemists are widely used today in astronomy and chemistry. Similarly the methods and nomenclature developed in eugenics have found their places in modern genetics.

Later Pearson and the Laboratory for National Eugenics used stylized symbols for males and females, only distantly resembling the original astronomical signs. A typical pedigree in this style is shown in Figure 5.

This nomenclature was used in medical and genetics journals up to the 1970s, usually as an alternative to the modern one. Sometimes the disputes between the proponents of these were rather bitter. Resta cites different possible reasons why Pearson-Galton symbols did not survive. We would suggest



I:1 André Bourbon; né: ignoré; âge au décès: ignoré.

I:2 Jeanne Trossier; né: ignoré; âge au décès: ignoré.

II:1 Alexis Vidal; né: 1777; âge au décès: 75; négociant, propriétaire.

II:2 Adélaïde Bourbon; né: 1775; âge au décès: 80.

II:3 Victor Chartron; né: ignoré; âge au décès: ignoré; négociant.

II:4 Laure Jaricot; né: ignoré; âge au décès: ignoré.

III:1 Pierre Joannon; né: 1806; âge au décès: 32; négociant, rentier.

III:2 Élise Vidal; né: 1814; âge au décès: 54.

III:3 Alexis Vidal; né: 1812; âge au décès: 73; chevalier de la Légion d'honneur, président du tribunal de commerce de Lyon, membre de la chambre de commerce de Lyon, président du conseil d'administration de la Caisse d'épargne de Lyon, administrateur des hôpitaux et hospices civils de Lyon, négociant.

III:4 Alexandre de Mortillet; né: 1794; âge au décès: 65; chevalier de la Légion d'honneur et de l'ordre de S^{t}-Ferdinand d'Espagne, capitaine de cavalerie, p^{taire} rentier.

III:5 Félicité Chartron; né: 1820; âge au décès: 53.

III:6 Alexandrine de Mortillet; né: 1843; âge au décès: 19; fiancée à Paul Vigière d'Anval.

IV:1 Paul Joannon; né: 1834; âge au décès: 48; avocat.

IV:2 Laure de Mortillet; né: 1840; âge au décès: 31.

Figure 2: A pedigree based on the data of Figure 1

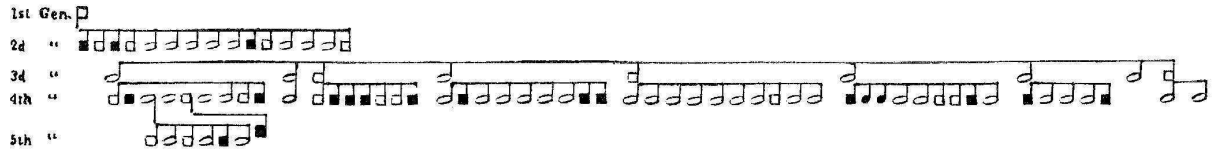


Figure 3: One of the first pedigrees, Earle (1845). From Resta, 1993

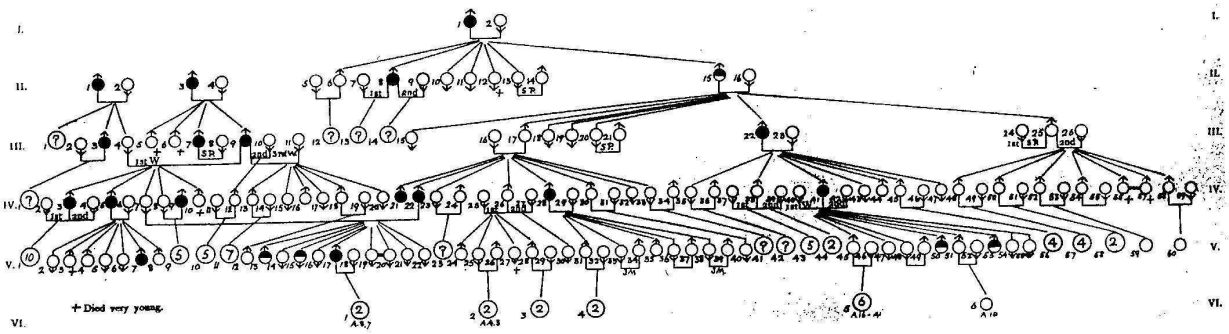


Figure 4: A pedigree using Mars and Venus symbols, Pearson (1912). From Resta, 1993

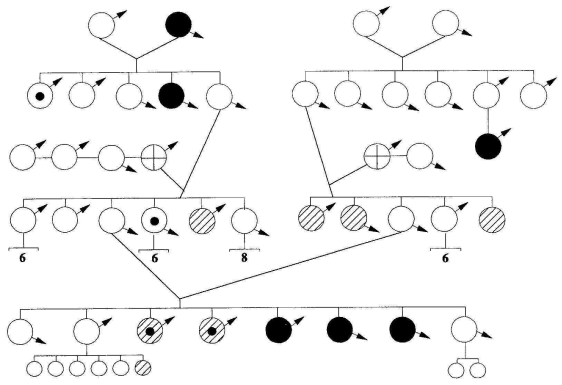


Figure 5: A pedigree using stylized Galton-Pearson Symbols (Mott, 1910). From Resta, 1993

that the problem was in the legibility; sometimes it is difficult to see the thin arrows used to differentiate between males and females on the diagrams, especially at small sizes. The modern method of using squares and circles leads to much better legibility. One of the lessons of classical typography is that the more legible solution usually wins in the long run (the run may be very long, however).

A more legible way to represent pedigrees can be found in the works of German authors. A rather forceful pedigree in the German *Sippschaftstafel* style is shown in Figure 6. It is anything but illegible. Use of circles and squares makes the task of distinguishing between males and females rather easy.

This style was developed by Henry Goddard and Charles Davenport, and approved by the powerful Eugenics Records Office (ERO) in the US. ERO popularized this style in numerous publications. It provided instruction, free blank forms and stamps with circles and squares to the families willing to furnish their pedigrees to ERO. A sample pedigree from an ERO pamphlet is shown in Figure 7. It looks almost like a modern pedigree, especially if we compare it to a hand drawn one (Figure 8).

Today pedigrees in Goddard-Davenport styles are ubiquitous. It is almost impossible to find a journal in genetics, hereditary diseases or related fields without at least several papers with pedigrees. Some typical pedigrees are shown in Figures 9 and 10. Note the interesting way to show haplotypes in Figure 10.

Modern pedigrees are legible and clear. They are also beautiful. The relatively short history of medical pedigrees shows, that the quest for clarity and legibility also causes beauty, almost as a side effect. This should not be surprising for students of typographical art: we know that aesthetics of a printed copy is always related to a clear representation of the author's thoughts.

3 Interfaces

Since our programs (Veytsman and Akhmadeeva, 2007a; Veytsman and Akhmadeeva, 2007c) are intended primarily for people with minimal TEXnical background, user-friendly interfaces are important for them. There are two main items to set up for a program: *what to print* and *how to print.* Accordingly there are two major interfaces: setting up the pedigree contents and setting up the pedigree layout and appearance.

We choose spreadsheets for the first task, setting up the contents of the pedigree. A user puts the information about proband and her or his relatives in a table, like the one shown in Figure 11. Such tables can be easily prepared with popular spreadsheet programs. One can argue whether a spreadsheet is an optimal input interface, but it has the advantage of being familiar to most users. Indeed, many geneticists already store the information in spreadsheets, so the use of our programs would be natural to them.

Another advantage of this interface is the fact that tables like the one in Figure 11 can be easily obtained from SQL databases. Thus our program can be integrated with larger database-driven projects.

For the second task, setting up the layout of the pedigree, the interface is much less easy. It involves setting up configuration files, which are snippets of Perl code, as shown in Figure 12. This was a design decision: we did not foresee our users changing the styles of their pedigrees. While this conclusion might be right for some (many?) situations, it was definitely not right for the application described in Section 1: the default style was not designed for a huge pedigree with more than three hundred persons, so we ought to change the configuration file. Nevertheless we still doubt that making changes in the pedigree layout too easy is necessarily a Good Thing. Probably a better solution would be to offer a user a set of preconfigured layouts.

4 The future

Our quest for better typesetting of pedigrees surprisingly showed that the preliminary design decisions we made (Veytsman and Akhmadeeva, 2007a; Veytsman and Akhmadeeva, 2007c) were sound. Of course, there are still many ways to improve the program suites.

When this work was reported at TUG 2009, Karl Berry suggested setting up a web site, where

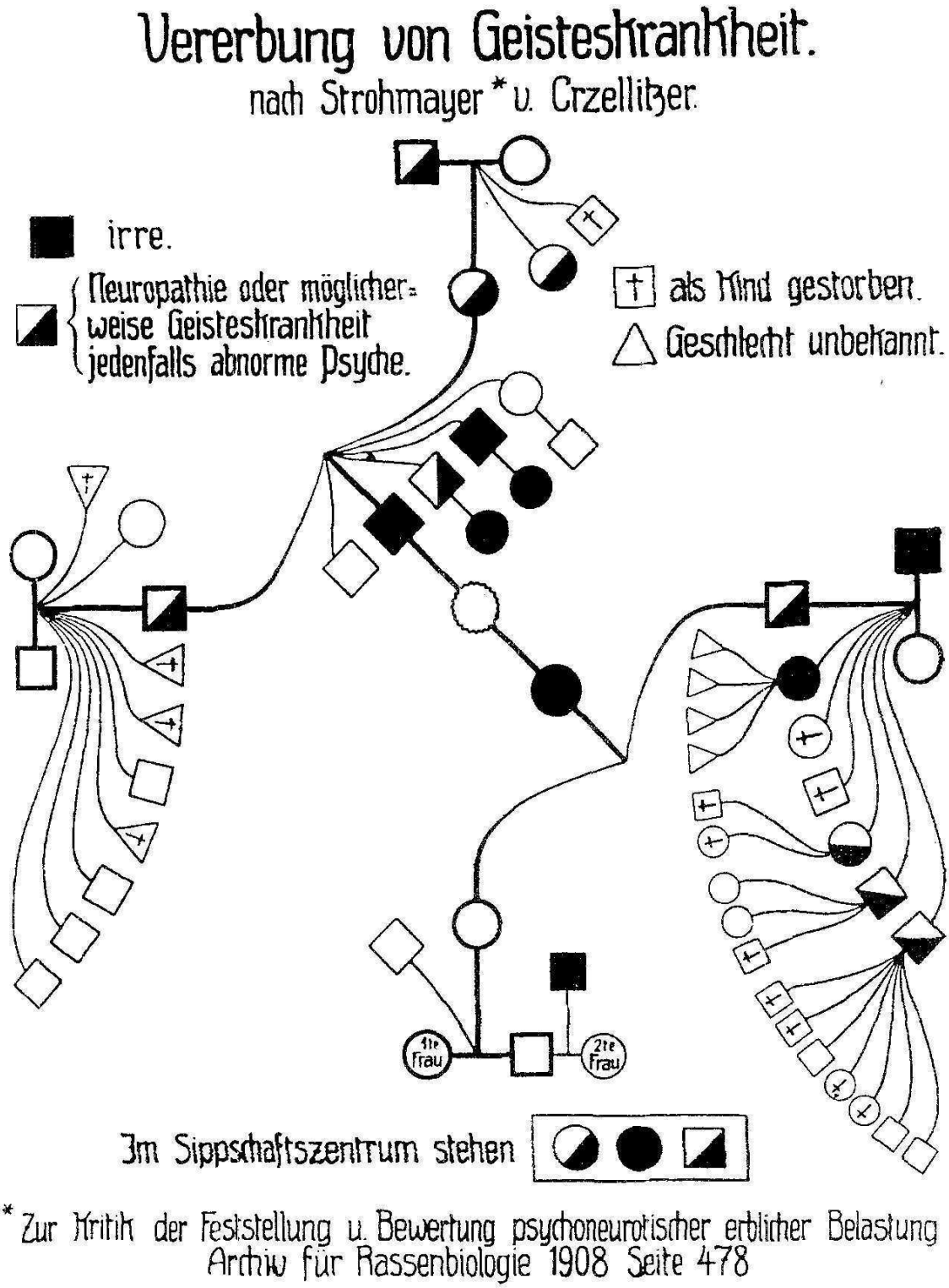


Figure 6: A pedigree in *Sippschaftstafel* style (Rüdin, 1910). From Resta, 1993

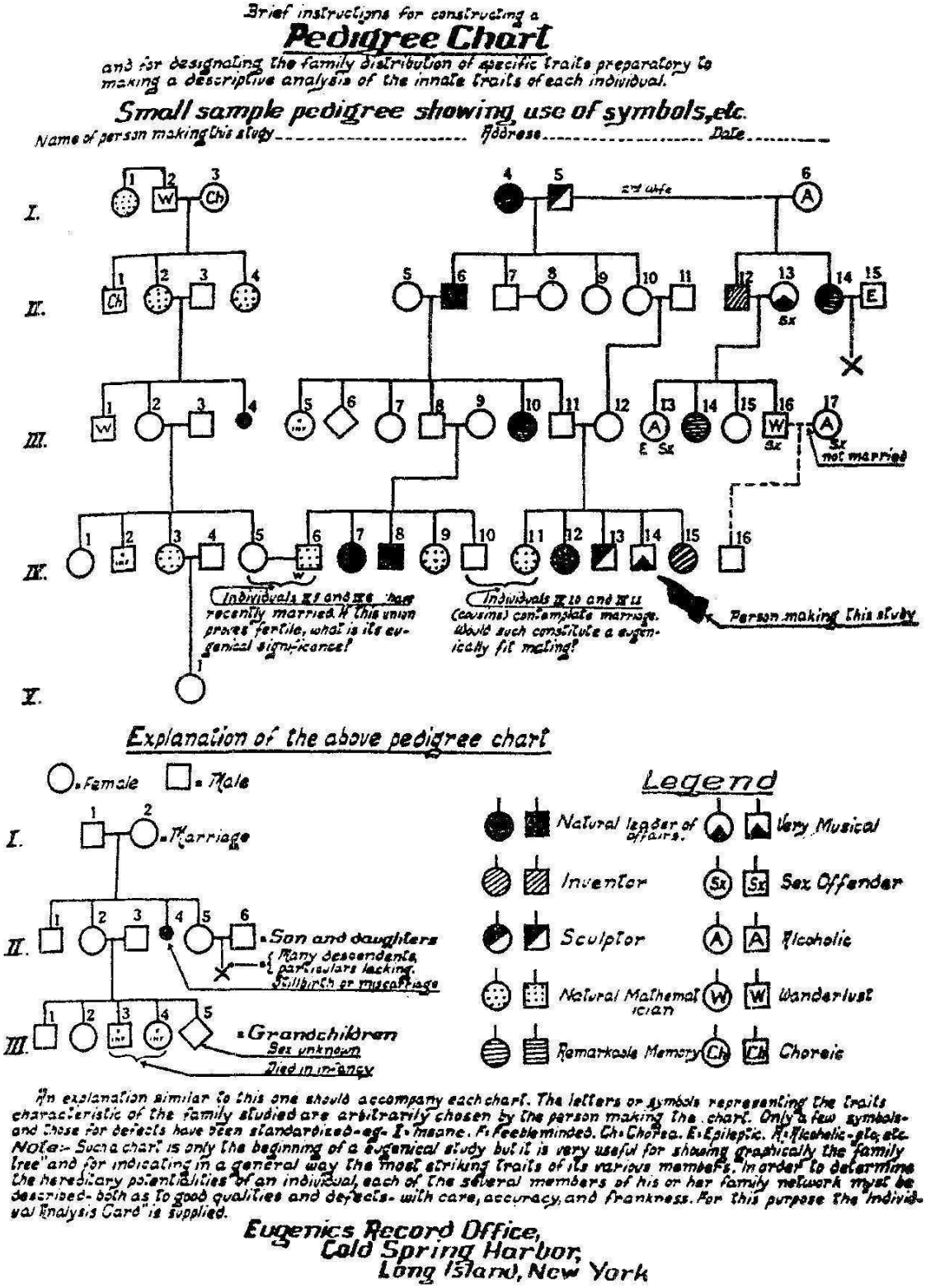


Figure 7: Sample pedigree from a Eugenics Records Office pamphlet, 1911. From Resta, 1993

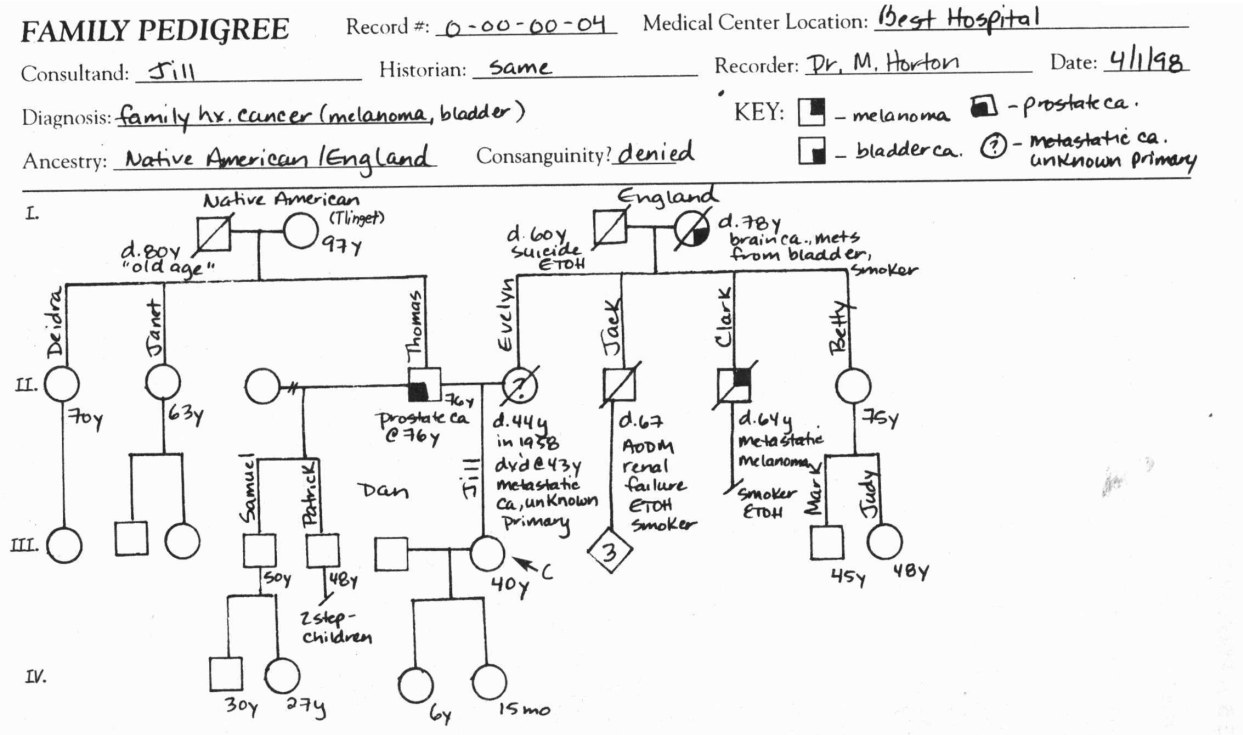


Figure 8: A modern hand-drawn pedigree. From Bennett, 1999

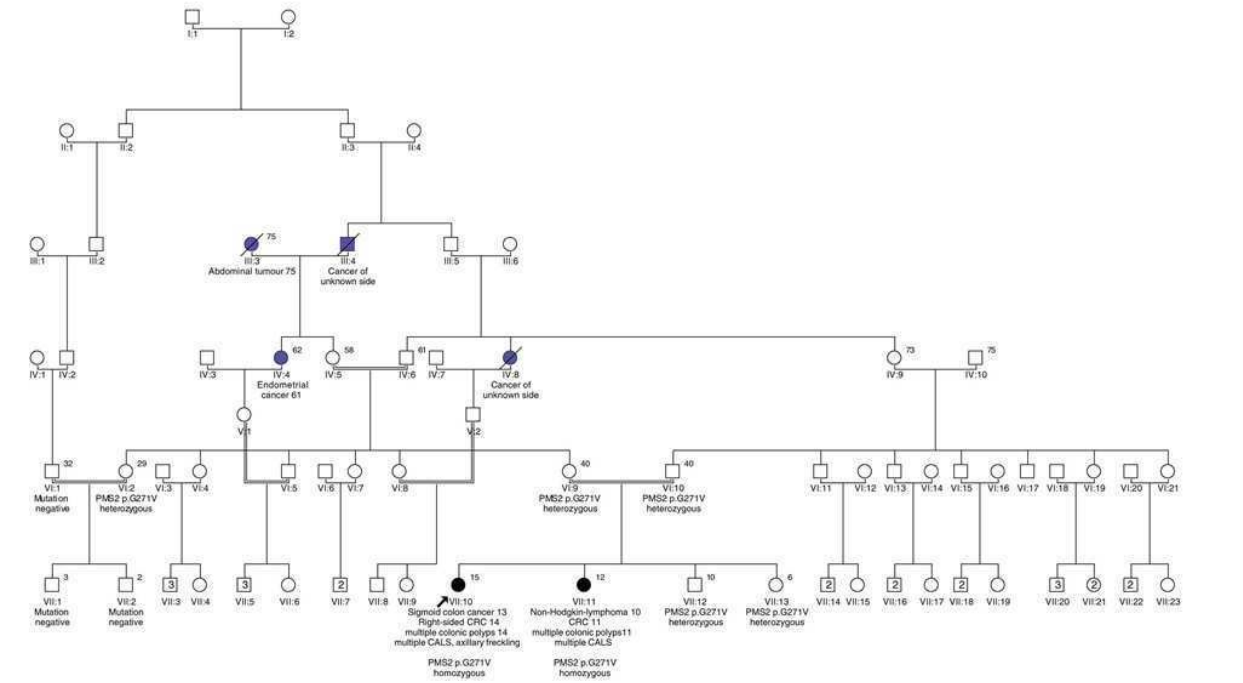


Figure 2 Pedigree of family 2. Tumours and age of onset are reported. Black symbols indicate CCS-related tumours (CRC, colorectal cancer). Blue symbols indicate other tumours. Numbers on right-top give the current age or age of death. The index patient is indicated by an arrow. In addition, signs of NF1 and mutational status are given.

Figure 9: A pedigree from Krüger et al., 2008

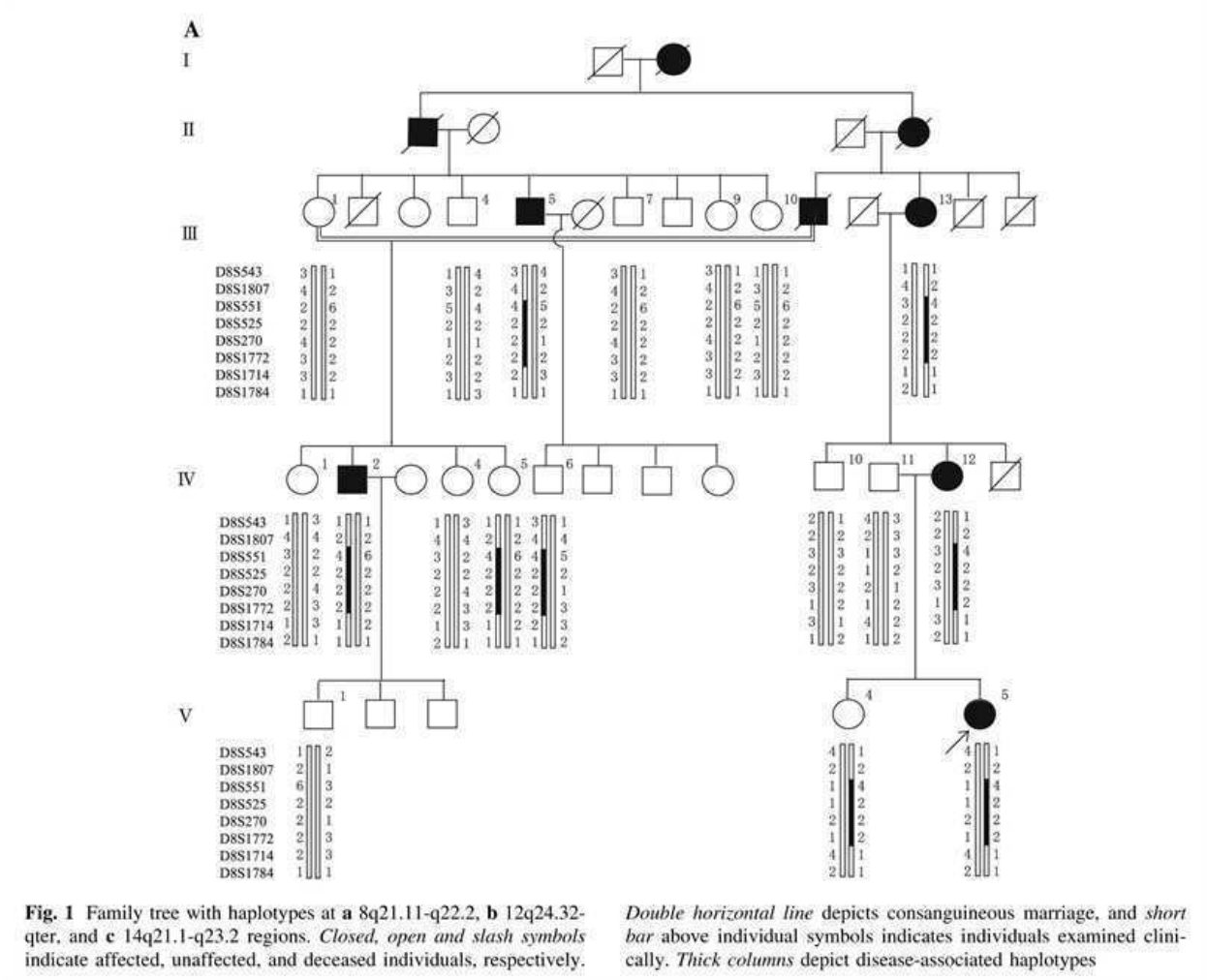


Fig. 1 Family tree with haplotypes at **a** 8q21.11-q22.2, **b** 12q24.32-qter, and **c** 14q21.1-q23.2 regions. *Closed, open and slash symbols* indicate affected, unaffected, and deceased individuals, respectively. *Double horizontal line* depicts consanguineous marriage, and *short bar* above individual symbols indicates individuals examined clinically. *Thick columns* depict disease-associated haplotypes

Figure 10: A pedigree from Nakashima et al., 2008

```
Id |Name                    |Sex   |DoB | DoD|Mother|Father|Proband|Comment
AV |Alexis Vidal            |male  |1777|1852|      |      |       |n\'egociant, propri\'etaire
AB |Ad\'ela\"\i{}de Bourbon|female|1775|1855|JT    |AnB   |       |
EV |\'Elise Vidal           |female|1814|1868|AB    |AV    |       |
PJ |Pierre Joannon          |male  |1806|1838|      |      |       |n\'egociant, rentier
PaJ|Paul Joannon            |male  |1834|1882|EV    |PJ    |       |avocat
...
```

Figure 11: An example of pedigree input

people could create simple typeset pedigrees without the trouble of installing the full TEX suite — and perhaps becoming interested enough in the results to install it anyway. This is an idea worth pursuing.

Karl Pearson noted in 1912, *A complete pedigree is often a work of great labour, and its finished form is frequently a work of art.* We hope our work helps to make pedigree drawing less laborious — while preserving the beauty of the result.

Acknowledgements

We appreciate the kind permission by Prof. Resta to use his illustrations for this paper.

We are grateful to Karl Berry for his tireless support and great suggestions.

One of the authors (LA) would like to acknowledge the travel support from the TEX Users Group, Russian Federation President's Board for Support

```
# Do we want to print a legend?
$printlegend = 1;

# Fields to include in the legend.
# Delete Name for privacy.
@fieldsforlegend = qw(Name DoB AgeAtDeath
                      Comment);

# Fields to put at the node.
# Delete Name for privacy.
our @fieldsforchart = qw();
```

Figure 12: A fragment of our configuration file

of Young Russian Scientists and Scientific Schools (grant No. MD-1195.2008.7), Russian Foundation for Basic Research (grant No. 09-04-08060-z) and Ministry of Education and Science of the Russian Federation (grant No. NK-30P/12).

References

Akhmadeeva, Leila. "Using a New Package for Drawing Pedigrees for Teaching Medical Genetics". *Eur. J. Hum. Gen.* **15**(Suppl. 1), 338, 2007.

Bennett, Robin L. *The Practical Guide to the Genetic Family History*. Wiley-Liss, Inc., New York; Chichester; Weinheim; Brisbane; Singapore; Toronto, 1999.

Coustillas, Françoise, and D. Girou. "[Example of Genealogy Tree with PSTricks]". `http://tug.org/PSTricks/main.cgi?file=Examples/Genealogy/genealogy`, 2004.

Krüger, Stefan, M. Kinzel, C. Walldorf, S. Gottschling, A. Bier, S. Tinschert, A. von Stackelberg, W. Henn, H. Görgens, S. Boue, K. Kölble, R. Büttner, and H. K. Schackert. "Homozygous PMS2 Germline Mutations in Two Families With Early-Onset Haematological Malignancy, Brain Tumours, HNPCC-Associated Tumours, and Signs of Neurofibromatosis Type 1". *Eur. J. Hum. Gen.* **16**(1), 62–72, 2008.

Nakashima, Mitsuko, M. Nakano, A. Hirano, T. Kishino, S. Kondoh, N. Miwa, N. Niikawa, and K.-i. Yoshiura. "Genome-wide linkage analysis and mutation analysis of hereditary congenital blepharoptosis in a Japanese family." *J. of Hum. Gen.* **53**(1), 34–41, 2008.

Resta, Robert G. "The Crane's Foot: The Rise of the Pedigree in The Human Genetics". *J. Genetic Couns.* **2**(4), 235–260, 1993.

Veytsman, Boris, and L. Akhmadeeva. "Drawing Medical Pedigree Trees with TEX and PSTricks". *PracTEX J.* (4), 2006. `http://www.tug.org/pracjourn/2006-4/veytsman`.

Veytsman, Boris, and L. Akhmadeeva. *Creating Medical Pedigrees with PSTricks and LATEX*, 2007a. `http://mirror.ctan.org/graphics/pstricks/contrib/pedigree/pst-pdgr`.

Veytsman, Boris, and L. Akhmadeeva. "Drawing Medical Pedigree Trees with TEX and PSTricks". *TUGboat* **28**(1), 100–109, 2007b. `http://www.tug.org/TUGboat/Articles/tb28-1/tb88veytsman-pedigree.pdf`.

Veytsman, Boris, and L. Akhmadeeva. *A Program For Automatic Pedigree Construction With pst-pdgr. User Manual and Algorithm Description*, 2007c. `http://mirror.ctan.org/graphics/pstricks/contrib/pedigree/pedigree-perl`.

Veytsman, Boris, and L. Akhmadeeva. "Medical Pedigrees with TEX and PSTricks: New Advances and Challenges". *TUGboat* **29**(3), 484, 2008. `http://www.tug.org/TUGboat/Articles/tb29-3/tb93abstracts.pdf`.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
`borisv (at) lk dot net`

◇ Leila Akhmadeeva
Bashkir State Medical University
3 Lenina St., Ufa, 450000, Russia
`la (at) ufaneuro dot org`

A first look at the TEX Gyre fonts

Jim Hefferon

Most LATEX documents look alike. The authors of such documents may respond to this observation with: the software is a tool that is adjunct to their work and so they just do what worked last time.

But for you, wouldn't it be nice to make your paper stand out, in a good way? One approach to that is to use different fonts. Recently a new set of fonts have been developed that are easy for TEX users and although they are still under development, they are nonetheless very handy today.

Besides that these fonts look good and have desirable technical properties, they are great because you will not need to do tricky incantations or to install additional materials. If you have a recent TEX distribution then you only need to add one line to your source file.

1 Background

In 1985, Steve Jobs persuaded Adobe to adapt their page description language PostScript for the first LaserWriter. Soon PostScript was so popular that authors could distribute a document in this format, trusting that recipients had a suitable printer or could view it onscreen. This was great for TEX users because there was no more need to compile the document for a particular printer model.

Adobe specified a set of ten font families that any PostScript printer must have (a family of fonts may contain boldface, italic, etc.). So these ten have become widely known and used.

Besides these ten, there are today thousands of fonts available, either commercially or for free. (In the past, fonts came in two formats, Type 1 and TrueType, but there is now a standard, OpenType, that should soon be the single format for all fonts.)

2 Gyre

So why do TEX users need new fonts when many thousands are available?

Not many available fonts include characters for all European languages (TEX can add accents and other marks but the mechanism is annoying and interferes with hyphenation) and not many are high quality, including the work needed to make the characters display correctly over a wide range of sizes. Of the fonts satisfying those criteria, very few are suitable for use with mathematics. And of these, very few indeed are Free.

Hans Hagen, well-known in the TEX community as the inventor of ConTEXt, initiated the Gyre project to provide such a set of fonts. The execution team comes from the Polish TEX users group GUST and includes Bogusław Jackowski, Janusz M. Nowacki, and Marcin Woliński. The team's designs trace their lineage to the Adobe standard ten (the descent runs through clone fonts made freely available by URW++). Thus, these are familiar and tested designs that fit the needs of many users.

Tout le monde dans la province de Candahar connaît l'aventure du jeune Rustan. Il était fils unique d'un mirza du pays; c'est comme qui dirait marquis parmi nous, ou baron chez les Allemands. Le mirza, son père, avait un bien honnête. On devait marier le jeune Rustan à une demoiselle, ou mirzasse de sa sorte. Les deux familles le désiraient passionnément. Il devait faire la consolation de ses parents, rendre sa femme heureuse, et l'être avec elle.

Figure 1: Gyre Chorus

To transform the existing designs into fonts that are optimal for TEX, the team went through a complex sequence of steps; see the papers on the Gyre web page [2]. This included adding characters with accents and other diacritic marks to cover all European languages. This also includs adding the mathematical symbols that TEX users expect.

The team is now making the fonts as usable with mathematics as are Knuth's original fonts; for instance, they are adjusting the spacing between characters and subscripts. But you can nonetheless start using these fonts today in documents that have mathematics, by using packages intended to work with the Adobe fonts.

3 Choices

Two of Adobe's standard ten, Symbol and Zapf Dingbats, are not text fonts and so Gyre does not include extensions of them. This table gives the remaining eight with their Gyre equivalents.

Adobe name	Gyre name
Zapf Chancery	Chorus
Courier	Cursor
Helvetica	Heros
Avant Garde	Adventor
Bookman	Bonum
New Century Schoolbook	Schola
Times	Termes
Palatino	Pagella

A sample of the first in that table, Chorus, is shown in Figure 1, which is produced by this source (from [5]).

```
\documentclass{article}
\usepackage{tgchorus}

\usepackage{ucs} % Unicode support
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
```

x	sin	tan	cot	cos
.00	.00000	.00000	∞	1.00000
.01	.01000	.01000	99.997	0.99995
.02	.02000	.02000	49.993	.99980
.03	.03000	.03001	33.323	.99955

Figure 2: Gyre Cursor used for a table

```
\pagestyle{empty}
\begin{document}\thispagestyle{empty}
Tout le monde dans la province de Candahar
connaît l'aventure du jeune Rustan.
...
rendre sa femme heureuse, et l'être avec elle.
\end{document}
```

Note the lack of backslashes in the text, that is, the French author can write in French. Note also that the one line `\usepackage{tgchorus}` is all that we need to use Gyre Chorus in this non-mathematical document. While Chorus is a text font, the sample shows that it is specialized for things like wedding invitations, and is not suitable for a typical TEX document.

The table's second entry, Cursor, is also specialized; see Figure 2 (from [1]); the column headers are from a different font. This font is monospaced — the digits have equal widths — so it is useful for showing a table of numbers or a computer code listing. However, a TEX author wouldn't use it for a main body font.

Our focus is on trying Gyre in documents with mathematics so we will not further discuss these two.

4 Math font options

There are some packages of TEX fonts that were designed to provide mathematics capabilities to supplement one or more of the Adobe text fonts. We can use these with the Gyre fonts.

Young Ryu's `pxfonts` matches Adobe Palatino and Gyre Pagella, and provides all of the symbols of the Computer Modern and AMS fonts. The `txfonts` package does the same for Adobe Times and Gyre Termes. (The LATEX packages `qpxmath` and `qtxmath` use the math from these two but leave the text font unchanged, so you can load the text and math font in either order.)

Diego Puga's `mathpazo` is a set of PostScript fonts for typesetting mathematics in combination with Adobe Palatino or Gyre Pagella. The fonts include the uppercase Greek alphabet in upright and slanted shapes in regular and bold weights, lowercase Greek alphabet in slanted shape in regular and bold weights, several mathematical glyphs in regular and bold weights, and some others. The set also includes true small-caps fonts.

The `mathptmx` package changes the main font to Times, and for math uses Times Italic, Computer Modern, Ralph Smith's Formal Script, and Adobe Symbol.

Christophe Caignaert's package `kpfonts` shares its heritage with Gyre Pagella, and among other things includes all the symbols from the AMS fonts.

5 Sans serif fonts

In the table relating Adobe's standard fonts to the Gyre fonts, the next two, Heros and Adventor, are sans serif fonts. (A serif is a stroke at the ends of some characters. In these two examples: A and A, the first has small horizontal strokes at the bottom. Those are serifs. The second one is sans serif.)

Sans serif fonts are typically used more for presentations or short work than for long or intricate technical material. But TEX users certainly often use sans serif fonts, and there are several combinations that are quite suitable. See Figure 3.

6 Serif fonts

Serif fonts are the ones typically used by TEX authors. In Gyre these are Bonum, Schola, Termes, and Pagella.

Starting with Figure 5 (adapted from [4]), there are a number of combinations that you may find suitable for your document. (In addition to the math font packages described above, one of the figures uses the Euler font.)

7 Closing

Work is continuing at a steady pace on the Gyre fonts. When they appear, their math support should be excellent. But you don't have to wait to have a first look. Of the combinations shown above, Gyre Termes with `qtxmath`, Gyre Pagella with `qpxmath`, and Gyre Pagella with `eulervm` seem particularly worth a try. (For even more options, see [3].)

References

[1] *CRC Standard Mathematics Tables*. W Beyer, ed, 24ed, CRC Press, 1976. p 209.

[2] *The TEX Gyre (TG) Collection of Fonts*. GUST. `http://www.gust.org.pl/projects/e-foundry/tex-gyre/`

[3] *A Survey of Free Math Fonts*. S Hartke. `http://mirror.ctan.org/info/Free_Math_Font_Survey/survey.html`

[4] *Infinite Sequences and Series*. K Knopp, Dover, 1956. p 78.

[5] *Le Blanc et le Noir*. Voltaire, 1764, p 1. `http://www.gutenberg.org/dirs/etext03/betn810.txt`

◇ Jim Hefferon
Saint Michael's College
Colchester, Vermont USA
`ftpmaint (a) tug dot ctan dot org`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s+t$ and $s-t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s + t$ and $s - t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 3: Gyre Heros with Computer Modern math `\usepackage{tgheros}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s + t$ and $s - t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s+t$ and $s-t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 4: Gyre Adventor with `txfonts` math `\usepackage{kmath,tgadventor}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s + t$ and $s - t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s + t$ and $s - t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 5: Gyre Bonum with `txfonts` for math `\usepackage{kmath,tgbonum}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s+t$ and $s-t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s+t$ and $s-t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 6: Gyre Schola with Fourier math `\usepackage{fouriernc,tgschola}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s+t$ and $s-t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s+t$ and $s-t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 7: Gyre Termes with `qtx` math `\usepackage{qtxmath,tgtermes}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s+t$ and $s-t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s+t$ and $s-t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 8: Gyre Termes with `ptx` math `\usepackage{ptxmath,tgtermes}`

If $\sum \mathfrak{a}$ and $\sum \mathfrak{b}$ are convergent series with the respective values $\mathfrak{s}$ and $\mathfrak{t}$, then the series

$$\sum(\mathfrak{a}_\nu + \mathfrak{b}_\nu) \quad \text{and} \quad \sum(\mathfrak{a}_\nu - \mathfrak{b}_\nu)$$

are also convergent and have the respective values $\mathfrak{s}+\mathfrak{t}$ and $\mathfrak{s}-\mathfrak{t}$. Likewise, the series

$$\mathfrak{a}_0 + \mathfrak{b}_0 + \mathfrak{a}_1 + \mathfrak{b}_1 + \ldots \quad \text{and} \quad \mathfrak{a}_0 - \mathfrak{b}_0 + \mathfrak{a}_1 - \mathfrak{b}_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $\mathfrak{s} + \mathfrak{t}$ and $\mathfrak{s} - \mathfrak{t}$. Finally, if $\mathfrak{c}$ is an arbitrary number then $\sum \mathfrak{c}\mathfrak{a}_\nu$ is also convergent and has the value $\mathfrak{c}\mathfrak{s}$.

Figure 9: Gyre Pagella with Euler math `\usepackage{eulervm,tgpagella}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s+t$ and $s-t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s + t$ and $s - t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 10: Gyre Pagella with qpx math `\usepackage{qpxmath,tgpagella}`

If $\sum a$ and $\sum b$ are convergent series with the respective values s and t, then the series

$$\sum(a_v + b_v) \quad \text{and} \quad \sum(a_v - b_v)$$

are also convergent and have the respective values $s + t$ and $s - t$. Likewise, the series

$$a_0 + b_0 + a_1 + b_1 + \ldots \quad \text{and} \quad a_0 - b_0 + a_1 - b_1 + \ldots$$

resulting from the removal of the parentheses, are convergent with respective values $s + t$ and $s - t$. Finally, if c is an arbitrary number then $\sum ca_v$ is also convergent and has the value cs.

Figure 11: Gyre Pagella with pazo math `\usepackage{mathpazo,tgpagella}`

Plain TEX and OpenType

Hans Hagen

Abstract

This article demonstrates how you can use OpenType fonts in plain LuaTEX, using a few generic modules that ship with the ConTEXt distribution.

1 Running

For testing basic LuaTEX functionality it makes sense to have a minimal system, and traditionally plain TEX has been the most natural candidate. It is for this reason that it had been on the agenda for a while to provide basic OpenType font support for plain TEX as well. Although the MkIV node mode subsystem is not yet perfect, the time was right to start experimenting with a subset of the MkIV code.

Using plain roughly comes down to the following. First you need to generate a format:

```
luatex --ini --fmt=luatex.fmt luatex-plain.tex
```

This format has to be moved to a place where it can be found by the `kpse` library. Since this can differ per distribution there is no clear recipe for it, but for TEX Live some path ending in `web2c/luatex` is probably the right spot. After that you can run

```
luatex luatex-test.tex
```

This file lives under `generic/context`. When it is run it is quite likely that you will get an error message because the font name database cannot be found. You can generate one with the following command (which assumes that you have ConTEXt installed):

```
mtxrun --usekpse --script fonts --names
```

The resulting file `luatex-fonts-names.lua` has to be placed somewhere in your TEX tree so that it can be found anytime. Beware: the `--usekpse` flag is only used outside ConTEXt and provides very limited functionality, just enough for this task. Again this is a distribution specific issue so we will not dwell upon it here.

The way fonts are defined is modelled after XETEX, as it makes no sense to support the somewhat more fancy ConTEXt way of doing things. Keep in mind that although ConTEXt MkIV does support the XETEX syntax too, the preferred way there is to use a more symbolic feature definition approach.

As this is an experimental setup, it might not always work out as expected. Around LuaTEX version 0.50 we expect the code to be more or less okay.

2 Implementation

The `luatex-fonts.lua` file is the first in a series of basic functionality enhancements for LuaTEX derived from the ConTEXt MkIV code base. Please don't pollute the `luatex-*` namespace with code not coming from the ConTEXt development team as we may add more files.

This file implements a basic font system for a bare LuaTEX system. By default LuaTEX only knows about the classic TFM fonts but it can read other font formats and pass them to Lua. With some glue code one can then construct a suitable TFM representation that LuaTEX can work with. For more advanced font support a bit more code is needed that needs to be hooked into the callback mechanism.

This file is currently rather simple: it just loads the Lua file with the same name. An example of a `luatex.tex` file that is just the plain TEX format:

```
\catcode`\{=1 % { is begin-group character
\catcode`\}=2 % } is end-group character
\input plain
\everyjob\expandafter{\the\everyjob
    \input luatex-fonts\relax}
\dump
```

We could load the Lua file in `\everyjob` but maybe some day we will need more here.

When defining a font, in addition to the XETEX way, you can use two prefixes. A `file:` prefix forces a file search, while a `name:` prefix will result in consulting the names database. The font definitions shown in figure 1 are all valid.

You can load maths fonts but as Plain TEX is set up for Computer Modern (and as we don't adapt Plain TEX) loading Cambria does not give you support for its math features automatically.

If you want access by name you need to generate a font database, using:

```
mtxrun --script font --names
```

and put the resulting file in a spot where LuaTEX can find it.

3 Remarks

The code loaded in `luatex-fonts.lua` does not come out of thin air, but is mostly shared with ConTEXt; however, in that macro package we go beyond what is provided in the plain variant. When using this code you need to keep a few things in mind:

- This subsystem will be extended, improved etc. at about the same pace as ConTEXt MkIV. However, because ConTEXt provides a rather high level of integration not all features will be supported in the same quality. Use ConTEXt if you want more goodies.
- There is no official API yet, which means that using functions implemented here is at your own risk, in the sense that names and namespaces

```
\font\testa=file:lmroman10-regular at 12pt
\font\testb=file:lmroman12-regular:+liga; at 24pt
\font\testc=file:lmroman12-regular:mode=node;+liga; at 24pt
\font\testd=name:lmroman10bold at 12pt
\font\testh=cmr10
\font\testi=ptmr8t
\font\teste=[lmroman12-regular]:+liga at 30pt
\font\testf=[lmroman12-regular] at 40pt
\font\testj=adobesongstd-light % cid font
\font\testk=cambria(math) {\mathtest 123}
\font\testl=file:IranNastaliq.ttf:mode=node;script=arab;\
    language=dflt;+calt;+ccmp;+init;+isol;+medi;+fina;+liga;\
    +rlig;+kern;+mark;+mkmk at 14pt
```

Figure 1: Font definition examples in LuaTEX

might change. There will be a minimal API defined once LuaTEX version 1.0 is out. Instead of patching the files it's better to overload functions if needed.

- The modules are not stripped too much, which makes it possible to benefit from improvements in the code that take place in the perspective of ConTEXt development. They might be split a bit more in due time so the baseline might become smaller.
- The code is maintained and tested by the ConTEXt development team. As such it might be better suited for this macro package and integration in other systems might demand some additional wrapping. The plain version discussed here is the benchmark and should be treated as a kind of black box.
- Problems can be reported to the team but as we use ConTEXt MkIV as our baseline, you'd better check if the problem is a general ConTEXt problem too.
- The more high level support for features that is provided in ConTEXt is not part of the code loaded here as it makes no sense elsewhere. Some experimental features are not part of this code either but some might show up later.
- Math font support will be added but only in its basic form once the Latin Modern and TEX Gyre math fonts are available. Currently traditional and OpenType math fonts can be loaded.
- At this moment the more nifty speedups are not enabled because they work in tandem with the alternative file handling that ConTEXt uses. Maybe around LuaTEX 1.0 we will bring some speedup into this code too (if it pays off at all).
- The code defines a few global tables. If this code is used in a larger perspective then you can best make sure that no conflicts occur. The ConTEXt package expects users to work in their own namespace (`userdata`, `thirddata`, `moduledata` or `document`). We give ourselves the freedom to use any table at the global level but will not use tables that are named after macro packages. Later, ConTEXt might operate in a more controlled namespace but it has a low priority.
- There is some tracing code present but this is not enabled and not supported as it integrates quite tightly into ConTEXt. In case of problems you can use ConTEXt for tracking down problems.
- Patching the original code in distributions is dangerous as it might fix your problem but introduce new ones for ConTEXt. So, best keep the original code as it is and overload functions and callbacks when needed. This is trivial in Lua.
- Attributes are (automatically) taken from the range 127–255 so you'd best not use these yourself. Don't count on an attribute number staying the same and don't mess with these attributes.

If this all sounds a bit strict, keep in mind that it makes no sense for us to maintain multiple code bases and we happen to use ConTEXt.

◇ Hans Hagen
Pragma ADE
The Netherlands
http://pragma-ade.org

Integrating Unicode and OpenType math in ConTEXt

Aditya Mahajan

Abstract

In this article, I briefly explain the approach taken by ConTEXt to integrate Unicode math with OpenType fonts.

1 A bit of history

Since around 2000, ConTEXt has supported Unicode math input. Under the `utf-8` input regime (obtained with `\enableregime[utf-8]`), you could type `$αβγ$` to get the Greek letters $\alpha\beta\gamma$. This was achieved by lot of macro jugglery; similar to the kind of macro jugglery needed for accents to work without having to type in the arcane plain TEX accent macros. Basically, the input regime ensured that inside math mode `α` got mapped to `\alpha`, and the rest was taken care of by the usual font mappings that mapped `\alpha` to the correct glyph in the correct font. However, these mappings were not clean. If you defined a macro, say `\MACRO`, that took one argument, then `\MACRO α` would not work; you had to type `\MACRO{α}`. This small detail of grouping UTF-8 input was a constant reminder that things were not so clean underneath.

LuaTEX was designed to handle input encoding cleanly. The engine only understands UTF-8 encoding and provides enough hooks to implement other encodings. ConTEXt MkIV assumes that the input is either UTF-8 or UTF-16. The input can then be directly mapped to the correct glyph locations in TrueType and OpenType fonts. However, handling of math input was much trickier, mainly because of the effort needed to support OpenType math. So, for some time, the handling of math fonts did not change: in math mode `α` was mapped to `\alpha` and traditional TEX font mapping ensured that `\alpha` was mapped to the correct glyph in the correct font.

Around the beginning of this year, ConTEXt MkIV completely moved to Unicode math. Thus, `0x1D6FC` (math italic small alpha) was mapped to the same position in an OpenType math font. If you type this Unicode character in math mode, the output is α. For convenience, typing `0x03B1` (Greek small letter alpha) in math mode gets mapped to `0x1D6FC`; as does the macro `\alpha`. All this is transparent to the user, except when he accidentally types `$\MACRO α$` and is pleasantly surprised not to get a nasty error message.

There are two steps involved in the integration of Unicode input with OpenType math fonts: (i) map the input characters or macros to the correct Unicode character; (ii) map the Unicode character to the correct OpenType glyph at the correct size and with the correct kerning. The bulk of these mappings are done by five files: `char-def.lua`, `math-ini.lua`, `math-map.lua`, `math-noa.lua` and `math-vfu.lua`. Below I explain briefly what these files do. Please take everything in this article with a pinch of salt. I do not understand how OpenType math fonts work, so specific details may be wrong. The main idea of this article is to convey the gist of how things are done in ConTEXt, and where to search if you want to know specific details.

2 char-def.lua

Mapping UTF-8 input chapters to Unicode characters is straightforward — the byte sequence of the input character is the same as the Unicode slot. Thus, input byte sequence `0x1D6FC` is the Unicode character `0x1D6FC`. However, mapping macros to Unicode byte sequences is different. We need to explicitly tell ConTEXt that `\alpha` corresponds to the Unicode character `0x1D6FC`. Furthermore, just getting the correct glyph is not sufficient for typesetting mathematics. We also need to know the math class of the glyph, so that TEX can correctly position the characters.

All this information is stored as a Lua table in `char-def.lua`. This is a huge file, initially generated from Unicode tables and later updated manually from data present in ConTEXt files and elsewhere (it is still not complete). For example, the entry for `0x1D6FC` is:

```
[0x1D6FC]={
 category="ll",
 description="MATHEMATICAL ITALIC SMALL ALPHA",
 direction="l",
 linebreak="al",
 mathclass="default",
 mathname="alpha",
 specials={ "font", 0x03B1 },
 unicodeslot=0x1D6FC,
},
```

Amongst other things, this tells ConTEXt that the macro `\alpha` (indicated by `mathname="alpha"`) corresponds to this Unicode slot. It also tells that the math class of this character is `default`. Similar details are provided for a large fraction of the Unicode characters.

Some Unicode characters correspond to more than one macro (with different math classes). One such example is `0x007C`, which corresponds to `\arrowvert`, `\vert`, `\lvert`, `\rvert`, and `\mid`, each belonging to a different math class. This Uni-

code character is represented as:

```
{
 adobename="bar",
 category="sm",
 cjkwd="na",
 contextname="textbar",
 description="VERTICAL LINE",
 direction="on",
 linebreak="ba",
 mathspec={
  { class="nothing",   name="arrowvert" },
  { class="delimiter", name="vert"      },
  { class="open",      name="lvert"     },
  { class="close",     name="rvert"     },
  { class="relation",  name="mid"       },
 },
 unicodeslot=0x007C,
},
```

The different macros and their corresponding math classes are encoded as part of a `mathspec` key. When the character `|` (`0x007C`) is typed the math class is set to the class of the first element of the `mathspec` table (`nothing` in this case).

3 math-ini.lua

All the information in the Lua table in `char-def.lua` by itself is useless. We need to tell ConTEXt how to use it. For the math mappings, this is done in `math-ini.lua`.

This file begins by defining names for the different math classes:

```
local classes = {
    ord       =  0,  -- mathordcomm
    op        =  1,  -- mathopcomm
    bin       =  2,  -- mathbincomm
    rel       =  3,  -- mathrelcomm
    open      =  4,  -- mathopencomm
    close     =  5,  -- mathclosecomm
    punct     =  6,  -- mathpunctcomm
    alpha     =  7,  -- mathalphacomm
    accent    =  8,  -- class 0
    radical   =  9,
    xaccent   = 10,  -- class 3
    topaccent = 11,  -- class 0
    botaccent = 12,  -- class 0
    under     = 13,
    over      = 14,
    delimiter = 15,
    inner     =  0,  -- mathinnercomm
    nothing   =  0,  -- mathnothingcomm
    choice    =  0,  -- mathchoicecomm
    box       =  0,  -- mathboxcomm
    limop     =  1,  -- mathlimopcomm
    nolop     =  1,  -- mathnolopcomm
}
```

For each math class, this file has functions to return the LuaTEX code to define the corresponding math characters. A few examples of such functions:

```
local function delcode(target,family,slot)
    return format('\\Udelcode%s="%X "%X ',
                  target,family,slot)
end
local function mathchar(class,family,slot)
    return format('\\Umathchar "%X "%X "%X ',
                   class,family,slot)
end
local function mathaccent(class,family,slot)
    return format('\\Umathaccent "%X "%X "%X ',
                  0,family,slot)
end
```

Similar functions are there for defining math symbols, for example:

```
function setmathsymbol(name,class,family,slot)
  if class == classes.accent then
    texsprint(
        format("\\unexpanded\\xdef\\%s{%s}",
        name,
        mathaccent(class,family,slot)))
  elseif class == classes.topaccent then
    texsprint(
        format("\\unexpanded\\xdef\\%s{%s}",
        name,
        mathtopaccent(class,family,slot)))
  elseif class == classes.botaccent then
    texsprint(
        format("\\unexpanded\\xdef\\%s{%s}",
        name,
        mathbotaccent(class,family,slot)))
  ...
end
```

`math-ini.lua` then defines a Lua function named `mathematics.define` that goes through all the elements in the table in `char-def.lua` and maps them to the correct LuaTEX command.

These mappings are all that is needed to work with OpenType math fonts like Cambria and Asana Math. ConTEXt has predefined typescripts for Cambria; so, to use Cambria you can just type

```
\usetypescript[cambria]
\setupbodyfont[cambria]
```

There are no predefined typescripts for Asana Math, but defining one on our own is not too hard. First, we need to define a math typescript:

```
\starttypescript [math] [asana] [name]
  \definefontsynonym
    [MathRoman]
    [name:Asana-Math]
    [features=math\mathsizesuffix]
\stoptypescript
```

The `features=math\mathsizesuffix` option activates the OpenType math features. The rest of the typescript can be defined in the usual manner.

```
\starttypescript [asana]
  \definetypeface [asana] [rm] [serif]
      [palatino] [default]
  \definetypeface [asana] [ss] [sans]
      [modern]   [default] [rscale=1.075]
  \definetypeface [asana] [tt] [mono]
      [modern]   [default] [rscale=1.075]
  \definetypeface [asana] [mm] [math]
      [asana]    [default]
  \quittypescriptscanning
\stoptypescript
```

To use this typescript, we need to type

```
\usetypescript[asana]
\setupbodyfont[asana]
```

4 math-map.lua and math-noa.lua

So far, we have mapped Unicode input and macros to OpenType math fonts. However, when using TEX one expects traditional TEX input to work. Thus, `$a$` should typeset math italic *a*. No one is likely to type `0x1D44E`, even if Unicode suggests that. The same is true for bold fonts. One expects `${\bfa}$` to give bold **a**, even if Unicode suggests that we should have typed `0x1D41A`.

To accommodate this, in math mode ConTEXt maps upper and lower case letters `A-Z`, `a-z`, digits `0-9`, and upper and lower case Greek letters `α-ω`, `A-Ω` to the corresponding ranges in Unicode math, depending on the current font style. These mappings are defined in `math-map.lua` file.

The mappings are defined using a Lua table, which looks like this.

```
mathematics.alphabets = {
    regular = {
              tf = { ... },
              it = { ... },
              bi = { ... },
              bf = { ... },
    },
    sansserif = {
              tf = { ... },
              it = { ... },
              bi = { ... },
              bf = { ... },
    },
    monospaced = {
              tf = { ... },
    },
    blackboard = {
              tf = { ... },
    },
    fraktur = {
              tf = { ... },
              bf = { ... },
    },
    script  = {
              tf = { ... },
              bf = { ... },
    }
}
```

Each of these subtables maps input letters to their corresponding Unicode characters. These subtables look as follows.

```
regular = {
    ...
    it = {
      ucletters = 0x1D434,
      lcletters = { -- H
        [0x00061]=0x1D44E, [0x00062]=0x1D44F,
        [0x00063]=0x1D450, [0x00064]=0x1D451,
        [0x00065]=0x1D452, [0x00066]=0x1D453,
        [0x00067]=0x1D454, [0x00068]=0x0210E,
        ... },
     symbols = {
        [0x0391]=0x1D6E2, [0x0392]=0x1D6E3,
        [0x0393]=0x1D6E4, [0x0394]=0x1D6E5,
        [0x0395]=0x1D6E6, [0x0396]=0x1D6E7,
        ... },
    },
},
```

The line `ucletters = 0x1D434` tells ConTEXt to map upper case letters to Unicode characters starting from `0x1D434`. The line `lcletters = {...}` tells ConTEXt to map `0x00061` to `0x1D44E`, `0x00062` to `0x1D44F`, as so on. For lower case letters, simply specifying `lcletters = 0x1D44E` would not work because Unicode mathematical italic small letters are not in contiguous slots. For example, the slot `0x1D455` (which corresponds to lower case h) is empty; lower case h should map to slot `0x0210E`.

Other subtables are filled in a similar manner.

`math-map.lua` also defines Lua functions that use these tables to remap characters on the fly. The actual transformation takes place in `math-noa.lua` which goes through the math noad list and carries out the actual transformations according to the mappings in `math-map.lua`.

5 math-vfu.lua

Using the above infrastructure, it is easy to use OpenType math fonts in ConTEXt. Unfortunately, at present there are only two Unicode math fonts — Cambria and Asana Math. OpenType math version of TEX Gyre math fonts are planned, but until they are developed, we need a way to use traditional TEX fonts in ConTEXt MkIV. ConTEXt creates virtual OpenType math fonts to use traditional TEX fonts. The mappings for creating the virtual font are in `math-vfu.lua`. Once a virtual OpenType math font is created, the above infrastructure can be used to access the font.

First, `math-vfu.lua` defines many encoding

vectors that map Unicode characters to glyph locations of the font. One such encoding vector is

```
fonts.enc.math["tex-mi"] = {
    [0x1D6E4] = 0x00, -- Gamma
    [0x1D6E5] = 0x01, -- Delta
    [0x1D6E9] = 0x02, -- Theta
    [0x1D6F3] = 0x02, -- varTheta (not in TeX)
    [0x1D6EC] = 0x03, -- Lambda
    [0x1D6EF] = 0x04, -- Xi
    [0x1D6F1] = 0x05, -- Pi
    [0x1D6F4] = 0x06, -- Sigma
    ...
}
```

This tells that Unicode character `0x1D6E4` should be mapped to the font glyph `0x00` and so on. A virtual font that associates such encoding vectors with traditional TEX fonts is created using

```
mathematics.make_font ( "lmroman10-math", {
  { name="lmroman10-regular.otf",
    features="virtualmath", main=true },
  { name="lmmi10.tfm", vector="tex-mi",
    skewchar=0x7F },
  { name="lmsy10.tfm", vector="tex-sy",
    skewchar=0x30, parameters=true },
  { name="lmex10.tfm", vector="tex-ex",
    extension=true },
  { name="msam10.tfm", vector="tex-ma" },
  { name="msbm10.tfm", vector="tex-mb" },
  { name="lmroman10-bold.otf", "tex-bf" } ,
  { name="lmmib10.tfm", vector="tex-bi",
    skewchar=0x7F },
  { name="lmsans10-regular.otf",
    vector="tex-ss", optional=true },
  { name="lmmono10-regular.otf",
    vector="tex-tt", optional=true },
  { name="eufm10.tfm", vector="tex-fraktur",
    optional=true },
  { name="eufb10.tfm",
    vector="tex-fraktur-bold", optional=true },
} )
```

This creates a virtual font `lmroman10-math` which takes bits and pieces from various fonts. This virtual font can be used as follows.

```
\starttypescript [math] [modern]
  ...
  \definefontsynonym
    [LMMathRoman10-Regular]
    [LMMath10-Regular@lmroman10-math]
  ...
\stoptypescript
```

The `@lmroman-math` in the name uses the above virtual font. The `LMMathRoman10-Regular` font can be used to complete the math typescript in the usual manner.

ConTEXt provides virtual OpenType math fonts for Latin Modern, Times (`txfonts` and MathTime), Palatino (`pxfonts`), Iwona, Lucida, and MathDesign (Charter, Garamond, and Utopia) fonts.

6 Conclusion

ConTEXt now supports OpenType math fonts. In fact, even support for traditional TEX fonts now involves creating a virtual OpenType math font. Thus, as far as ConTEXt MkIV is concerned, OpenType math fonts are the future. However, the current implementation is still evolving, so some of the implementation details described in this paper will likely change with time.

◇ Aditya Mahajan
adityam (at) umich dot edu

LuaTEX: A user's perspective

Aditya Mahajan

Abstract

In this article, I explain how to use Lua to write macros in LuaTEX. I give some examples of macros that are complicated in pdfTEX, but can be defined easily using Lua in LuaTEX. These examples include macros that do arithmetic on their arguments, use loops, and parse their arguments.

1 Introduction

TEX is getting a new engine — LuaTEX. As its name suggests, LuaTEX adds Lua, a programming language, to TEX, the typesetter. I cannot overemphasize the significance of being able to program TEX in a high-level programming language. For example, consider a TEX macro that divides two numbers. Such a macro is provided by the `fp` package and also by `pgfmath` library of the `TikZ` package. The following comment is from the `fp` package

```
\def\FP@div#1#2.#3.#4\relax#5.#6.#7\relax{%
 % [...] algorithmic idea (for x>0, y>0)
 %   - %determine \FP@shift  such that
 %      y*10^\FP@shift < 100000000
 %                     <=y*10^(\FP@shift+1)
 %   - %determine \FP@shift' such that
 %      x*10^\FP@shift'< 100000000
 %                     <=x*10^(\FP@shift+1)
 %   - x=x*\FP@shift'
 %   - y=y*\FP@shift
 %   - \FP@shift=\FP@shift-\FP@shift'
 %   - res=0
 %   - while y>0 %fixed-point representation!
 %   -   \FP@times=0
 %   -   while x>y
 %   -     \FP@times=\FP@times+1
 %   -     x=x-y
 %   -   end
 %   -   y=y/10
 %   -   res=10*res+\FP@times/1000000000
 %   - end
 %   - %shift the result according to \FP@shift
```

The `pgfmath` library implements the macro in a similar way, but limits the number of shifts that it does. These macros highlight the state of affairs in writing TEX macros. Even simple things like multiplying two numbers are hard; you either have to work extremely hard to circumvent the programming limitations of TEX, or, more frequently, hope that someone else has done the hard work for you. In LuaTEX, such a function can be written using the `/` operator (I will explain the details later):

```
\def\DIVIDE#1#2{\directlua{tex.print(#1/#2)}}
```

Thus, with LuaTEX ordinary users can write simple macros; and, perhaps more importantly, can read and understand macros written by TEX wizards.

Since the LuaTEX project started it has been actively supported by ConTEXt.[1] These days, the various "How do I write such a macro" questions on the ConTEXt mailing list are answered by a solution that uses Lua. I present a few such examples in this article. I have deliberately avoided examples about fonts and non-Latin languages. There is already quite a bit of documentation about them. In this article, I want to highlight how to use LuaTEX to write macros that require some "flow control": randomized outputs, loops, and parsing.

2 Interaction between TEX and Lua

To a first approximation, the interaction between TEX and Lua is straightforward. When TEX (i.e., the LuaTEX engine) starts, it loads the input file in memory and processes it token by token. When TEX encounters `\directlua`, it stops reading the file in memory, *fully expands the argument of* `\directlua`, and passes the control to a Lua instance. The Lua instance, which runs with a few preloaded libraries, processes the expanded arguments of `\directlua`. This Lua instance has a special output stream which can be accessed using `tex.print(...)`. The function `tex.print(...)` is just like the Lua function `print(...)` except that `tex.print(...)` prints to a "TEX stream" rather than to the standard output. When the Lua instance finishes processing its input, it passes the contents of the "TEX stream" back to TEX.[2] TEX then inserts the contents of the "TEX stream" at the current location of the file that it was reading; expands the contents of the "TEX stream"; and continues. If TEX encounters another `\directlua`, the above process is repeated.

As an exercise, imagine what happens when the following input is processed by LuaTEX.[3]

```
\directlua%
  {tex.print("Depth 1
           \\directlua{tex.print('Depth 2')}")}
```

[1] Not surprising, as two of LuaTEX's main developers — Taco Hoekwater and Hans Hagen — are also the main ConTEXt developers.

[2] The output of `tex.print(...)` is buffered and not passed to TEX until the Lua instance has stopped.

[3] In this example, I used two different kinds of quotations to avoid escaping quotes. Escaping quotes inside `\directlua` is tricky. The above was a contrived example; if you ever need to escape quotes, you can use the `\startluacode ... \stopluacode` syntax explained later.

On top of these LuaTEX primitives, ConTEXt provides a higher level interface. There are two ways to call Lua from ConTEXt. The first is a macro \ctxlua (read as ConTEXt Lua), which is similar to \directlua. (Aside: It is possible to run the Lua instance under different name spaces. \ctxlua is the default name space; other name spaces are explained later.) \ctxlua is good for calling small snippets of Lua. The argument of \ctxlua is parsed under normal TEX catcodes (category codes), so the end of line character has the same catcode as a space. This can lead to surprises. For example, if you try to use a Lua comment, everything after the comment gets ignored.

```
\ctxlua
  {-- A lua comment
   tex.print("This is not printed")}
```

This can be avoided by using a TEX comment instead of a Lua comment. However, working under normal TEX catcodes poses a bigger problem: special TEX characters like &, #, $, {, }, etc., need to be escaped. For example, # has to be escaped with \string to be used in \ctxlua.

```
\ctxlua
  {local t = {1,2,3,4}
   tex.print("length " .. \string#t)}
```

As the argument of \ctxlua is fully expanded, escaping characters can sometimes be tricky. To circumvent this problem, ConTEXt defines a environment called \startluacode ... \stopluacode. This sets the catcodes to what one would expect in Lua. Basically only \ has its usual TEX meaning, the catcode of everything else is set to other. So, for all practical purposes, we can forget about catcodes in \startluacode ... \stopluacode. The above two examples can be written as

```
\startluacode
  -- A lua comment
   tex.print("This is printed.")
  local t = {1,2,3,4}
   tex.print("length " .. #t)
\stopluacode
```

This environment is meant for moderately sized code snippets. For longer Lua code, it is more convenient to write the code in a separate Lua file and then load it using Lua's dofile(...) function.

ConTEXt also provides a Lua function to conveniently write to the TEX stream. The function is called context(...) and it is equivalent to tex.print(string.format(...)).

Using the above, it is easy to define TEX macros that pass control to Lua, do some processing in Lua, and then pass the result back to TEX. For example, a macro to convert a decimal number to hexadecimal can be written simply, by asking Lua to do the conversion.

```
\def\TOHEX#1{\ctxlua{context("\%X",#1)}}
\TOHEX{35}
```

The percent sign had to be escaped because \ctxlua assumes TEX catcodes. Sometimes, escaping arguments can be difficult; instead, it can be easier to define a Lua function inside \startluacode ... \stopluacode and call it using \ctxlua. For example, a macro that takes a comma separated list of strings and prints a random item can be written as

```
\startluacode
  userdata = userdata or {}
  math.randomseed( os.time() )
  function userdata.random(...)
    context(arg[math.random(1, #arg)])
  end
\stopluacode

\def\CHOOSERANDOM#1%
  {\ctxlua{userdata.random(#1)}}

\CHOOSERANDOM{"one", "two", "three"}
```

I could have written a wrapper so that the function takes a list of words and chooses a random word among them. For an example of such a conversion, see the "sorting a list of tokens" page on the LuaTEX wiki [2].

In the above, I created a name space called userdata and defined the function random in that name space. Using a name space avoids clashes with the Lua functions defined in LuaTEX and ConTEXt.

In order to avoid name clashes, ConTEXt also defines independent name spaces of Lua instances. They are

user	: a private user instance
third	: third party module instance
module	: ConTEXt module instance
isolated	: an isolated instance

Thus, for example, instead of \ctxlua and \startluacode ... \stopluacode, the user instance can be accessed via the macros \usercode and \startusercode ... \stopusercode. In instances other than isolated, all the Lua functions defined by ConTEXt (but not the inbuilt Lua functions) are stored in a global name space. In the isolated instance, all Lua functions defined by ConTEXt are hidden and cannot be accessed. Using these instances, we could write the above \CHOOSERANDOM macro as follows

```
\startusercode
  math.randomseed( global.os.time() )
```

```
  function random(...)
    global.context(arg[math.random(1, #arg)])
  end
\stopusercode

\def\CHOOSERANDOM#1%
  {\usercode{random(#1)}}
```

Since I defined the function `random` in the `user` instance of Lua, I did not bother to use a separate name space for the function. The Lua functions `os.time`, which is defined by a LuaTEX library, and `context`, which is defined by ConTEXt, needed to be accessed through a `global` name space. On the other hand, the `math.randomseed` function, which is part of Lua, could be accessed as is.

A separate Lua instance also makes debugging slightly easier. With `\ctxlua` the error message starts with

```
! LuaTeX error <main ctx instance>:
```

With `\usercode` the error message starts with

```
! LuaTeX error <private user instance>:
```

This makes it easier to narrow down the source of error.

Normally, it is best to define your Lua functions in the `user` name space. If you are writing a module, then define your Lua functions in the `third` instance and in a name space which is the name of your module. In this article, I will simply use the default Lua instance, but take care to define all my Lua functions in a `userdata` name space.

Now that we have some idea of how to work with LuaTEX, let's look at some examples.

3 Arithmetic without using a abacus

Doing simple arithmetic in TEX can be extremely difficult, as illustrated by the division macro in the introduction. With Lua, simple arithmetic becomes trivial. For example, if you want a macro to find the cosine of an angle (in degrees), you can write

```
\def\COSINE#1%
  {\ctxlua(context(math.cos(#1*2*pi/360))}
```

The built-in `math.cos` function assumes that the argument is specified in radians, so we convert from degrees to radians on the fly. If you want to type the value of π in an article, you can simply say

```
$\pi = \ctxlua{context(math.pi)}$
```

or if you want less precision (notice the percent sign is escaped)

```
$\pi = \ctxlua{context("\%.6f", math.pi)}$
```

4 Loops without worrying about expansion

Loops in TEX are tricky because macro assignments and macro expansion interact in strange ways. For example, suppose we want to typeset a table showing the sum of the roll of two dice and want the output to look like this

(+)	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

The tedious (but faster!) way to achieve this is to simply type the whole table by hand. For example,

```
\bTABLE
  \bTR \bTD $(+)$ \eTD \bTD 1 \eTD ... ... \eTR
  \bTR \bTD   1   \eTD \bTD 2 \eTD ... ... \eTR
  ...   ...   ...   ...   ...   ...
  ...   ...   ...   ...   ...   ...
\eTABLE
```

It is however natural to want to write this table as a loop, and compute the values. A first ConTEXt implementation using the recursion level might be:

```
\bTABLE
 \bTR
  \bTD $(+)$ \eTD
  \dorecurse{6}
   {\bTD \recurselevel \eTD}
  \eTR
\dorecurse{6}
 {\bTR
     \bTD \recurselevel \eTD
     \edef\firstrecurselevel{\recurselevel}
  \dorecurse{6}
  {\bTD
   \the\numexpr\firstrecurselevel+\recurselevel
  \eTD}%
  \eTR}
\eTABLE
```

However, this does not work as expected, yielding all zeros. A natural table stores the contents of all the cells, before typesetting it. But it does not expand the contents of its cell before storing them. So, at the time the table is actually typeset, TEX has already finished the `\dorecurse` and `\recurselevel` is set to 0.

The solution is to place `\expandafter` at the correct location(s) to coax TEX into expanding

the \recurselevel macro before the natural table stores the cell contents. The difficult part is figuring out the exact location of \expandafters. Here is a solution that works:

```
\bTABLE
 \bTR
  \bTD $(+)$ \eTD
  \dorecurse{6}
   {\expandafter \bTD \recurselevel \eTD}
  \eTR
\dorecurse{6}
{\bTR
 \edef\firstrecurselevel{\recurselevel}
 \expandafter\bTD \recurselevel \eTD
 \dorecurse{6}
 {\expandafter\bTD
  \the\numexpr\firstrecurselevel+\recurselevel
  \relax
  \eTD}
 \eTR}
\eTABLE
```

We only needed to add three \expandafters to make the naive loop work. Nevertheless, finding the right location pf \expandafter can be frustrating, especially for a non-expert.

By contrast, in LuaTEX writing loops is easy. Once a Lua instance starts, TEX does not see anything until the Lua instance exits. So, we can write the loop in Lua, and simply print the values that we would have typed to the TEX stream. When the control is passed to TEX, TEX sees the input as if we had typed it by hand. Consequently, macro expansion is no longer an issue. For example, we can get the above table by:

```
\startluacode
context.bTABLE()
  context.bTR()
    context.bTD() context("$(+)$") context.eTD()
    for j=1,6 do
      context.bTD() context(j) context.eTD()
    end
  context.eTR()
  for i=1,6 do
    context.bTR()
    context.bTD() context(i) context.eTD()
    for j=1,6 do
      context.bTD() context(i+j) context.eTD()
    end
    context.eTR()
  end
context.eTABLE()
\stopluacode
```

The Lua functions such as context.bTABLE() and context.bTR() are just abbreviations for running context ("\\bTABLE"), context("\\bTR"), etc. See the ConTEXt Lua document manual for more details about such functions [3]. The rest of the code is a simple nested for-loop that computes the sum of two dice. We do not need to worry about macro expansion at all!

5 Parsing input without exploding your head

In order to get around the weird rules of macro expansion, writing a parser in TEX involves a lot of macro jugglery and catcode trickery. It is a black art, one of the biggest mysteries of TEX for ordinary users.

As an example, let's consider typesetting chemical molecules in TEX. Normally, molecules should be typeset in text mode rather than math mode. For example, $H_2SO_4^-$, can be input as H\low{2}SO\lohi{4}{--}. Typing so much markup can be cumbersome. Ideally, we want a macro such that we type \molecule{H_2SO_4^-} and the macro translates this into H\low{2}SO\lohi{4}{--}. Such a macro can be written in TEX as follows.

```
\newbox\chemlowbox
\def\chemlow#1%
  {\setbox\chemlowbox
          \hbox{{\switchtobodyfont[small]#1}}}

\def\chemhigh#1%
  {\ifvoid\chemlowbox
      \high{{\switchtobodyfont[small]#1}}%
   \else
      \lohi{\box\chemlowbox}
           {{\switchtobodyfont[small]#1}}
   \fi}

\def\finishchem%
  {\ifvoid\chemlowbox\else
      \low{\box\chemlowbox}
   \fi}

\unexpanded\def\molecule%
  {\bgroup
   \catcode`\_=\active \uccode`\~=`\_
       \uppercase{\let~\chemlow}%
   \catcode`\^=\active \uccode`\~=`\^
       \uppercase{\let~\chemhigh}%
   \dostepwiserecurse {65}{90}{1}
   {\catcode \recurselevel = \active
    \uccode`\~=\recurselevel
        \uppercase{\edef~{\noexpand\finishchem
    \rawcharacter{\recurselevel}}}}%
    \catcode`\-=\active \uccode`\~=`\-
        \uppercase{\def~{--}}%
    \domolecule }%

\def\domolecule#1{#1\finishchem\egroup}
```

This monstrosity is a typical TEX parser. Ap-

propriate characters need to be made active; occasionally, `\lccode` and `\uccode` need to be set; signaling tricks are needed (for instance, checking if `\chemlowbox` is void); and then magic happens (or so it seems to a flabbergasted user). More sophisticated parsers involve creating finite state automata, which look even more monstrous.

With LuaTEX, things are different. LuaTEX includes a general parser based on PEG (parsing expression grammar) called `lpeg` [4]. This makes writing parsers in TEX much more comprehensible. For example, the above `\molecule` macro can be written as

```
\startluacode
userdata = userdata or {}

local lowercase   = lpeg.R("az")
local uppercase   = lpeg.R("AZ")
local backslash   = lpeg.P("\\")
local csname      = backslash * lpeg.P(1)
                  * (1-backslash)^0
local plus        = lpeg.P("+") / "\\textplus "
local minus       = lpeg.P("-") / "\\textminus "
local digit       = lpeg.R("09")
local sign        = plus + minus
local cardinal    = digit^1
local integer     = sign^0 * cardinal
local leftbrace   = lpeg.P("{")
local rightbrace  = lpeg.P("}")
local nobrace     = 1 - (leftbrace + rightbrace)
local nested      = lpeg.P {leftbrace
                  * (csname + sign + nobrace
                  + lpeg.V(1))^0   * rightbrace}
local any         = lpeg.P(1)

local subscript   = lpeg.P("_")
local superscript = lpeg.P("^")
local somescript  = subscript + superscript

local content     = lpeg.Cs(csname + nested
                          + sign + any)

local lowhigh     = lpeg.Cc("\\lohi{%s}{%s}")
                    * subscript   * content
                    * superscript * content
                    / string.format
local highlow     = lpeg.Cc("\\hilo{%s}{%s}")
                    * superscript * content
                    * subscript   * content
                    / string.format
local low         = lpeg.Cc("\\low{%s}")
                    * subscript   * content
                    / string.format
local high        = lpeg.Cc("\\high{%s}")
                    * superscript * content
                    / string.format
local justtext    = (1 - somescript)^1
local parser      = lpeg.Cs((csname + lowhigh
                          + highlow + low
                          + high + sign + any)^0)

userdata.moleculeparser = parser

function userdata.molecule(str)
    return parser:match(str)
end
\stopluacode

\def\molecule#1%
  {\ctxlua{userdata.molecule("#1")}}
```

This is more verbose than the TEX solution, but is easier to read and write. With a proper parser, I do not have to use tricks to check if either one or both _ and ^ are present. More importantly, anyone (once they know the Lpeg syntax) can read the parser and easily understand what it does. This is in contrast to the implementation based on TEX macro jugglery which require you to implement a TEX interpreter in your head to understand.

6 Conclusion

LuaTEX is removing many TEX barriers: using system fonts, reading and writing Unicode files, typesetting non-Latin languages, among others. However, the biggest feature of LuaTEX is the ability to use a high-level programming language to program TEX. This can potentially lower the learning curve for programming TEX.

In this article, I have mentioned only one aspect of programming TEX: macros that manipulate their input and output some text to the main TEX stream. Many other kinds of manipulations are possible: LuaTEX provides access to TEX boxes, token lists, dimensions, glues, catcodes, direction parameters, math parameters, etc. The details can be found in the LuaTEX manual [1].

7 References

[1] LuaTEX reference manual, http://www.luatex.org/documentation.html

[2] Sorting a list of tokens, in the Joy of LuaTEX. http://luatex.bluwiki.com/go/Sort_a_token_list

[3] Hans Hagen, "CLD: ConTEXt Lua document", http://www.pragma-ade.com/general/manuals/cld-mkiv.pdf

[4] Lpeg: Parsing Expression Grammars for Lua, http://www.inf.puc-rio.br/~roberto/lpeg/lpeg.html

◇ Aditya Mahajan
adityam (at) umich dot edu

BibTeX meets relational databases

Dedicated to the memory of
Edgar Frank "Ted" Codd (1923–2003) and
James Nicholas "Jim" Gray (1944–2007)

Nelson H. F. Beebe

Abstract

After giving some background and comments on the BibTeX bibliographic database system, we discuss the problem of searching large collections of such data. We briefly describe how relational databases are structured and queried.

Portable new programs, bibtosql and bibsql, are introduced and their use is illustrated. The first handles the conversion of BibTeX data to input for three free, popular, and portable, database systems. The second provides a uniform and simple user interface for issuing search queries to any of the supported backend databases.

We finish with a discussion of the contributions of the two late computer scientists to whom this article is dedicated.

Contents

1 Introduction

Oren Patashnik's BibTeX has been in wide use in the TeX and LaTeX communities now for more than two decades after it was introduced in the *LaTeX User's Guide and Reference Manual* (Lamport 1985, Appendix B). The first mention of BibTeX in that book is at the bottom of page 72, where we find

> With LaTeX, you can either produce the list of sources yourself or else use a separate program called BibTeX to generate it from information contained in a bibliographic *database*.

The last word on that page has been emphasized here, and the current online edition of the *Oxford English Dictionary* defines it like this:

> **database**, n. ('deɪtəbeɪs) [f. DATA n. pl. + BASE n]
>
> 1. A structured collection of data held in computer storage; esp. one that incorporates software to make it accessible in a variety of ways; transf., any large collection of information.
> 2. Special Comb.: database management, the organization and manipulation of data in a database; database management system, a software package that provides all the functions required for database management; abbrev. DBMS s.v. D III.
> 3. database manager = database management system above; database system, a database together with a database management system.

The earliest mention recorded in the word history in that entry is from a technical memo of System Development Corporation, California in 1962 that describes it like this:

> A 'data base' is a collection of entries containing item information that can vary in its storage media and in the characteristics of its entries and items.

As often happens in English, increased use of a compound word has collapsed it into a single word.

2 An overview of BibTeX

BibTeX is a specialized database engine that takes from the `.aux` file a list of bibliography files, bibliographic style information, and a reference to a publication, all encoded in TeX commands like these:

```
\bibdata{bibsql,master}
\bibstyle{alpha}
\citation{Lamport:1985:LDP}
```

BibTeX locates bibliography files on a user-definable search path, and searches each file in turn for a BibTeX entry identified by the specified citation label, terminating the search at the first match. It also finds the first instance of the style file, here `alpha.bst`, in another user-specified search path. It then decodes the BibTeX entry to collect fields such as `author`, `title`, `year`, and so on, and then executes functions in the style file that format the publication data, usually with TeX markup, and write it to the `.bbl` file. When TeX or LaTeX next process the document, the expansion of the `\bibliography` macro includes commands to read the `.bbl` file and typeset the entry, as illustrated in the **References** section at the end of this article.

The key observation here is that by suitable *explicit* and *implicit* markup, computer software can attach meaning to strings of characters.

The BibTeX entry specifies the document authors as a list, with author names separated by the word *and* in the string value of the `author` key. In most cases, no additional markup is needed, because BibTeX contains hard-coded algorithms for parsing an author name into four parts: *first*, *von*, *last*, and *junior*. Thus, BibTeX recognizes the name *Ludwig von Beethoven, Sr.* if it is coded as **`von Beethoven, Sr., Ludwig`**, and BibTeX can convert it, if required, to *Beethoven, Ludwig von, Sr.*, and even abbreviate it to *Beethoven, L. von, Sr.* or *von Beethoven, L., Sr.*, depending on format settings in the style file. Name reordering is needed both for citation styles that require it, and also for sorting of reference list entries. Abbreviation of all but the family name is also common in some bibliography styles.

Embedded commas and *and* in author names are handled by minimal extra markup in the form of enclosing braces, such as in the corporate author **`{Chapman and Hall, Ltd.}`**. Some styles convert titles to lowercase letters after the first word, but braces prevent downcasing of proper names and/or German text in titles (recall that nouns in German, and pre-1948 Danish, are always capitalized). BibTeX interprets a macro name at the start of a braced group as a font change, rather than protection against lettercase conversion. An extra level of braces restores the protection, as in the name of a soil-living amoeba popular in biological research: **`{{\em Dictyostelium discoideum}}`**.

BibTeX's processing handles most personal names in Western Europe and the English-speaking world, but it does not deal with the case of Hungarian, and some Asian languages, where the family name comes first: **`Erd\H{o}s P\'al`** and **`K'ung Fu-Tzu`** are usually known as *Paul Erdős* and *Confucius* in English. Abbreviations of Spanish surnames that include both paternal and maternal family names, in that order, like **`Juan Vald{\'e}s Rodr{\'\i}guez`**, are also problematic, since that name can be shortened to *Juan Valdés R.*, or *J. Valdés R.*, or just *J. Valdés*. Some South Indians, and some people from a few other cultural groups, carry only a single name: *Sureshchander* and *Luqi* are examples found in computer-science bibliography articles. The current version of BibTeX, 0.99c, is dated 4 April 1988, but a final version of BibTeX is planned, and its author expects to incorporate additional markup to handle these difficult cases (Patashnik 2003).

Although standard BibTeX styles recognize a dozen or so standard key names, the system permits other keys to be used: their values are simply ignored if the style file does not reference them. Consequently, BibTeX markup has proved *extensible*, and this author has even employed it for representing equipment-inventory records and for editorial tracking of journal-article submissions.

The flexibility of BibTeX key/value markup, and its separation of most of the formatting job into (potentially) user-programmable style files, has proved of great value. The TeX Live 2008 software distribution includes 277 distinct BibTeX style files, and collections of millions of entries in BibTeX markup exist in some communities. The reference list in this article is typeset according to a new style, `acmtrans-v2.bst`, developed by the ACM for their publications in computer science, and as-yet absent from TeX Live. A few publisher Web sites, notably in the areas of computer science, electrical engineering, mathematics, and physics, can return dynamically-constructed BibTeX entries as search results.

3 BibTeX style-file language

BibTeX's style-file language is unique, and it uses reverse Polish notation, like some Hewlett–Packard calcula-

tors, the *Forth* programming language, and the *PostScript*™ page-description language. Such languages are easily parsed with a stack onto which operands are pushed by the lexical scanner, and then popped off the stack during the execution of operators encountered by the scanner. Parsing then requires no lookahead at all, but simply recognition of tokens, one at time, with the parser action either a push or an execute.

In particular, this language design means that functions are called with their arguments already in place on the stack, but the problem is, they could have been put there at any earlier time, and may not be visible at the location of the operator in the source code. This makes errors of incorrect type or number of arguments hard to diagnose, and even harder to find in the source code at some earlier level in the call tree.

While efficient to execute, such languages are horrid for humans to code in, except possibly for tiny programs. Consequently, some researchers have developed other markup schemes and bibliography-style descriptions. The *TUGboat* journal archives contain descriptions of work by James Alexander (Alexander 1986; Alexander 1987), Jean-Michel Hufflen (Hufflen 2003b; Hufflen 2003a), Tristan Miller (Miller 2005), and Robert Burgess (Burgess and Siren 2007). Patrick W. Daly's custom-bib system (Daly 1994; Viton 2000) in the TEX Live distribution generates dozens of BIBTEX style files from a generic file, and provides support for variants in many languages other than English.

Had BIBTEX been developed somewhat later after the importance of scripting languages like awk, JavaScript, lua, perl, php, python, and ruby was better appreciated, it could have been built on those software technologies. Hans Hagen and Taco Hoekwater are now doing this with ConTEXt and lua (Hoekwater 2007), creating a new TEX-like engine named LuaTEX whose internals are much more accessible to the author, document designer, and programmer.

4 BIBTEX grammar and software tools

Lamport's description of BIBTEX in his cited book is informal, but BIBTEX's markup language can be defined with a rigorous grammar, and tools have been developed to check for adherence to that grammar, for parsing BIBTEX files into easier-to-handle data streams, for prettyprinting and syntax checking BIBTEX files, and for converting Web pages from numerous library catalogs and publishers into BIBTEX entries (Beebe 1993a; Beebe 1993b; Beebe 2004a).

A recent, and as-yet-undescribed, software tool, cattobib, builds on international conventions for library-catalog markup and network catalog services to convert catalog entries to rough BIBTEX format (Beebe 2004b). While the conversion is syntactically accurate and generally acceptable to BIBTEX, a certain amount of manual cleanup is almost always required because of irregular and sloppy library cataloging. cattobib searches one or more user-specified catalogs sequentially, and can communicate with large national libraries of several countries. An obvious, although as-yet-unimplemented, extension is to perform the lookups in parallel.

5 The search problem

As the collection of data in BIBTEX markup grows, it becomes important to be able to search that data efficiently. More than a decade ago, this author developed bibsearch (Beebe 1997), leveraging advanced search technology developed by an international team of researchers and described in their *Managing Gigabytes* books (Witten, Moffat, and Bell 1994; Witten, Moffat, and Bell 1999). Their mg search engine stores documents in a highly-compressed in-memory index, and matches a user-supplied list of words, possibly qualified with *and*, *or*, and *not* operators, against the documents. It then returns an ordered list of documents that most closely match the word list. The search is astonishingly fast, often under a millisecond on modern workstations, even for a collection of a half-million BIBTEX entries.

Unfortunately, as so often happens with university research projects, interests changed, students graduated, and software development ceased before mg acquired the necessary feature of subdocument searching. With bibsearch, it is not possible to specify that a search string must, or must not, lie in, say, the `author` field. Such a facility is necessary if one is to quickly identify articles about, say, the *Knuth–Bendix* theorem without also finding articles by those authors themselves. As a result, bibsearch often returns much more than one would like, forcing the human searcher to visually examine a large list of returned BIBTEX entries, or save the results for searching with a text editor, to find the one desired.

BIBTEX was developed about the time that computer networking was evolving from the original Arpanet to what became the Internet, and still several years before the World-Wide Web, and the search engines that began with Archie at McGill University (Emtage and Deutsch 1992), Gopher at the University of Minnesota (Lindner 1993; Anklesaria, McCahill, Lindner, Johnson, Torrey, and Albert 1993), and AltaVista at Digital Equipment Cor-

poration (Seltzer, Ray, and Ray 1997). The first of these is eponymous with a long-running American comic strip (Archie Comics 2008), and was followed later by support tools named after strip companions Veronica and Jughead (Jonzy's Universal Gopher Hierarchical Extracter And Digger, created by a friend and colleague at the University of Utah) (Harris 1993).

At the time of writing this, three large search engines at Google, Microsoft, and Yahoo! dominate the field. They are hosted on thousands of geographically-distributed computers running highly-customized standard operating systems with specialized filesystems. They run Web crawlers to collect files from Web and FTP servers and they handle search queries with massive parallelism and sophisticated search algorithms that combine the history of previous searches with information about the number, and trustworthiness, of links to Web pages from other Web pages.

The search engine companies make their money from targeted advertising, and from fees paid by Web sites to increase the frequency of crawler visits. Their historical stock market performance makes an interesting case study of the successful commercialization of software technology.

The Google search algorithm is called *PageRank*, and the TUG bibliographic archives record more than two dozen publications about it in journals in computer science, mathematics, and oriental languages. Here, we cite only the earliest and latest of them (Page, Brin, Motwani, and Winograd 1998; Brezinski and Redivo-Zaglia 2008). A key component of the PageRank algorithm is the mathematical algorithm known as the *Singular Value Decomposition* (SVD), originally developed in 1965 by the late Gene Golub (1932–2007) at Stanford University, near where Don Knuth would later invent TEX and METAFONT, and by William Kahan at the University of California, Berkeley (Golub and Kahan 1965; Chan, Greif, and O'Leary 2007). These two researchers have strongly influenced modern numerical mathematics, and Kahan is a key figure in the improvement of floating-point arithmetic on computers (Beebe 2007a; Beebe 2007b). Comprehensive BIBTEX bibliographies of their publications can be found in the BibNet Project archives at `http://www.math.utah.edu/pub/bibnet/`.

Many computer users today consider Internet access a critical component of their environment, and routinely use search engines, even for tasks as mundane as finding the nearest designer caffeine-delivery service or where a new movie is shown. While putting BIBTEX files on the Web makes them accessible to Internet search engines, that action does not resolve the issue of how to restrict searches to particular subfields of entries, not does it provide speedy access to new data without substantial cost. The remainder of this article describes a significant solution to the BIBTEX search problem.

6 Relational databases

In their early days, computers had just two main applications: numerical computation, and data processing and management. The human brain stores information, but it is unreliable, has comparatively slow retrieval times, cannot be backed up reliably or quickly to other brains, and eventually, expires. Computers can solve the storage, retrieval, and backup problems if their human companions keep them clean, cool, and supplied with energy, and occasionally migrate their data to newer storage technologies.

Database systems were originally developed in the 1960s, but the early ones were inflexible, and searching could require a sequential scan of the entire corpus of data. However, in 1969, IBM researcher E. F. Codd proposed a new organization, called the *relational model* (Codd 1969; Codd 1970), and that idea eventually revolutionized computer database software.

Codd's essential idea was to consider the data as a set of tables, where each table row is a single logical item. For example, in a parts catalog, one row might contain columns labeled *customer demand*, *description*, *part number*, *price*, *profit margin*, *quantity available*, *supplier*, and *total sales*. Software can then *create a view* of a subset of the data by choosing only certain columns for consideration, with rows possibly further selected by applying constraints like *supplier in or near Madrid* and *price under €100*. Data subsetting can separate public information, such as part number, description, and price that might be recorded in a catalog for customers, from proprietary information that an organization uses for competitive advantage.

If at least one column in each row holds a unique identifier that distinguishes that row from all other rows, then it is possible to *join* tables by creating a new table that appends columns to rows that share a common unique identifier. This operation can be generalized to allow *union* (include all columns), *intersection* (include only common columns), *difference* (include only columns present in the first table and absent from the second), and *symmetric difference* (include only columns present in a single table). These operations are fundamental in the mathematical field of *set theory*, once introduced in American schools as "New Math" (Kline 1973). That idea was delightfully parodied by Tom Lehrer in a song of that name (Lehrer 1965), and later brought to the London (1980)

and New York (1981) stages in the play *Tomfoolery* (Macintosh 1980). For those interested in the mathematical background of relational databases, there is a recent book on that subject (de Haan and Koppelaars 2007).

It is precisely the idea of a unique identifier that provides great power for the collection and management of huge sets of data. While part numbers are possibly innocuous, personal identifiers like biometrics (blood groups, DNA sequences, facial photographs, fingerprints, retinal scans, voice patterns, ...), taxpayer numbers, pension-plan numbers, credit-card numbers, vehicle-license numbers, telephone numbers, house numbers, e-mail addresses, highway toll e-passes, and so on have made it possible for governments and commercial organizations to amass and collate enormous amounts of information about most of the population of the developed world, with a complete loss of privacy (Rambam 2008). The database *join* operation allows this material to be collected and sifted so that, for example, a medical-insurance company might maximize profits by insuring only customers who are both healthy and wealthy. The dangers of collecting information about people are well-chronicled in the book *IBM and the Holocaust* (Black 2002), and any study of human history soon shows that power and victory often go to those with the best information technology.

7 Structured Query Language (SQL)

By the mid-1980s, it became clear that, at least for users, standardization of the search interface to database systems was desirable. Vendors eventually settled on what is now known as the *Structured Query Language*, usually abbreviated as SQL, and pronounced either by its letters (ess cue ell) or like the noun *sequel*.

There have been several national, government, international, and industry standards for SQL, including ANSI (X3.168-1989, X3.135-1992, 9579-2-1993, 9075-3-1995, and 9075-4-1996), FIPS (127:1990, 127-2:1993, 193:1995), ISO/IEC, (9075:1987, 9075:1989, 9075:1992, 9075:2003, and 13249:2007), and X/Open (CAE 1994) (X/Open 1994). Many 2008-vintage SQL systems claim conformance to most of the 1992 ISO Standard. There are more than 500 books about SQL, and BibTeX entries for many of them can be found at `http://www.math.utah.edu/pub/tex/bib/index-table-s.html#sqlbooks`. The prolific writer, Christopher J. Date, has produced dozens of books about databases, and four editions of one of them chronicle the development of SQL (Date 1987; Date 1989; Date and Darwen 1993; Date and Darwen 1997), and eight editions of another cover database systems in general (Date 2004). There are also very readable treatments of the use of SQL in the books by Judith Bowman and her co-authors (Emerson, Darnovsky, and Bowman 1989; Bowman, Emerson, and Darnovsky 1993; Bowman, Emerson, and Darnovsky 1996; Bowman, Emerson, and Darnovsky 2001; Bowman 2001). For deeper coverage of database theory and practice, see the O'Neils' treatise (O'Neil and O'Neil 2001).

The language SQL offers much to criticize, and Ted Codd was never happy with it. One of members of the ISO SQL Standard Committee wrote this about it (Date and Darwen 1993, p. 374):

> I lay most of the blame for the following problems firmly on SQL itself. If the world needs a better database standard, it should prepare itself to move to a better database *language*.

Everyone who has searched an online bookstore or library catalog has almost certainly used SQL, but that use is generally hidden behind a fill-in-the-box screen interface that destroys much of the true power of the language.

Three observations about SQL are relevant for our discussion here:

- The **S** in **SQL** is for *Structured*, not *Standard*. Life with SQL is very different from life with conventional programming languages, like Fortran, C, C++, C#, and Java, for which highly-portable subsets and international standards exist.

  With SQL, there are far too many gratuitous and irritating differences between database systems that serve no purposes but customer lock-in, database programmer job security, and user confusion.
- SQL is *not* a programming language: there are no variables, no loops, no conditionals, and no user-defined functions or procedures, although particular databases may offer some of these extended features. When programmability is required, it is conventional to embed calls to an SQL interface library, either in a high-level compiled language like Ada, C, C++, C#, COBOL, or Java, or in a scripting language, such as JavaScript, perl, php, python, or ruby, all of which have database module interfaces.
- SQL is complex, large, and powerful, but it is not difficult to learn a useful tiny subset: indeed, in Section 10, we concentrate primarily on just *one* statement in the language.

8 BibTeX database tables

The first step in making BibTeX entries amenable to storage in a relational database is to identify column headings that are likely to be useful. All that we need to do is to mentally rotate the entry by 90 degrees, so that the entry keys

become column names, their values become table-cell data, and the citation label becomes a (possibly-)unique identifier characterizing the table row. Because a BIBTEX file often contains string definitions for use in entries, they too are fodder for another table.

At the time of writing this, we have implemented three distinct tables for BIBTEX data:

`bibtab` with data from BIBTEX entries under these columns:

`authorcount`, `editorcount`, `pagecount`, `bibtype`, `filename`, `label`, `author`, `editor`, `booktitle`, `title`, `crossref`, `journal`, `volume`, `type`, `number`, `institution`, `organization`, `publisher`, `school`, `address`, `edition`, `pages`, `day`, `month`, `monthnumber`, `year`, `CODEN`, `DOI`, `ISBN`, `ISBN13`, `ISSN`, `LCCN`, `MRclass`, `MRnumber`, `MRreviewer`, `bibdate`, `bibsource`, `bibtimestamp`, `note`, `series`, `URL`, `abstract`, `keywords`, `remark`, `subject`, `TOC`, `ZMnumber`, `entry`.

`strtab` a table of BIBTEX string definitions with three columns:

`key`, `value`, `entry`.

`namtab` a table of personal names used in `author` and `editor` fields, with just two columns:

`name`, `count`.

Cells in the `*count` columns have integer values, and the cells in all of the remaining columns contain variable-length character strings, where the permitted string lengths are well beyond what is needed for typical BIBTEX entries.

The `entry` column in the first two tables contains the *exact* original text from the BIBTEX file.

It is conventional in modern database systems to use a special value, *NULL*, to represent indeterminate, unknown, or unset values. NULL is distinguishable from both numeric zeros and empty strings, and most database operators, and functions that act on column data, ignore rows with NULL values. Thus, if you take the sum or average of a numerical column, only rows with numbers in them participate. Similarly, if you select data with an expression **`year > '2005'`**, rows with a NULL year do not match that expression.

The query facilities described in Section 10 allow testing for the presence or absence of NULL values with expressions like ***`column`*** **`IS NULL`** and ***`column`*** **`IS NOT NULL`**.

While most of the key names in the `bibtab` table are standard ones in BIBTEX, a few are not. These include at least these keys:

`CODEN` Chemical Abstracts *CODE Name* for serial publications.

`DOI` *Digital Object Identifier*, a *unique and permanent* name for the master copy of an electronic document. If it does not look like an Internet URL, convert it to one by prefixing it with `http://dx.doi.org/`.

`ISSN` *International Standard Serial Number*: eight digits, optionally written as two groups of four digits separated by a hyphen.

`ISBN` old-style *International Standard Book Number*: 10 digits, the last of which may be *X* or *x*. Optional hyphens separate it into four digit groups that define the country or language, publisher, book number, and a final check digit for error detection.

`ISBN13` new-style (since 2007) *International Standard Book Number*: 13 digits, the last of which may be *X* or *x*. This contains the old ten-digit ISBN, with a new prefix of `978-`, and a revised check digit. When the supply of unused ISBNs is exhausted, a new prefix `979-` will be assigned, and such numbers will then not be convertible to the old ten-digit form.

`LCCN` US *Library of Congress Catalog Number*, used for book identification at many US libraries, and at some libraries in other countries.

`MRclass` list of American Mathematical Society *Math Reviews* five-character subject classification codes.

`MRnumber` *Math Reviews* database number.

`MRreviewer` *Math Reviews* reviewer name.

`bibdate` Unix date string in the formats `Sat Oct 25 20:15:00 MDT 2008` or `Sat Oct 25 20:15:00 2008`; it records a timestamp of the last significant change to the BIBTEX entry data.

`bibsource` arbitrary text indicating source(s) of bibliographic data, often as a list of Internet URLs or Z3950 library-catalog addresses.

`bibtimestamp` alternate representation of `bibdate` value in the form `2008.10.25 20:15:00 MDT`, for which lexicographic sort order is also time order.

URL	Internet *Uniform Resource Locator(s)* for retrieval of an electronic form of the document.
remark	additional commentary about the entry that is not intended to be included in a typeset reference list.
subject	subject classification phrases.
TOC	table of contents of the publication.
ZMnumber	European Mathematical Society *Zeitschrift für Mathematik* database number.

9 BibTeX/SQL software tools

This author has created two new programs, bibtosql and bibsql, to get BibTeX files into relational databases with an SQL interface and make that data searchable. The programs are freely available in various distribution formats at `http://www.math.utah.edu/pub/bibsql`. Both are written as portable shell scripts, and the first of them uses another program in the awk scripting language, which is standardly available on all Unix platforms, and versions can be found for Microsoft Windows as well. The awk program needs no changes whatever to move between systems, and at a given site, it is likely that the shell scripts, once configured, are identical across local systems.

The first program, bibtosql, handles the conversion of BibTeX files to input for one of three commonly-used, and freely-available, SQL database systems: *MySQL*, *PostgreSQL*, and *SQLite*. All three are reasonably portable, and they can be installed from source code, or from binary distributions, on all common desktop platforms. In 2007, Sun Microsystems purchased MySQL, but the source code is expected to remain available. We also investigated use of the *IBM DB2*, *Ingres*, and *Microsoft SQL Express* databases, but abandoned further work with them because their limits on the sizes of strings or table rows make them unusable for our needs, and they are not available on the broad range of platforms that the others are. Also, their SQL input syntax is somewhat different from the three that we support, and that would introduce unnecessary confusion for users of bibsql.

MySQL and PostgreSQL are large and complex systems that implement a client/server model of database access. A single server can typically handle simultaneous requests from many client programs elsewhere on the network. Large sites may already have one or more instances of them installed and operational, making it easy to create yet another database for them to serve.

The third is much simpler, and recommended for single-user systems, and small groups. An SQLite database consists of a *single file* that is independent of operating system, CPU architecture, and storage byte order. Once created, an SQLite database file can be used on any system where SQLite can be compiled, and is ideally suited for distribution on read-only media such as CD-ROMs or DVDs, or via the Web.

The SQLite software is *public domain* and may be used for any purpose, and redistributed, without restriction. This makes it of interest, for example, for future TeX Live distributions.

The user interface to bibtosql looks like this:

```
% bibtosql --help
Usage: /usr/local/bin/bibtosql
                [ --author ]
                [ --create ]
                [ --database dbname ]
                [ --help ]
                [ --version ]
                [ --server ( MySQL | psql | PostgreSQL | SQLite ) ]
                [ -- ]
                BibTeXfiles or <infile
                >outfile

Options may begin with either one or two dashes, and lettercase in
options and database names is ignored.

Examples:
        bibtosql --create *.bib                       | sqlite3 bibtex.db
        bibtosql --create --server sqlite *.bib       | sqlite3 bibtex.db
        bibtosql --create --server psql *.bib         | psql ...
        bibtosql --create --server mysql *.bib        | mysql ...
```

```
bibtosql -c *.bib                              | sqlite3 bibtex.db
bibtosql -c -s p *.bib                         | psql ...
bibtosql -c -s m *.bib                         | mysql ...

bibtosql newstuff.bib                          | sqlite3 bibtex.db

bibclean *.bib | biborder | bibtosql --create | sqlite3 bibtex.db
```

The manual pages for bibtosql contain extensive details about setting up the supported databases, and they need not be repeated here. SQLite is by far the easiest to use, since a single command of the type shown in the first example does the job.

The second program, bibsql, provides the user interface to the SQL database system, and hides the messy details of where the data are located, and how the SQL client program communicates with the server. Its help option reports:

```
% bibsql --help
Usage: /usr/local/bin/bibsql
                [ --author ]
                [ --command ' command1; command2; ... ' ]
                [ --database dbname ]
                [ --help ]
                [ --options ' ... server options ...' ]
                [ --server ( MySQL | psql | PostgreSQL | SQLite ) ]
                [ --user dbuser ]
                [ --version ]

Options may begin with either one or two dashes, and lettercase in
options and database names is ignored.

Examples:
        bibsql
        sqlite> ... user input here ...

        bibsql -s psql
        psql>  ... user input here ...
```

The default database is currently called **bibtex**, but the option is there to allow access to others, should that prove useful.

At this author's site, SQL clients for all three database systems are operational on about 25 flavors of Unix, including Apple Mac OS X, covering all of the common CPU architectures of the last decade. This demonstrates the impressive portability of these database systems. The Cygwin environment on Microsoft Windows XP and Vista systems provides clients for PostgreSQL and SQLite; see `http://www.cygwin.com/`. MySQL for Windows can be downloaded from its vendor Web site, and its client program runs nicely with Cygwin. All three servers supply essentially the same data, but they vary in performance and also in search and display facilities. The default server is always SQLite, but users of bibsql should have few surprises if they select a different database server with a command-line option.

The bibsql distribution includes sample C programs that can communicate with one of each of the supported SQL databases, providing a useful coding template for those who wish to incorporate bibliographic-database access directly into their own programs.

10 Using SQL with bibsql

It is now time to look at exactly how searches are done in SQL. The language has an imperative English-like grammar that is nevertheless precise enough to allow unambiguous interpretation by computer software. Once bibsql has started the SQL client program, all further user interaction is with that program: there is no input filtering whatever.

For user convenience, lettercase is ignored in SQL commands. Lettercase is ignored in search strings as well, except with PostgreSQL, where nonstandard special operators are required.

Character strings in SQL are written with surrounding single quotes, like this: **'SQL'**. Some SQL clients also support quotation-mark delimiters, **"SQL"**.

To represent a single quote inside a string, double it: **'O''Neil'** is the name *O'Neil*. In MySQL, use a backslash escape instead: **'O\'Neil'**.

Long strings can be split into separate strings that are joined with the double-bar concatenation operator. For example, these two expressions evaluate to the same string:

```
'Aloisius Baldwin Chadwick, IV'
'Aloisius ' || 'Baldwin ' || 'Chadwick, ' || 'IV'
```

Comments in SQL take two forms. An Ada-style double hyphen starts a remark that continues to end of line or end of file, whichever comes first. MySQL additionally requires that the double hyphen be followed by at least one space to eliminate an ambiguity in the SQL expression grammar. Otherwise, C-style `/* ... */` comments can span one or more lines. Comments cannot be nested, and can appear inside commands anywhere that whitespace can (except inside character strings).

The format of the output depends on the database system. For SQLite, it can be set by a dotted command described in the help system like this:

```
.mode MODE ?TABLE?  Set output mode where MODE is one of:
                      csv    Comma-separated values
                      column Left-aligned columns.  (See .width)
                      html   HTML <table> code
                      insert \SQL\ insert statements for TABLE
                      line   One value per line
                      list   Values delimited by .separator string
                      tabs   Tab-separated values
                      tcl    TCL list elements

.nullvalue STRING   Print STRING in place of NULL values

.output FILENAME    Send output to FILENAME

.output stdout      Send output to the screen

.separator STRING   Change separator used by output mode and .import

.width NUM NUM ...  Set column widths for "column" mode
```

The default is to separate cell values by vertical bars, with each table row output on a single line (unless the string data contain linebreaks). Examples are given later in this section. However, other output styles chosen by the SQLite `.mode` command make it easy to output the data in formats suitable for input to other databases, spreadsheets, and Web pages.

The simplest query asks for a return of all records, where the asterisk means all data, and a final semicolon is required to terminate the command:

```
select * from bibtab;
1||9|article|acm-turing-awards.bib|Perlis:1967:SAS|
Alan J. Perlis|||The Synthesis of Algorithmic Systems||
j-J-ACM|14||1|||||||1--9||jan|1|1967|JACOAH|
http://doi.acm.org/10.1145/321371.321372|||0004-5411 OR 00045411|
||||Mon Dec 05 19:37:58 1994||1994.12.05 19:37:58 ???|
|||||This is the 1966 ACM Turing Award Lecture, and the first award.||||
@Article{Perlis:1967:SAS,
  author =       "Alan J. Perlis",
  title =        "The Synthesis of Algorithmic Systems",
  journal =      j-J-ACM,
```

```
  volume =       "14",
  number =       "1",
  pages =        "1--9",
  month =        jan,
  year =         "1967",
  CODEN =        "JACOAH",
  DOI =          "http://doi.acm.org/10.1145/321371.321372",
  ISSN =         "0004-5411",
  bibdate =      "Mon Dec 05 19:37:58 1994",
  acknowledgement = ack-nhfb,
  remark =       "This is the 1966 ACM Turing Award Lecture, and the
                 first award.",
}
...
```

Here, we wrapped the long first line for readability, and showed only the first record returned. The order of records depends on the database creation and update history, and is unpredictable without further specifications.

Next, we limit the output to just three specified columns, and further limit the selection with a `WHERE` clause:

```
select year, author, title from bibtab
    where author like '%Perlis%' and year = '1967';
1967|Alan J. Perlis|The Synthesis of Algorithmic Systems
1967|B. A. Galler and A. J. Perlis|A proposal for definitions in ALGOL
```

Because the command is long, we wrote it on separate lines.

The `LIKE` operator is followed by a string pattern wherein *percent* represents zero or more characters, *underscore* a single character, and, in MySQL and SQLite, lettercase is *ignored*. Use `NOT LIKE` to negate the comparison. These matches are against the full text of the string, so patterns usually need to begin and end with a percent.

To search for a literal percent or underscore, double them in the search pattern. With PostgreSQL, use the nonstandard operator `ILIKE`, or its synonym `~~*`, for case-insensitive searches.

To make string comparisons case sensitive in SQLite, set a library option like this:

```
pragma case_sensitive_like = on;
```

Set it to `off` to restore the default behavior. SQLite recognizes synonyms `true`, `yes`, and `1` for `on`, and `false`, `no`, and `0` for `off`.

SQLite also has a `GLOB` operator that uses Unix pathname matching, where *asterisk* matches zero or more characters, *question mark* matches a single character, and lettercase is always *significant*. Unfortunately, there is no standard support in SQLite for regular-expression matching like that provided by many other Unix tools and scripting languages, and some other SQL systems.

We can also use string-equality tests, but then the match must be exact, including lettercase:

```
select year, author, title from bibtab
    where author = 'Alan J. Perlis'
    order by year;
1958|Alan J. Perlis|Announcement
1963|Alan J. Perlis|Computation's development critical to our society
1967|Alan J. Perlis|The Synthesis of Algorithmic Systems
1969|Alan J. Perlis|Introduction to extensible languages
1978|Alan J. Perlis|The American side of the development of Algol
1986|Alan J. Perlis|Two Thousand Words and Two Thousand Ideas --- The 650 at Carnegie
```

In the `ORDER BY` clause, the operand can be a comma-separated list of column names or ordinal numbers: **`ORDER BY 1, 3`** sorts by the first column, and when that column has the same values, by the third column.

Such a search is likely to miss many entries belonging to alternate spellings of the selected author, such as:

```
select year, author, title from bibtab
    where author = 'A. J. Perlis'
    order by year;
1964|A. J. Perlis|A format language
```

```
1964|A. J. Perlis|Programming of digital computers
1964|A. J. Perlis|How should ACM publish computer research?
1966|A. J. Perlis|A Forum on Algorithms: A new policy for algorithms?
1975|A. J. Perlis|Introduction to Computer Science
1981|A. J. Perlis|The American side of the development of ALGOL
```

It also misses entries where Perlis is one of multiple authors, since SQL string-equality tests always compare against the full string values.

Use the namtab table to find the frequencies and variations of an author or editor name in the database:

```
select count, name from namtab
    where name like '%Steele%'
    order by 1 desc;
```

```
15|Guy L. Steele Jr.
3|Guy L. Steele
2|Guy L. Steele, Jr.
1|G. L. Steele, Jr.
1|G. Steele
```

A more complex query requests unique output from a range of years, sorts the data in descending-year order, and limits the number of records returned by the command to just five:

```
select distinct year, author, title from bibtab
    where author like '%D%Knuth'
    and '1955' < year
    and year < '1970'
    order by year desc
    limit 5;
```

```
1969|Donald E. Knuth|Seminumerical Algorithms
1968|Donald E. Knuth|Very magic squares
1967|Donald E. Knuth|The Remaining Trouble Spots in ALGOL 60
1966|Donald E. Knuth|Errata: ‘‘Additional comments on a problem in concurrent ...’’
1966|Donald E. Knuth|Letter to the Editor: Additional comments on a concurrent ...
```

Some SQL systems permit the quotes around the year values to be omitted, but strictly, they are strings, not integers, since they occasionally contain a list or range of years.

The expression

```
'1955' < year and year < '1970'
```

can also be written as

```
year between '1956' and '1969'
```

The endpoints of the BETWEEN operator are included in the range test.

When multiple logical operators are used in an expression without disambiguating parentheses, AND is evaluated before OR. Thus, **a and b or c and d** is treated as if it were written **(a and b) or (c and d)**. When in doubt about the meaning of a complex expression, parenthesize!

The SELECT command can be used for rudimentary expression evaluation in SQL, simply by omitting cell selections, as shown in Table 1. Notice that character-string collation differs among the three systems. Numerical expressions are evaluated in floating-point arithmetic if at least one of the operands contains a decimal point, except in MySQL. Although the underlying arithmetic is IEEE 754, the treatment of NaN, Infinity, subnormal numbers, and zero divides is irregular and unreliable in the three systems. Also, large numbers are not handled sensibly. In the view of this author, floating-point arithmetic is not to be trusted in these systems.

The larger SQL systems offer a wide range of numerical and string functions, similar to those of many programming languages. However, SQLite has only a minimal repertoire of built-in functions. Here are examples of some of the SQL functions that can operate on the returned results, including ones that just report counts, averages, extrema, and sums. We pose them as questions in prose, then express them in a SELECT statement to retrieve an answer.

Table 1: Expression evaluation in SQL systems. SQLite lacks the illustrated functions from the square root to end of table.

SQL statement	Value	Remark
`select 'ABC' > 'DEF';`	0	
`select 'ABC' < 'ABCDEF';`	1	
`select 'a' < 'A', 'a' = 'A', 'a' > 'A';`	0,1,0	MySQL
`select 'a' < 'A', 'a' = 'A', 'a' > 'A';`	t, f, f	PostgreSQL
`select 'a' < 'A', 'a' = 'A', 'a' > 'A';`	0,0,1	SQLite
`select length('The Lion of TeX');`	15	
`select substr('The Lion of TeX', 13, 3);`	'TeX'	
`select 1.0 / 3.0;`	0.333333333333333	
`select 1.0 / 3;`	0.333333333333333	
`select 1 / 3;`	0	
`select 1 / 3;`	0.3333	in MySQL
`select 1/3 - 0.33333333333333333;`	−0.00000000033333333	in MySQL
`select 1.0/3.0 - 0.33333333333333333;`	0	in MySQL
`select null IS NULL;`	1	
`select max(1,2,3);`	3	SQLite only
`select min(1,2,3);`	1	SQLite only
`select (1 + sqrt(5))/2;`	1.6180339887499	
`select sin(22);`	−0.00885130929040388	
`select pi();`	3.14159265358979	
`select rand();`	0.54459485182967	in [0,1] in MySQL
`select random();`	0.751422385685146	in [0,1] in PostgreSQL
`select random();`	−5624361857185364588	in $[-2^{63}, 2^{63}-1]$ in SQLite
`select log(100);`	4.6051701859881	natural logarithm in MySQL
`select log10(1000);`	3	base-10 logarithm in MySQL
`select ln(10);`	4.60517018598809	natural logarithm in PostgreSQL
`select log(100);`	2	base-10 logarithm in PostgreSQL

How many BibTeX entries are in the database?

```
select count(*) from bibtab;
417349
```

How many titles are longer than 250 characters?

```
select count(length(title)) from bibtab where length(title) > 250;
772
```

What fraction of the entries have titles longer than 250 characters?

```
select '1 / ' || round(417349 / 772);
1 / 540.0
```

What is the average length of nonempty titles?

```
select avg(length(title)) from bibtab where length(title) > 0;
61.3055823817683
```

How long is the longest title in the database?

```
select max(length(title)) from bibtab;
1746
```

What is the average length in pages, rounded to the nearest integer, of the documents with known page counts?

```
select round(avg(pagecount)) from bibtab where pagecount > 0;
46.0
```

What is the largest page count in the database?

```
select max(pagecount) from bibtab;
4412
```

What is the largest number of editors in any document?

```
select max(editorcount) from bibtab;
18
```

What is the largest number of authors in any document?

```
select max(authorcount) from bibtab;
115
```

That number is unexpectedly large. *What is the citation label and title of that document?*

```
select label,title from bibtab where authorcount = 115;
Adiga:2002:OBS|An Overview of the BlueGene/L Supercomputer
```

Evidently, it takes a lot of people to build IBM's largest computer.

What is the largest number of authors of any book?

```
select max(authorcount) from bibtab where bibtype = 'book';
23
```

How small is the shortest BibTeX entry?

```
select min(length(entry)) from bibtab;
101
```

How big is the longest BibTeX entry?

```
select max(length(entry)) from bibtab;
19269
```

What is the average size of a BibTeX entry in the database?

```
select avg(length(entry)) from bibtab;
770.447505564887
```

How many pages are in the documents that have known page counts?

```
select sum(pagecount) from bibtab where pagecount > 0;
13415587
```

What are the numbers of documents in each month of publication?

```
select distinct count(month), lower(month) from bibtab
       where length(month) = 3
       group by lower(month)
       order by cast(monthnumber as number);
19111|jan
```

```
15411|feb
20423|mar
19398|apr
17489|may
21585|jun
19537|jul
16577|aug
20039|sep
18896|oct
16859|nov
20924|dec
```

This example shows how strings can be converted to lowercase with the **lower()** function (there is also an **upper()** function), and to numbers with the **cast()** function. It may also suggest that publisher staff prefer February, August, and November for their vacation time.

How many documents are there for each of the top ten publishers, ordering the results by descending counts?

```
select count(publisher), publisher from bibtab
    where length(publisher) > 0
    group by publisher
    order by count(publisher) desc
    limit 10;
```

```
5679|pub-SV
2154|pub-ORA
1820|pub-PROJECT-GUTENBERG
1593|pub-IEEE
1307|pub-AW
1076|pub-PH
891|pub-WILEY
838|pub-ACM
524|pub-PHPTR
425|pub-MCGRAW-HILL
```

How many documents are there for each of the top ten journals, ordering the results by descending counts?

```
select count(journal), journal from bibtab
    where length(journal) > 0
    group by journal
    order by count(journal) desc
    limit 10;
```

```
60788|j-LECT-NOTES-COMP-SCI
14540|j-J-MATH-PHYS
9988|j-CACM
8594|j-APPL-MATH-COMP
7896|j-SIGPLAN
7253|j-LINEAR-ALGEBRA-APPL
6435|j-THEOR-COMP-SCI
5344|j-COMPUTER
5335|j-MATH-COMPUT
4974|j-INFO-PROC-LETT
```

How many documents have a nonzero author count? (Remember that some document entries have only editors.)

```
select count(*) from bibtab where authorcount > 0;
```

```
406451
```

What percentage of the documents have one, two, . . . , five authors?

```
select round(100 * count(authorcount) / 406451) || '%', authorcount from bibtab
    where authorcount > 0
    group by authorcount
    order by count(authorcount) desc
    limit 5;
```

```
48.0%|1
28.0%|2
13.0%|3
5.0%|4
1.0%|5
```

The rule that NULL values are excluded from consideration in expressions does not apply to the special case of the **count(*)** expression, because it just counts rows, and no row is completely NULL.

The last four searches are complex queries that use the SQL GROUP BY feature to simultaneously compute counts of publishers and journals, producing a list of the top ten that occur most frequently in the database, and do a similar computation to find the percentage of publications with one to five authors.

To avoid the need to first compute the count of documents with authors, the last pair of queries can be rephrased as a single command with an embedded SELECT command, but we now make a further restriction.

What is the percentage of journal articles that have each of one to five authors?

```
select round(100 * count(authorcount) /
             (select count(*) from bibtab
                     where authorcount > 0 and
                           bibtype = 'article')) || '%',
           authorcount
    from bibtab
    where authorcount > 0 and
          bibtype = 'article'
    group by authorcount
    order by count(authorcount) desc
    limit 5;
```

```
47.0%|1
29.0%|2
14.0%|3
5.0%|4
1.0%|5
```

BIBTEX string abbreviations are commonly used for data that are repeated in many entries, particularly for journals, institutions, organizations, publishers, and addresses. The abbreviations can be retrieved from the strtab table with searches like these:

Which publishers are in the Wiley family?

```
select entry from strtab
    where key like 'pub-WILEY%'
      and key not like '%adr%';
```

```
@String{pub-WILEY               = "Wiley"}
@String{pub-WILEY-INTERSCIENCE  = "Wiley-In{\e-}ter{\e-}sci{\e-}ence"}
```

Which publishers have offices in the city of Boston?

```
select entry from strtab where value like '%Boston%' order by entry;
```

```
@String{pub-ALLYN-BACON:adr     = "Boston, MA, USA"}
```

```
@String{pub-AP-PROFESSIONAL:adr = "Boston, MA, USA"}
...
@String{pub-LITTLE-BROWN:adr    = "Boston, Toronto, London"}
@String{pub-MORGAN-KAUFMANN:adrbo = "Boston, MA, USA"}
```

How many of the document publishers are in the city of New York?

```
select count(value) from strtab where value like '%New York%';
59
```

What are the string names and addresses for publishers in the Birkhäuser family?

```
select distinct key from strtab
    where key like '%BIRK%'
    order by key;
pub-BIRKHAUSER
pub-BIRKHAUSER-BOSTON
pub-BIRKHAUSER-BOSTON:adr
pub-BIRKHAUSER:adr
```

It is possible, although complex, to combine searches in multiple tables to collect output data from all of them. Here is just one example, which also introduces another SQL feature of providing short aliases for database and table names within a single query.

Can we find the strings and BibTeX entry for the Classic Shell Scripting *book?*

```
.separator "\n"
select s.entry, t.entry, b.entry
       from bibtab b, strtab s, strtab t
       where b.author like '%Robbins%'
       and b.title like '%classic%'
       and b.publisher = s.key
       and b.address = t.key;
@String{pub-ORA-MEDIA           = "O'Reilly Media, Inc."}
@String{pub-ORA-MEDIA:adr       = "1005 Gravenstein Highway North,
                                   Sebastopol, CA 95472, USA"}
@Book{Robbins:2005:CSS,
  author =       "Arnold Robbins and Nelson H. F. Beebe",
  title =        "Classic Shell Scripting",
  publisher =    pub-ORA-MEDIA,
  address =      pub-ORA-MEDIA:adr,
  pages =        "xxii + 534",
  year =         "2005",
  ISBN =         "0-596-00595-4",
  ISBN-13 =      "978-0-596-00595-5",
  LCCN =         "QA76.76.063 R633 2005",
  bibdate =      "Tue Jul 12 16:13:16 2005",
  URL =          "http://www.oreilly.com/catalog/shellsrptg/",
  acknowledgement = ack-nhfb,
}
```

Changing the output separator to a newline suppresses unwanted additional decoration, producing output that can be copied directly into a BibTeX file.

To list the column fields in the SQLite database, and to get further help, use the commands

```
.schema bibtab
.help
```

The output of `.schema` is similar to the `CREATE TABLE` command shown in the manual pages for bibtosql.

For more on the command syntax of SQLite, consult its Web site documentation collection: `http://www.sqlite.org/docs.html`.

More details on additional features of MySQL and PostgreSQL searching and output formatting are provided in the manual pages for bibsql. Here, we only note that both provide for regular-expression pattern matching, and allow lettercase to be significant, or to be ignored. Because key is a built-in function in MySQL, when used as a column name in MySQL search expressions, it must be protected with back quotes, like this: `` `key` ``.

All three client programs record a command-history file in the user's home directory, and support the GNU readline library for command recall and editing. This feature is of great convenience, especially because it is common to repeat search commands after minor edits.

11 The impact of Ted Codd and Jim Gray

Since 1966, the *Association for Computing Machinery* (ACM) has awarded an annual prize for significant work that furthers the young field of computer science. This is known as the *Turing Award*, and it has come to be regarded as a kind of Nobel Prize for computer scientists. It is named after the renowned English mathematician Alan Mathison Turing, who produced early important work on artificial intelligence, complexity theory, the now-famous *Turing Machine* test, computer design, cryptography, and floating-point arithmetic, as well as in pure mathematics and mathematical biology. BibTeX biographies of Turing's complete works, and of the ACM Turing Awards, are available in the previously-cited BibNet Project and in the TUG bibliography archives at `http://www.math.utah.edu/pub/tex/bib/index-table-a.html#acm-turing-awards`.

The first award went to Alan Perlis (1922–1990) of Yale University for his important work on compiler construction and programming languages, notably, Algol. Some of his papers are mentioned in the search examples in Section 10.

The 1981 recipient was the English computer scientist Edgar Frank "Ted" Codd (1923–2003), who revolutionized computing with the idea of relational databases. He spent much of his career at IBM research laboratories in the USA, where he was influential in the development of IBM's *System R* database. One of the spinoffs of that work was the founding by Larry Ellison of Oracle Corporation, today the largest database vendor, and a fierce competitor of others in that field, particularly Microsoft. There is a tribute to Ted Codd in one of the chapters of a recent memorial volume (Date 2008), which notes in its first paragraph: *The entire relational database industry, now worth many billions of dollars a year, owes its existence to Ted's original work.* An earlier version appeared in a journal article (Date 2003), and there is a Web site containing pointers to his publications (McJones 2003).

The 1998 award went to James Nicholas "Jim" Gray (1944–2007), who worked in academia, and in industry at IBM, Tandem, DEC, and Microsoft. He was the first recipient of a Ph.D. from the Department of Computer Science at the University of California, Berkeley, a department that is one of three great leaders in the field (the others are MIT and Stanford), and the one that brought Unix and VAXes into academia. Jim was tragically lost without a trace on January 28, 2007, during a solo sailing trip in clear and calm weather to the Farallon Islands offshore west of San Francisco, California. A large satellite and sea rescue effort failed to find either him, or his boat, or any evidence whatever of what happened; see `http://openphi.net/tenacious/`.

In the view of many people in the field, Jim was the person who made relational databases really work. He made important contributions by applying database technology in the Microsoft Virtual Earth Project, and the SkyServer database for the Sloan Digital Sky Survey (SDSS) (see `http://www.sdss.org/`, `http://skyserver.sdss.org/`, and `http://doi.acm.org/10.1145/1400214.1400231`). SkyServer has revolutionized the field of astronomy, because it makes the large and expensive-to-create astronomical data archives freely available to all of the world's astronomers and astrophysicists, even those who are amateurs or hobbyists.

A conference in memory of Jim Gray was held at Berkeley on 31 May 2008 (see `http://www.eecs.berkeley.edu/IPRO/JimGrayTribute/`). Its proceedings are recorded in the June 2008 issue of ACM *SIGMOD Record*; see `http://www.math.utah.edu/pub/tex/bib/index-table-s.html#sigmod`. The November 2008 issue of *Communications of the ACM* has a charcoal-gray portrait of Jim on its cover, and articles about his work; see `http://portalparts.acm.org/1410000/1400214/fm/frontmatter.pdf`.

12 Conclusion

Connecting BibTeX to SQL databases has been a long-term goal of mine, because the output of bibsearch is often overwhelming due to its inability to narrow the search. However, bibsearch is much faster than any SQL database, and my concern was that an SQL interface to bibliographic data might be too sluggish to be useful. Fortunately,

advances in computers have been dramatic in the years since BIBTEX was first developed. With the database server running on a fast modern multicore or multiprocessor system, and particularly with SQLite and MySQL, search speeds with bibsql are acceptably fast, and often under a second.

The ability to restrict searches to subfields of BIBTEX entries is liberating. It facilitates asking questions that could not be answered with previous tools, as we illustrated in some of the search examples in Section 10. It also makes it much easier, and more reliable, to collect new entries for the many subject-specific bibliographies in the TUG archive, since the `WHERE` clause can use expressions that test the `bibtimestamp` and `filename` columns to select entries newer than a given date, while ignoring entries already in the subject bibliography file.

Exploratory searches of our large body of bibliographic data have also shown that it is quite easy to detect, report, and repair irregularities in the data, even down to such simple things as a missing period following an author/editor initial.

My hope is that future editions of the TEX Live software distributions will include bibsql and bibtosql, along with binaries for SQLite for all supported platforms, accompanied by URLs from which to retrieve current collections of bibliographic data in SQLite format. That way, TEX users the world over can enjoy the benefits of fast access, via SQL searches, to bibliographic data without the trouble of producing and maintaining that data themselves. The SQL learning experience with bibsql may then encourage them to think of other applications of SQL databases for their own interests and problems, and appreciate the lessons from Ted Codd and Jim Gray that relational databases are *really useful*.

The BIBTEX entries for the publications cited in the **References** section that follows were collected almost entirely with the help of bibsql, bibsearch, and cattobib.

References

James Alexander. 1986. Tlb: a reference setting package for TEX. *TUGboat* 7, 3 (Oct.), 138–140. ISSN 0896-3207.

James Alexander. 1987. Tlb: a reference setting package, update. *TUGboat* 8, 2 (July), 102–102. ISSN 0896-3207.

F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, and B. Albert. 1993. RFC 1436: The Internet Gopher Protocol (a distributed document search and retrieval protocol). (Mar.). See `ftp://ftp.internic.net/rfc/rfc1436.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc1436.txt`.

Archie Comics. 1941–2008. Archie Archive. Web site. See `http://www.archiecomics.com/`.

Nelson H. F. Beebe. 1993a. Bibliography prettyprinting and syntax checking. *TUGboat* 14, 3 (Oct.), 222–222. ISSN 0896-3207.

Nelson H. F. Beebe. 1993b. Bibliography prettyprinting and syntax checking. *TUGboat* 14, 4 (Dec.), 395–419. ISSN 0896-3207.

Nelson H. F. Beebe. 1997. `bibsearch`: search BIBTEX bibliography files. (May). See `http://www.math.utah.edu/pub/bibsearch`. Unpublished, but software released.

Nelson H. F. Beebe. 2004a. A bibliographer's toolbox. *TUGboat* 25, 1, 89–104. ISSN 0896-3207.

Nelson H. F. Beebe. 2004b. `cattobib`: convert Z39.50 library catalog server data to BIBTEX markup. (15 Nov.). See `http://www.math.utah.edu/pub/cattobib`. Unpublished, but software released.

Nelson H. F. Beebe. 2007a. Extending TEX and METAFONT with floating-point arithmetic. *TUGboat* 28, 3, 319–328. ISSN 0896-3207.

Nelson H. F. Beebe. 2007b. New directions in floating-point arithmetic. In *Computation in Modern Science and Engineering: Proceedings of the [Fifth] International Conference on Computational Methods in Science and Engineering 2007 (ICCMSE 2007), Corfu, Greece, 25–30 September 2007 (AIP Conference Proceedings (#963))*. Theodore E. Simos and George Maroulis (Eds.), Vol. 2A. American Institute of Physics, Woodbury, NY, USA, xxvi + 730 + 10 (vol. 2A) book pages, 155–158. ISBN 0-7354-0476-3 (set), 0-7354-0477-1 (vol. 1), 0-7354-0478-X (vol. 2). ISBN-13 978-0-7354-0476-2 (set), 978-0-7354-0477-9 (vol. 1), 978-0-7354-0478-6 (vol. 2). ISSN 0094-243X. LCCN Q183.9 .I524 2007. See `http://www.springer.com/physics/atoms/book/978-0-7354-0478-6`.

Edwin Black. 2002. *IBM and the Holocaust: the strategic alliance between Nazi Germany and America's most powerful corporation*. Three Rivers Press, New York, NY, USA. 551 pages. ISBN 0-609-80899-0 (paperback). ISBN-13 978-0-609-80899-3 (paperback). LCCN HD9696.2.U64 I253 2002.

Judith S. Bowman. 2001. *Practical SQL: the sequel*. Addison-Wesley, Reading, MA, USA. xv + 329 pages. ISBN 0-201-61638-6. ISBN-13 978-0-201-61638-5. LCCN QA76.73.S67 B695 2001.

Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. 1993. *The practical SQL handbook: using Structured Query Language*, (second ed.). Addison-Wesley, Reading, MA, USA. xvii + 453 pages. ISBN 0-201-62623-3. ISBN-13 978-0-201-62623-0. LCCN QA76.73.S67 E54 1993.

Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. 1996. *The practical SQL handbook: using Structured Query Language*, (third ed.). Addison-Wesley Developers Press, Reading, MA, USA. xxvi + 454 pages. ISBN 0-201-44787-8. ISBN-13 978-0-201-44787-3. LCCN QA76.73.S67 B69 1996.

Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. 2001. *The practical SQL handbook: using SQL variants*, (fourth ed.). Addison-Wesley, Reading, MA, USA. xxxvi + 469 pages. ISBN 0-201-70309-2. ISBN-13 978-0-201-70309-2. LCCN QA76.73.S67 B688 2001.

C. Brezinski and M. Redivo-Zaglia. 2008. Rational extrapolation for the PageRank vector. *Math. Comp.* 77, 263 (July), 1585–1598. CODEN MCMPAF. ISSN 0025-5718 (paper), 1088-6842 (electronic). See http://www.ams.org/mcom/2008-77-263/S0025-5718-08-02086-3/home.html.

Robert Burgess and Emin Gün Siren. 2007. CrossTEX: A modern bibliography management tool. *TUGboat* 28, 3, 342–349. ISSN 0896-3207.

Raymond H. Chan, Chen Greif, and Dianne P. O'Leary (Eds.). 2007. *Milestones in matrix computation: the selected works of Gene H. Golub with commentaries*. Oxford University Press, Walton Street, Oxford OX2 6DP, UK. xi + 565 + 3 pages. ISBN 0-19-920681-3. ISBN-13 978-0-19-920681-0. LCCN QA188 .G67 2007. See http://www.loc.gov/catdir/enhancements/fy0737/2007276086-d.html.

E. F. Codd. 1969. *Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks*. Research Report RJ599, IBM Corporation, San Jose, CA, USA. (19 Aug.).

E. F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 6 (June), 377–387. DOI http://doi.acm.org/10.1145/362384.362685. CODEN CACMA2. ISSN 0001-0782. http://dblp.uni-trier.de/db/journals/cacm/Codd70.html. Reprinted in (Stonebraker 1988, pp. 5–15).

Patrick W. Daly. 1994. Customised BIBTEX styles. Web-accessible software. See http://www.ctan.org/tex-archive/help/Catalogue/entries/custom-bib.html.

C. J. Date. 1987. *A guide to the SQL standard: a user's guide to the standard relational language SQL*. Addison-Wesley, Reading, MA, USA. xiv + 205 pages. ISBN 0-201-05777-8 (paperback). ISBN-13 978-0-201-05777-5 (paperback). LCCN QA76.9.D3 D36951 1987.

C. J. Date. 1989. *A guide to the SQL standard: a user's guide to the standard relational language SQL*, (second ed.). Addison-Wesley, Reading, MA, USA. xii + 228 pages. ISBN 0-201-50209-7. ISBN-13 978-0-201-50209-1. LCCN QA76.9.D3 D24 1989.

C. J. Date. 2003. Edgar F. Codd: a tribute and personal memoir. *SIGMOD Record (ACM Special Interest Group on Management of Data)* 32, 4 (Dec.), 4–13. DOI http://doi.acm.org/10.1145/959060.959061. CODEN SRECD8. ISSN 0163-5808.

C. J. Date. 2004. *An introduction to database systems*, (eighth ed.). Pearson/Addison Wesley, Boston, MA, USA. xxvii + 983 + 22 pages. ISBN 0-321-19784-4. ISBN-13 978-0-321-19784-9. LCCN QA76.9.D3 D3659 2003.

C. J. Date. 2008. Edgar F. Codd. In *Memorial Tributes*. Vol. 12. National Academy of Engineering, Washington, DC, USA, 80–87 (of xiv + 362). ISBN 0-309-12639-8. ISBN-13 978-0-309-12639-7. See http://books.nap.edu/openbook.php?record_id=12473&page=80.

C. J. Date and Hugh Darwen. 1993. *A guide to the SQL Standard: a user's guide to the standard relational language SQL*, (third ed.). Addison-Wesley, Reading, MA, USA. xvii + 414 pages. ISBN 0-201-55822-X. ISBN-13 978-0-201-55822-7. LCCN QA76.9.D3 D3695 1993.

C. J. Date and Hugh Darwen. 1997. *A guide to the SQL standard: a user's guide to the standard database language SQL*, (fourth ed.). Addison-Wesley, Reading, MA, USA. xxii + 522 pages. ISBN 0-201-96426-0. ISBN-13 978-0-201-96426-4. LCCN QA76.73.S67 D38 1997.

Lex de Haan and Toon Koppelaars. 2007. *Applied mathematics for database professionals*. Apress, Berkeley, CA, USA. xxviii + 376 pages. ISBN 1-59059-745-1. ISBN-13 9781590597453. LCCN QA76.9.D3 H239 2007. See http://www.loc.gov/catdir/toc/fy0804/2008299427.html. This book provides the mathematical background of logic and set theory for relational databases.

Sandra L. Emerson, Marcy Darnovsky, and Judith S. Bowman. 1989. *The practical SQL handbook: using Structured Query Language*. Addison-Wesley, Reading, MA, USA. xiii + 393 pages. ISBN 0-201-51738-8. ISBN-13 978-0-201-51738-5. LCCN QA76.73.S67 E54 1989.

Alan Emtage and Peter Deutsch. 1992. archie — *An Electronic Directory Service for the Internet*. Technical report, McGill University, Montréal, Québec, Canada.

G. H. Golub and W. Kahan. 1965. Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2, 2, 205–224. ISSN 0887-459X. Reprinted in (Chan, Greif, and O'Leary 2007).

Judi Harris. 1993. Mining the Internet: Networked Information Location Tools: Gophers, Veronica, Archie, and Jughead. *Computing Teacher* 21, 1 (1 Aug.), 16–19. ISSN 0278-9175.

Taco Hoekwater. 2007. LuaTEX. *TUGboat* 28, 3, 312–313.

Jean-Michel Hufflen. 2003a. European bibliography styles and MlBIBTEX. *TUGboat* 24, 3, 489–498. ISSN 0896-3207.

Jean-Michel Hufflen. 2003b. MlBIBTEX's Version 1.3. *TUGboat* 24, 2, 249–262. ISSN 0896-3207.

Morris Kline. 1973. *Why Johnny can't add: the failure of the New Math*. Vintage Books, New York, NY, USA. xi + 208 pages. ISBN 0-394-71981-6. ISBN-13 978-0-394-71981-8. LCCN QA13 .K62 1974.

Leslie Lamport. 1985. *LaTeX—A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA. xiv + 242 pages. ISBN 0-201-15790-X. ISBN-13 978-0-201-15790-1. LCCN Z253.4.L38 L35 1986.

Tom Lehrer. 1965. New Math. In *That Was the Year That Was*. Vol. R/RS 6179. Reprise Records, Burbank, CA, USA. See http://dmdb.org/discographies/lehrer.disco.html,http://www.stlyrics.com/songs/t/tomlehrer3903/newmath185502.html. Audio recording.

P. (Paul) Lindner. 1993. *Internet Gopher User's Guide*. Tech. rep., University of Minnesota, Minneapolis, MN, USA. vi + 103 pages. See also RFC 1436.

Cameron Macintosh. 1980. Tomfoolery. Web site and stage play. See http://www.tomlehrer.org/tomlehrer/enter.html.

Paul R. McJones. 2003. Collected Works of E. F. Codd. Web site. See http://www.informatik.uni-trier.de/~ley/db/about/codd.html.

Tristan Miller. 2005. Biblet: A portable BibTeX bibliography style for generating highly customizable XHTML. *TUGboat* 26, 1, 85–96. ISSN 0896-3207.

Patrick O'Neil and Elizabeth O'Neil. 2001. *Database—principles, programming, and performance*, (second ed.). Morgan Kaufmann Publishers, Los Altos, CA 94022, USA. xxiv + 870 pages. ISBN 1-55860-580-0 (paperback), 1-55860-438-3 (hardcover). ISBN-13 978-1-55860-580-0 (paperback), 978-1-55860-438-4 (hardcover). LCCN QA76.9.D3 O489 2001. See http://www.loc.gov/catdir/toc/els033/99089041.html.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. *The PageRank Citation Ranking: Bringing Order to the Web*. Tech. rep., Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, USA. 17 pages. (29 Jan.). See http://dbpubs.stanford.edu/pub/1999-66;http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf.

Oren Patashnik. 2003. BibTeX yesterday, today, and tomorrow. *TUGboat* 24, 1, 25–30. ISSN 0896-3207.

Steven Rambam. 2008. The Grill: Privacy and Databases. *ComputerWorld* 42, 41 (13 Oct.), 18, 20. See http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=326821.

Richard Seltzer, Deborah S. Ray, and Eric J. Ray. 1997. *The AltaVista Search Revolution: How to find anything on the Internet*. Osborne/McGraw-Hill, Berkeley, CA, USA. xxii + 274 pages. ISBN 0-07-882235-1. ISBN-13 978-0-07-882235-3. LCCN TK5105.875 .I57 S44 1997.

Michael Stonebraker (Ed.). 1988. *Readings in Database Systems*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA. xii + 644 pages. ISBN 0-934613-65-6. ISBN-13 978-0-934613-65-1. LCCN QA76.9.D3 R4 1988.

Philip A. Viton. 2000. Getting Started with custom-bib: An introduction for SWP users. Web tutorial. (25 Aug.). See http://facweb.knowlton.ohio-state.edu/pviton/support/custom-bib.html.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1994. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, NY, USA. xiv + 429 pages. ISBN 0-442-01863-0. ISBN-13 978-0-442-01863-4. LCCN TA1637 .W58 1994.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*, (second ed.). Morgan Kaufmann Publishers, Los Altos, CA 94022, USA. xxxi + 519 pages. ISBN 1-55860-570-3. ISBN-13 978-1-55860-570-1. LCCN TA1637 .W58 1994. See http://www.math.utah.edu/pub/mg/.

X/Open (Ed.). 1994. *X/Open CAE specification*. X/Open Co., Reading, Berkshire, UK. xvi + 156, x + 66 pages. ISBN 1-872630-58-8, 1-872630-98-7, 0-13-353558-4 (Prentice Hall). ISBN-13 978-1-872630-58-8, 978-1-872630-98-4, 978-0-13-353558-7 (Prentice Hall). LCCN QA76.73.S67 X18 1994.

Mini-colophon: The body of this article is typeset with the fourier font package and the luximono typewriter font package, augmented with the eurosym and tipa packages for additional special characters. The bibliography style is acmtrans-v2.

◇ Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
WWW URL: http://www.math.utah.edu/~beebe
Telephone: +1 801 581 5254
FAX: +1 801 581 4148
Internet: beebe@math.utah.edu, beebe@acm.org, beebe@computer.org

TEX as an eBook reader

Kaveh Bazargan

Abstract

An important advantage of eBook readers is their ability to modify text size and page orientation for the most comfortable reading configuration. The eBook reader has to reformat the text on the fly and with minimum delay. Current eBook readers (e.g. Stanza on the iPhone) can do this reformatting, but cannot deal with complicated text such as mathematics. We have been experimenting with using TEX as the formatting engine. Of course it can handle complex mathematics, but it also creates the best line breaks of any eBook reader. We will report our experiments with using TEX as an ebook reader on the iPhone.

1 Problems in the eBook world

1.1 Format wars

There is a 'format war' going on in the world of eBooks. Several formats are proprietary, and several openly specified. This has produced confusion for the publishers, as they don't know which format to distribute their books on, and don't know whether or not they will have to redistribute in the future, in other formats. The XML format was originally envisaged for just this sort of format war. By using a single format which embodies all the logical structure and the content of the document, any other format, e.g. PDF, can be created automatically.

ePub is a comprehensive XML format for eBooks. In particular it supports MathML for mathematical text. So the ideal scenario would be that an eBook is saved in only one format, namely ePub, and an eBook reader would render this on the fly and present a readable view. We'll come back to this after discussing the workflow in our company.

1.2 Ugly output

One of the most compelling features of eBooks is that they can reflow the text to the user's taste. But the line breaking engine on these small devices is not that sophisticated, and the output does not look professionally typeset. For example, eBook readers on the iPhone show lots of bad breaks and large word gaps when the font size is increased. Again, we will come back to this.

2 Evolution of the workflow at River Valley Technologies

The main activity at our company is typesetting mathematical text. To explain most simply, we need to go from a TEX file submitted by the author, to a PDF file. Before the need for generation of XML, this was a straightforward process. We would put the TEX file into style, typeset it, proofread, etc. A few years ago, publishers started requesting XML, and rightly so. As we deal with mathematical material, we needed to use MathML in order to keep the structure for future re-use of XML.

But here is a problem with XML. It is easy to produce a file that parses and validates, but it is not easy to check the content. Checking the PDF, on the other hand is simple — it is called reading the document, or proofreading it. We had to find a way of guaranteeing the fidelity of the XML and PDF. The only solution we could think of was an automated way of generating the PDF from the XML. But this was a non-trivial task. XML deals only with content and logical structure. It does not deal with visual style, placement of figures, hyphenation, etc. There are other anomalies. For instance an XML file might list all figures at the top of the file, but the PDF will have them in the order that they appear.

Here is the workflow we came up with:

Author TEX → Structured TEX →
XML → Slave TEX → PDF

The structured TEX is not for typesetting, but simply uses TEX as markup for tagging elements; for instance `\firstname{John}` or `\journalyear{1985}`. This file is then transformed automatically into the XML. The program we use for this is a highly configured version of TEX4ht, written by the late Eitan Gurari. But for the purposes of the present discussion, it is the stages from the XML onwards that are important.

The XML file is transformed to a TEX file using XSLT. This is what we call a 'slave' TEX file which is not normally looked at. It is simply created in order to produce the final PDF file. The important point is that the translation of the XML to PDF is done with 100% automation. And the intermediate TEX file is not modified in any way, but simply run through TEX to produce the final PDF.

So we have two fully automated processes, one going from a TEX file to an XML file, and one from the XML to a second TEX file. The filters for these two processes were written independently, and checked rigorously. So let us suppose a proofreader is comparing the output from the author file and that from the final TEX file, and checks that the mathematical symbols match. It is then almost certain that the XML also matches. So the method gives us a high degree of confidence in the content of the XML.

3 Back to eBooks

So how does all this relate to eBooks? Well, we said in the section about 'format wars' that ideally ePub should be the one and only format, and it should be rendered on the fly by the eBook reader. So we need a rendering engine that renders ePub in near-real time. Well, our XML → Slave TEX → PDF is just such an engine. So we decided to try to implement this on an eBook reader. The only device which was accessible to us and had a software development kit available was the iPhone from Apple.

So there are two elements to the full process — transforming XML to TEX, and typesetting the TEX. The second seemed the most challenging and we sought the help of Jonathan Kew. Within a short time he managed to port the full TEX program to the iPhone. We decided to created a DVI file rather than a PDF, for speed and for compact file size. Jonathan also wrote a DVI reader for the device. The program runs very quickly indeed, and turning the iPhone on its side instantly reformats the output to the new page aspect ratio.

The transformation from ePub to the slave TEX has not yet been done, but it seems to be the easier part of the problem.

3.1 Quality of output

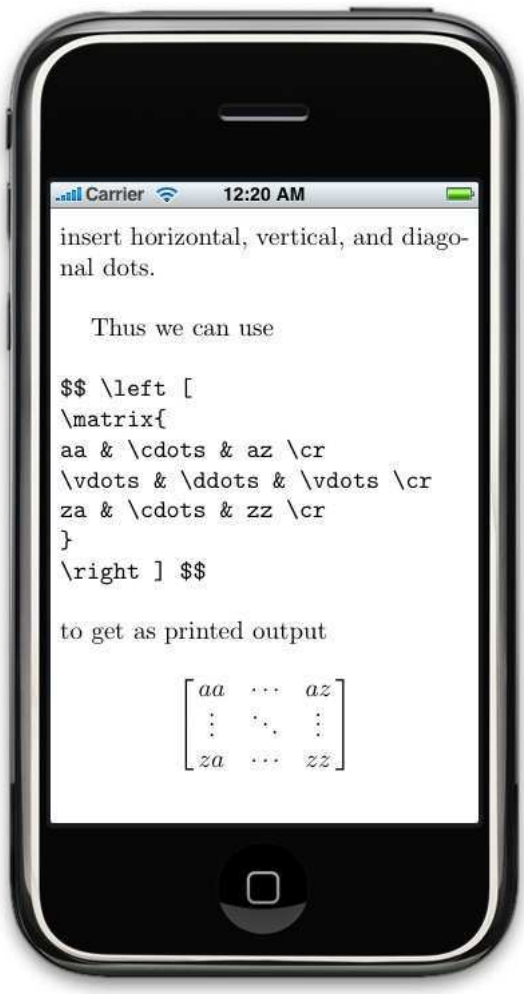
Regarding the 'ugly output' referred to above, we compared our output through TEX for a purely textual file, to the output from the other eBook readers on the iPhone and we think TEX does much better. Of course TEXies should not be surprised by this. We all know that the line breaking algorithm of TEX is the best!

4 Conclusions

Our automated workflow, from XML to PDF, can be modified and applied to eBook readers. The output looks good with good spacing and hyphenation, and only one format, ePub, need be produced by the publisher.

5 Acknowledgements

I am merely the 'Steve Jobs' of our company, others do the clever work and I take to the stage and take the credit! Here the main credit must go to Jonathan Kew who did the work reported here. Credit of course also to Radhakrishnan CV for discussions on the overall concept.



◇ Kaveh Bazargan
River Valley Technologies
kaveh (at) river-valley dot com
http://www.river-valley.com

From distribution to preservation of digital documents

Christian Rossi

Abstract

This article addresses the issues of conservation of (mostly textual) electronic documents. Its main objective is to describe the problems in terms of hardware and software mediators between the data and the user rather than focusing on formats. The point of view is that of a skeptic.

1 Introduction

The subject of this paper is the problem of preserving digital documents. It is mainly aimed at describing the problem of preservation in terms of the intermediary hardware and software between the data and the user, rather than in terms of file format. It is particularly focused on text documents and covers various aspects of the problem: hardware and software, migration, file formats, typography, XML, conversion, duration.

2 A digital document is not directly accessible to a human user

A key characteristic of the digital world is that there are many intermediaries between the medium on which digital information is stored and the user:

- the medium;
- a reader or drive;
- the operating system;
- a software application;
- the peripherals;
- operating instructions.

In the first place the medium (magnetic disk, cdrom, etc.) on which the information is physically held, then a drive used to convert the physical information into binary information, then the operating system used to group these 0s and 1s into files and directories. Finally, a software application that makes this file accessible to a human being via an audio or video peripheral or a printer, and this covers only the most standard cases. Not forgetting, of course, a person who knows how to use, and sometimes repair, all these intermediaries.

This is the major difference in relation to paper, which gives direct access to information.

In terms of preservation, a significant consequence of the existence of these intermediaries is that it is not only the medium itself which must be available, but the entire chain of consultation, and if not the original, then an equivalent. The length of this chain is a problem in itself, but its life expectancy, i.e. that of a computer and its peripherals, is rather short. At present, on average, it is common practice to replace a computer every three years, a new version of a standard operating system is available every year and security patches must be applied every month. And all this is necessary since for "old" equipment official maintenance soon stops or, if it is still available, is charged for at exorbitant rates. Ensuring the preservation of digital documents therefore now means choosing between constantly migrating all media and hardware and software chains, or creating living IT museums in which these intermediaries are kept in working condition.

In fact we can end up having to migrate not because the medium is nearing the limit of its physical life, but because of its technical/commercial obsolescence: the reader or drive is no longer on the market, is no longer maintained, and technical support is out of date. From this point of view there is no point in using a medium with a life expectancy of a hundred years, since in three years' time migration will be necessary. The technical/commercial life expectancy is shorter than the physical life expectancy.

Two other problems should be emphasized. The first is knowing what to migrate to — a decision that cannot be automated. Errors in this respect can give rise to many unnecessary and costly migration operations. The second is simply to guarantee, during successive migration operations, the preservation of document integrity. During migration, files are copied from one medium to another. Tools must then be used to check that the binary content of the files is identical. But if file format conversions are required, an automatic check appears to be impossible. How can a guarantee be given that a PostScript file converted to PDF will give an identical graphic output? The file is no longer the same, and neither is the consultation software.

Other solution: emulation of older technology, but this considerably extends and adds complexity to the chain between the document and the user, adding yet another intermediary. Another problem is the fact that the specifications of the technologies to be emulated are not necessarily in the public domain. But an emulator is also software like any other, which runs on a certain machine with a certain version of a certain operating system. In other words, emulation necessitates migrations of the emulator.

3 Autonomy and preservation

What will the life expectancy of a paper document be if it has to be de-acidified every three years to keep it intact? There is just one answer: the time

during which the people, the financial resources and the will to do so are all available. The situation is very similar with digital documents.

Taking into account the rapid evolution of techniques, the life expectancy of the hardware, software versions and file formats, it is clear that migrations and conversions of all sorts are inevitable. In other words, an electronic document is highly dependent on human intervention for its preservation — much more dependent than paper documents. We are a long way from the historic document that no one has touched for 200 years and that is rediscovered with great excitement. In an ideal world a document should be autonomous in terms of its preservation.

Moreover, an important constraint is that the preservation cost must be reasonable. This is not at present really compatible with frequent software and hardware migrations and the human operations involved in these migrations. But going beyond the issue of cost, relying on recurring and frequent human operations to ensure the preservation of documents in the long term does not appear to be reasonable.

4 Ease of use or life expectancy

If we consider the evolution of documentation media over time, we can observe two key points:

- ease of use has increased;
- life expectancy has decreased.

Effectively, while it is easier to use a sheet of paper than a tablet of stone, the life expectancy of the paper is clearly shorter. If we start with clay tablets, moving on to parchment, then paper, this evolution is very clear, an evolution that has also led to the real democratization of access to information, but an evolution that has also given rise to a real reduction in the life expectancy of documents. There has been a price to pay. We can summarize this situation by saying that for traditional media:

$$\text{ease of use} \times \text{life expectancy} = \text{constant}$$

Is the situation the same with digital media? In terms of ease of use, it is unquestionably very high. Digital is fantastic for creating, modifying, storing, searching, distributing, etc.

But in other respects digital is effectively complex, fragile, unstable ... and we have not had enough time to stand back and assess it. The high number of intermediaries between the medium and the human user does not make either access to information or its preservation any easier.

The situation therefore appears, for the moment, to be the same. And if we are actually in a universe where it is impossible to have only advantages, where what we gain in ease of use we lose in terms of life expectancy, it is better to use the digital media in full knowledge of the facts.

We are not talking here about stopping the use of digital media — as if we would want to — but of being realistic. For instance, we do not demand that paperback novels last forever, but no one would want to see them disappear. A paperback novel is used simply for its qualities, not criticized for its shortcomings. With digital media the same behaviour is reasonable: using it for its qualities of creation or distribution, while remaining aware of its current shortcomings in terms of preservation.

5 From the very short term to the long term

If the preservation time required is specific to each type of document type and its planned use, we can at least try to define a timescale for the preservation of digital documents:

- very short term: corresponds to the technical and commercial life expectancy of the consultation chain; in the case of a problem performance is assured by a maintenance service;
- short term: physical life expectancy, it works for as long as it works ...;
- medium term: access to data is assured thanks to the implementation of an organisation responsible for migrations or other operations;
- long term: from the moment at which this organisation no longer exists, we return to the physical life expectancy of the consultation chain resulting from the latest migration.

Once we are aware that digital data is not autonomous in terms of preservation and that it must be managed (hyperactive storage), it is possible to guarantee its medium term sustainability. This requires a reliable organisation and considerable human and financial resources. Such an organisation must ensure data collection and preservation and control data access. Of course there is always a possibility of loss, or of migration being forgotten — no organisation is perfect. But this does not give any indication of the possible duration of this medium term, or resolve the problem in the long term. Indeed, while medium term preservation is realistic, long term archiving poses a real problem. What will remain for the historians?

The remainder of this paper is focused on the software chain.

6 From source formats to viewable formats

Starting with a file, intermediaries are required, notably a software chain, so that a document can then be read.

The following are some of the different format types used for text documents:

- source format (LaTeX, RTF, TEI, HTML);
- viewable vector format (PostScript, PDF);
- viewable bitmap format.

And here are the possible stages involved in converting a source file into information that is comprehensible to a human being:

$$\text{computer>source file} \xrightarrow[\text{ms word}]{\text{pdftex}} \text{vector file}$$

$$\text{vector file} \xrightarrow[\text{acrobat}]{\text{xpdf}} \text{bitmap data>peripheral>human}$$

A source file will contain text and information expressed using a given language: typographic information (justification, italics, etc.), information on the structure (title, author, chapter, etc.), or even a mixture of the two. This is the type of file created by an author and therefore oriented towards document creation.

Once processed by tools such as MS Word or pdfLaTeX, the output is a viewable vector file (type PostScript or PDF). This is a file containing text and positioning information: put the letter x on the page in a specified position and use the Times font. It is a geometric description of the page. Draw me a circle of radius 1 in the middle of an A4 page. Of course, the equations involved in drawing a letter are more complicated than for a circle, but the principle is the same.

Tools such as Xpdf, Ghostscript, Acrobat and the PostScript interpreter on a printer are examples of software that can read these PostScript or PDF files and pass on this geometric description to an image on a screen or printer — i.e. a bitmap image identical to an image produced by a scanner. A viewable bitmap file which is "understood" by the peripherals: the pixel is on or off, ink must be deposited at such and such a point on the page. The file has been converted to an image that can be seen by a human.

With WYSIWYG tools such as MS Word or OpenOffice, these stages are invisible and hybridized, since all these functions are performed in the same software.

In software terms these stages have varying levels of complexity.

Displaying a bitmap file onscreen requires simple software. No arbitration is required between the content of the file and the final result: with a 0 the pixel stays black, with a 1 it becomes white. All the complex formatting work has already been done. But there is one disadvantage: the text, in the form of code ASCII, ISO-8859, etc.) has disappeared and the files are voluminous.

As for software applications processing vector data, these are of intermediate complexity. They start, for example, with a PDF file to produce bitmap data and send it to the screen. Since a vector format is a mathematical description of a page, in software terms the freedom of interpretation is limited: tracing a circle on a page is not ambiguous. But it is greater than with a bitmap file: pixelization must be processed, the designers have been able to leave some grey areas in the specifications. Vector formats are often complex, as can be seen simply by consulting the description of PostScript or PDF format, and programming errors or difficulties are more frequent. In terms of advantages the text is often present, the file takes up less space than a bitmap file, and zooming is possible. Here too, the formatting has already been done by another application.

A software application that can process a source file is far more complex. Its aim is to obtain a page description in PDF or another format. With these applications the link between the file content and the final result is the most sustained, the artistic component is greater and the programmer has more freedom. In the final analysis a source file contains very little information in terms of the expected result; it is the task of the software to make the difference.

The problem arises from the fact that for these formats very high level commands are used. What does a tag `<h1>` signify? The link between a tag and the graphic output is completely arbitrary. No two browsers give the same result. This can be seen using browsers with complex formatting, tables, CSS, etc. What does a "justify" type command signify? Behind this command there must be software to implement an algorithm — varying in effectiveness — to perform paragraph justification and hyphenation. And the same RTF file (or another source format) read by OpenOffice and by MS Word will not give the same document, since the justification algorithm is not the same. Most of the work of formatting and graphics is done by the software and not by the source file, which contains only text and commands.

So, in short, there are three different format types and three types of software of varying degrees of complexity. Files in source format are at the start of the software chain and therefore require the most complex processing. Bitmap files are located at the end of this chain. Therefore, from the point of view of file preservation, the bitmap format is of interest, since the software chain is limited to a single software application. Moreover this software is simple, easy to write or to rewrite. Of course

in relation to a file in PDF format many possible functions are lost (hypertext links, zoom, full-text search, etc.). But that is the price to be paid: by moving closer to human beings we move away from the machine with all its potential. And while it is possible to print a digital text document, it should be borne in mind that the existence of an analogue version is not always possible or even necessary with digital data (databases, for example). In terms of reaching a compromise between software complexity and functionality, PostScript and PDF appear to be interesting formats.

In general terms, a reasonable hypothesis is to say that a digital document has a life expectancy as long as the software required for its access is simple and the software chain short. The same applies to hardware intermediaries.

7 Structure and typography

There is a current trend to limit a text document to its structure. This can be viewed as the disappearance of a typographic culture and is often associated with the development of XML. However in areas such as physics, mathematics or computer science, authors may still pay great attention to formatting and to compliance with typographical rules. These are the main areas in which LaTeX is used. Can a document be limited to its text and its structure expressed in command form? A document is both content and form, in other words it is also graphic, and respecting an author's work means respecting both these aspects. In the same way, what the reader wants is a legible document, not a file. Any typography exists to improve legibility. As for the quality of a document, this plays an important role in the potential pleasure of reading. Concentrating on the structural aspect often makes us forget about graphic aspects, and with them the importance of the software applications controlling the graphics.

Of course, from the point of view of preservation, it is legitimate to ask ourselves questions on the input of each of these components. What needs to be preserved?

In fact the problem arises from the fact that in terms of preservation text documents have a polymorphous aspect:

$$\text{document} = \text{text} + \text{structure} + \text{images}$$

A text document consists of text represented using coding (ASCII, ISO-8859, Unicode) in a source file and in graphic form in a viewable file. It also consists of the images included in the document. It is a structure that may be expressed in source files in the form of commands associated with the text or directly in graphic form in a viewable document thanks to formatting and typography.

$$\text{text} + \text{structure} + \text{images} \xrightarrow{\text{software}} \text{graphics}$$

$$\text{graphics} \xrightarrow{\text{reader}} \text{text} + \text{structure} + \text{images}$$

Take the example of the well-known RFC (*Request For Comment*) documents describing the Internet standards. Since their creation in 1969 they have been available in the form of simple ASCII files on the site `http://www.rfc-editor.org`. Of course, with ASCII the graphics are very simple, without any multilingualism, maths, molecules, music, etc. ... but it works, and that is sufficient since the Internet exists.

In fact, if typography is of no concern and the document does not have any graphic components, a text file is sufficient for saving information. It can be read with text editor software, which is simpler than a word-processing application.

Note that tools are available to extract text from a LaTeX, RTF or PDF file (such as detex, rtf2text and pdftotext). These tools are also used by the full-text indexers of search engines to process non-HTML files.

Conversely, if the graphic component is important or we want to preserve the author's formatting from a source file, a complex software chain becomes necessary to access the document as it was created by the author. The software used has to be compatible with that used by the author, preferably the same application and the same version.

Generally speaking the question arises: is it essential to guarantee the absolute integrity of a digital recording or can it be modified along with its consultation chain, preserving only the aspects considered essential?

8 XML — a lasting format, and the software?

What signifies that the XML format is lasting, and is that really the case?

With a format such as HTML the tags are defined once and for all by an international consortium, W3C, and are not extensible to suit the user's requirements. XML (*eXtensible Markup Language*), on the other hand, allows users to create their own tag sets. OpenOffice XML, TEI and XHTML are examples of tag sets in the document domain. It is therefore possible to create XML tag sets describing the structure of a document (such as TEI, *Text Encoding Initiative*) as well as tags representing typography or formatting (such as XHTML, OpenOffice XML or XSL-FO). XML defines rules which must be complied with when users

create their own tags (every start-tag must have an associated end-tag, no intermingled tags, etc.). For each tag set (whose characteristics are defined by a DTD — *Document Type Definition* — or an XML schema), a software application to process these tags must be associated.

Therefore there is not really one XML format, but rather formats complying with the rules defined by XML; each one will then use a particular DTD. In fact it is an over-simplification of language to use the term "XML format".

That said, what signifies that XML is lasting? If it is a matter of saying that the *Extensible Markup Language* (XML) 1.0 W3C recommendation is lasting, why not: the rules that must be complied with when creating tags to comply with version 1.0 of the XML recommendation are lasting.

If however that means that whatever the DTD used, from the point at which the format complies with XML, the information is lasting, then that is debatable. Effectively, as we have already seen, between the file and a result accessible to the user there is always a software application. And there is nothing to guarantee the lasting nature of the software.

One format based on XML may be no more or no less lasting than another XML format. In fact we cannot really talk of a lasting format, it is the software which is or is not lasting. And software rarely is.

In 10 years will we be able to read a document using the XML format of OpenOffice or TEI? Yes, if software that can process this format exists — in fact the problem is the same as for RTF or LaTeX. It is true however that knowing the specifications of a format is a plus, and is even necessary. For example, the specifications of the MS Word format are not known. The potential reader is therefore a prisoner of the entity that knows the format and can write the corresponding software. That said, a format specification cannot replace available software, whether because of specifications that may be incomplete and the new software not fully compatible with the original, or because the financial resources required for its redevelopment are excessive. Moreover, nothing obliges the creator of an XML tag set to publish it. Or the DTD/XML schema may be inexact in relation to the available software applications that are supposed to process it.

It is true that XML has a certain advantage, if we comply with the substance of the recommendation, which is that it is not a binary format. The recommendation states that XML documents should be "human-legible and reasonably clear". The obligation to use standard coding such as Unicode for the text is important.

That said, in terms of preservation XML is one formalism among others, even if the aims of its designers are laudable and it is rather well placed in relation to the competition. The problem, where things are really complex, is at software level, and it is not realistic to believe that all the problems of sustainability can be solved by a matter of formalism.

The use of XML is often associated, in a very positive way, with discovery of the possible separation between structure and presentation, and of the problems posed by proprietary formats. But due to an excess of evangelization, we can be led to forget that a file, even based on XML, still needs a hardware and software chain in order to be consulted.

We could even add that this freedom for users to create their own tags while hiding behind the protection of XML compliance can give rise to problems. For example, in the domain of sound and musical scores, music and XML, there are currently no less than 18 different markup proposals: MusicXML, MusiXML, MusicML, etc. (For example, see a list on `http://xml.coverpages.org/xmlMusic.html`.) From Esperanto to the Tower of Babel ...

9 From LaTeX to HTML

Format conversion often gives rise to the same problems. Since there are tools in existence for conversion from LaTeX to HTML (HeVeA, TeX4ht, Tralics, etc.), we take a look at the situation. Here is a simplified description of how these tools work.

In fact there are two cases: text, which is easy to process, and the rest. Either there is a correspondence between firstly a LaTeX command and secondly an HTML command that a browser can display (title, bold) and the software performs the conversion, or there is no correspondence. In this case the conversion program runs LaTeX to generate a GIF or PNG image which will be inserted into the HTML document in the form of a link.

Some years ago mathematical equations were converted in the form of images, but the more recent tools now convert them into MathML as browsers are beginning to support this format. But extensions of LaTeX also represent chemical molecules or music scores. And in these cases converters to HTML still generate images.

This type of conversion poses two problems. The first arises from the wealth of LaTeX functions compared with the lesser capacities of the HTML format. Structured information is often converted to images. Secondly, the formatting of PostScript or PDF files generated by LaTeX is known for its quality, while for HTML it depends on the browser used.

LaTeX does not use a binary source format, which is good. But the advantage of LaTeX is not its format. Effectively there is not much difference between

```
\title{My title}
```

and

```
<title>My title</title>
```

The advantages of LaTeX lie in the many features of the software. And changing format also means changing software.

10 Software and developers

PostScript and PDF formats are widely used, not without reason. However there are in fact very few software applications available for consulting these files. The best known are:

- for PostScript: the Adobe interpreter, Ghostscript;
- for PDF: Acrobat, Ghostscript and Xpdf.

And few people are involved in developing these applications. For example, Ghostscript employs 16 people, and Xpdf one person (with some twenty contributors).

The operational knowledge is concentrated in a very small number of people — not because they want to maintain a monopoly but because no one else is really interested. There is a pyramid effect: many users faced with few software choices and few developers. At present this is not a problem; these software applications exist, they work and are maintained, but in the long term this could become a real problem.

11 To conclude

Long term preservation of digital documents is an ongoing problem. More time is certainly required to stand back and assess the situation. And while formats for which the specifications are not known pose a problem, resolving questions of formalism will not solve all the problems. It is important to remember the importance of hardware and software aspects in respect of preservation of digital documents.

Today we must not think too much in terms of everlasting formats. The formats are not everlasting, neither is the software, still less the hardware. But we must think in terms of file migration and hardware migration, as well as conversion of formats. The difficulties involved in preservation are intrinsic to the same technique that allows such marvels in terms of creation or distribution.

What does the future hold? In fact it is very difficult to guess at possible developments or miracles. Some aspects that appear worrying to us at present will no longer be so in the future, not because the problems have been resolved, but because they have simply ceased to be problems — or because workarounds have been developed.

Note: Within the framework of the movement for open access to knowledge, all the documents listed in the bibliography can be consulted freely on the Web.

References

[1] Inge Alberts. Préservation de l'information numérique, 2003. http://www.esi.umontreal.ca/~albertsi/INU1030/.

[2] Dutch National Archive. *The Virtual Library of the Digital Preservation Testbed*. http://www.digitaleduurzaamheid.nl/index.cfm?paginakeuze=185&lang=en.

[3] ATICA. *Guide pour la conservation des informations et des documents numériques*, 2002. http://web.archive.org/web/20040921132039/http://www.adae.gouv.fr/spip/article.php3?id_article=7.

[4] Michel Auffret. L'archivage pérenne des documents numériques. In *JRES Marseille*, 2005. http://2005.jres.org/paper/47.pdf.

[5] Marie-Anne Chabin. Document trace et document source. La technologie numérique change-t-elle la notion de document ? *Revue I3*, 4, 2004. http://www.revue-i3.org/volume04/numero01/revue_i3_04_01_09.pdf.

[6] Archives de France. Bulletin des archives de france sur l'archivage à long terme des documents électroniques. http://www.archivesdefrance.culture.gouv.fr/gerer/archives-electroniques/bulletins-archives-electroniques.

[7] Groupe de travail PIN : Pérennisation des informations numériques. http://www.aristote.asso.fr/PIN/.

[8] Association des Archivistes Suisses. *Stratégie globale pour la conservation à long terme des documents électroniques en Suisse*, 2002. http://www.vsa-aas.org/fr/aktivitaet/directeurs-des-archives/strategiestudie/.

[9] Catherine Dhérent. *Les archives électroniques. Manuel pratique*, 2002. http://www.archivesdefrance.culture.gouv.fr/static/1062.

[10] Claude Huc. La pérennité des documents électroniques points de vue alarmistes or réalistes ? *Bulletin des Archives de*

France sur l'archivage à long terme des documents électroniques, 7, October 2001. http://www.archivesdefrance.culture.gouv.fr/static/1671.

[11] Roger Pédauque. Document: Form, sign and medium, as reformulated for electronic documents. *@rchiveSIC*, 2003. http://archivesic.ccsd.cnrs.fr/sic_00000594.

[12] Roger Pédauque. Document: forme, signe et médium, les reformulations du numérique. *@rchiveSIC*, 2003. http://archivesic.ccsd.cnrs.fr/sic_00000511.

[13] Jean-Luc Philip. Le point de vue d'un généalogiste sur la conservation des documents électroniques. *Bulletin des Archives de France sur l'archivage à long terme des documents électroniques*, 6, July 2001. http://www.archivesdefrance.culture.gouv.fr/static/1670.

[14] Christian Rossi. De la diffusion à la conservation des documents numériques. *@rchiveSIC*, 2005. http://archivesic.ccsd.cnrs.fr/sic_00001379.

[15] Christian Rossi. De la diffusion à la conservation des documents numériques. *Cahiers GUTenberg*, 49, 2007. http://www.gutenberg.eu.org/publications/cahiers/r46-cahiers49/222-rossi.html.

[16] Chris Rusbridge. Excuse me... some digital preservation fallacies? *Ariadne*, 46, 2006. http://www.ariadne.ac.uk/issue46/rusbridge/.

◇ Christian Rossi
SEISM/DSI
INRIA Grenoble – Rhône-Alpes
655 avenue de l'Europe
38334 Saint Ismier Cedex
France
christian dot rossi (at)
inrialpes dot fr

Rich media annotations and AcroFLeX

D. P. Story

Abstract

The *Adobe Supplement to the ISO 32000* document introduces a new annotation type, the rich media annotation. This paper describes rmannot, a new LaTeX package that implements this new annotation. Additionally, the acroflex package, a major application of rmannot, is also discussed with some technical details.

Work on these two packages followed the time the author spent working for Adobe Systems in 2008 as part of the Acrobat 9 development team.

1 Introduction

A new and exciting feature of Acrobat 9 and Adobe Reader 9 is that an Adobe Flash player is embedded in the application's executable code; consequently, version 9 offers native support for Flash video (FLV), Flash applications (SWF), and the digital audio encoding format MP3. This enables a reliable cross-platform playback experience for the user. The user accesses the Flash player, which plays the media, through the new *rich media annotation*.

The Acrobat 9 Pro user interface allows for the creation of a rich media annotation, the specification of the media to be either embedded or streamed from the Internet, the selection of a skin from a collection of predefined skins to control the media, and so on. The rmannot package implements the rich media annotation, as specified in Chapter 9.6 in the extensions document [1], and all the user interface features of the rich media annotation.

One of the motivations for writing the rmannot package was the desire to write an interactive graphing system for PDF. The acroflex package creates the AcroFLeX Graphing System, and represents a major application of the rmannot package. To this end, the AcroFLeX graphing widget, an SWF file, was created using Adobe FLEX 3. The AcroFLeX graphing widget is embedded in the document using the rmannot package, and it, as its name implies, performs and displays the graphing.

Another important part of the Acrobat/Flash connection is to establish a communication link between the two. The scripting language of Acrobat is JavaScript, while the Flash player uses ActionScript. These two scripting languages can communicate using the *scripting bridge*, created for Acrobat for this purpose. Details of the scripting bridge, and how it is used, are presented in section 3 on the acroflex package.

2 The rmannot package

The *PDF Specification, Sixth Edition, version 1.7*, see [3], has been made into an international standard, called ISO 32000 (ISO 32000-1), see [4]. The PDF specification, as documented in [4], is no longer under control of Adobe, consequently, Adobe now publishes an extensions document [1] that specifies new PDF language features that are not in ISO 32000. The resource document for rmannot and acroflex is the *Adobe Supplement to the ISO 32000* [1].

2.1 Multimedia for Versions 6 and 9 compared

The rich media annotation, as specified in [1] turned out to be very straightforward to implement. The new annotation is much easier than the complex multimedia approach of version 6. The multimedia of version 6 uses what the *PDF Specification* [3] calls a *screen annotation*, and has been implemented for LaTeX in the movie15 package [5] by Alexander Grahn.

The multimedia approach of version 6, which supports a long list of media types, requires the underlying operating system to locate and launch an appropriate multimedia player residing on the user's system to play the media clip. In rich media annotation of version 9, only SWF, FLV, and MP3 files are supported (other media types can be converted to one of these using Acrobat Pro Extended, or a third party conversion utility), but the embedded Flash player is used to play the media, so no external multimedia player is used or needed.

2.2 Implementation notes for rmannot

As it is now written, rmannot requires the document author to use Acrobat 9 Pro to create a PostScript file, perhaps using dvips, and to distill it using Acrobat Distiller 9. Once the document is built into a PDF, it can be viewed with Adobe Reader 9 (or later). Currently, this package is not available to users of pdftex.

The rmannot package is part of the AeB Pro family of packages (see [7]), this family is a collection of packages that require the use of Acrobat Distiller to create PDF. The package, documentation, and demo files can be obtained from the home page of rmannot.[1]

2.2.1 Embedding media

To embed media (SWF, FLV, and MP3 files) in the document, rmannot uses another package in the AeB

[1] The rmannot home page is at `http://www.math.uakron.edu/~dpstory/rmannot.html`.

Pro family, called graphicxsp (see [7]). The rmannot package defines a command \saveNamedPath that is to be executed in the preamble of the document; \saveNamedPath uses graphicxsp commands. The syntax is as follows:

\saveNamedPath{⟨*name*⟩}{⟨*path*⟩}

The ⟨*path*⟩ is the absolute path to the media file (with extension .swf, .flv, or .mp3 included). The absolute path is required because Distiller does not work with relative paths and does not have a notion of current directory. The ⟨*name*⟩ is a symbolic name that is used to reference this media throughout the document.

2.3 Creating a rich media annotation

Once the media file has been embedded, in the body of the document, rich media annotations can be created using the \rmAnnot command:

\rmAnnot[⟨*options*⟩]{⟨*width*⟩}{⟨*height*⟩}{⟨*name*⟩}

The ⟨*options*⟩ are key-value pairs that allow the document author to specify any of the rich media options available through the Acrobat user interface. The ⟨*width*⟩ and ⟨*height*⟩ are the width and height, specified in any of the scales of measurement LaTeX supports. The annotation may be resized, using, for example, the \resizebox command of the graphicx package, to any size while maintaining the aspect ratio of the media clip. Finally, the ⟨*name*⟩ is the name given the media by a \saveNamedPath command. This example,

```
\rmAnnot[poster=myPoster,
    skin=skin3]{640bp}{480bp}{myFLV}
```

shows some of the many optional key-value pairs available; here we specify a poster for the media (the appearance of the media when the annotation is not activated), and the skin to use to control this Flash video.

The rmannot package embeds a media file only once, but that media can be displayed and played on multiple pages without significantly increasing file size. Acrobat, by contrast, through its user interface, will embed the same media multiple times, once for each rich media annotation that uses that media file. In this regard, the approach taken to embedding by rmannot is superior to that of Acrobat.

There are other economies that should be mentioned as well. The embedded Flash player handles SWF files natively, but requires an SWF application to play video and sound files. FLV and MP3 files are actually played by VideoPlayer.swf and AudioPlayer.swf, respectively. These two SWFs are shipped with Acrobat Pro (and Extended). In addition to these, there are seven skin files (also SWF files) that are shipped with Acrobat that provide control over Flash video. The rmannot package takes care to embed each of these, as needed, only once; while Acrobat embeds the players and skins multiple times. This saving of file size becomes very important with the acroflex package, where the document author may want many graphing screens throughout the document.

It should be noted that the SWF players and skins that come with Acrobat are not distributed with the rmannot package, which would violate the licensing with Adobe; rather, rmannot requires the document author to have Acrobat Pro, so these files are already on his system. The players and skins are, by the way, one of the problems with porting rmannot to pdftex. The SWFs cannot be redistributed, so a package developer must write his own players and skin SWFs, and provide them in the package distribution.

3 The acroflex package

The acroflex package is a major application of the rmannot package. The keys to this package are the AcroFLeX graphing widget, which was written using Adobe FLEX 3, and the scripting bridge that allows communication between Acrobat (or Adobe Reader) and the graphing widget.

FLEX 3 — available from Adobe without charge to educators and students — is an XML-like (MXML) markup language that is compiled into an SWF file.[2] FLEX has charting (graphing) capabilities that are exploited by the acroflex package.

The acroflex package is part of the AeB Pro family of packages (see [7]). The package, documentation and demo files can be obtained from the home page of acroflex.[3]

3.1 Features of AcroFLeX

The AcroFLeX graphing screen (the visual appearance of the AcroFLeX graphing widget) can be interactive or non-interactive. The document author can create as many graphing screens as needed; each screen can appear in the rich media annotation at a fixed position in the document, or in a floating window.

For an interactive graphing screen, the user can enter an expression (into an Acrobat form text field) representing a function of a single variable x, a polar function of t, or a set of parametric equations that are functions of t. Various controls are provided to

[2] Adobe FLEX 3 can be found at http://www.adobe.com/.

[3] The acroflex home page is at http://www.math.uakron.edu/~dpstory/acroflex.html.

change the viewing window, for shifting horizontally and vertically, and for zooming in or out.

The author can also pre-populate the fully interactive screen by creating one or more links using the \sgraphLink command. When activated by clicking, the link passes graphing data to the graphing screen to be viewed by the user. The user may then interact with the graph.

For a non-interactive graphing screen, no controls are provided to manipulate the graph; the user can only view the graph. The screen is populated when the user clicks a link created by the command \sgraphLink, as described above. Information passed by the executing JavaScript of the link to the graphing routines of AcroFLeX includes the functional expression (or list of points to plot), domain, and range.

In the current version of AcroFLeX , up to four functions can be graphed, four functions with the shaded regions between the graph and horizontal axis can be graphed, and four sets of plotted points can be displayed, all on one graphing screen.

3.2 Implementation notes

The two major challenges of this package were to write the graphing widget using FLEX 3, and to write document-level JavaScript to calculate plot data to be passed to the graphing widget. The plot data is passed using the scripting bridge.

Development of the AcroFLeX graphing widget evolved over time as my understanding of Adobe FLEX grew. The rmannot package is then used to embed the widget in the PDF document, and to display the widget as the graphing screen through a rich media annotation.

On the Acrobat side, the JavaScript functions Graph_xy and Graph_xyt were written to prepare plot data for functions of a single variable, and for polar functions and parametric equations, respectively. These functions get graphing data supplied by the user (in the case of an interactive graphing screen), or receive graphing data as part of their parameters (in the case of using \sgraphLink to pre-populate a graphing screen). The exerquiz package (part of AeB, see [6]) is used to parse the target function, and the graphing data then create the plot data.

The plot data created by the graphing functions is built as an XMList. For example, suppose the function is x^2, to be plotted over the interval $[-2, 2]$, with $n = 5$ data points, the plot data has the following form:

```
cPlotData=<points>
    <point><x>-2</x><y>4</y></point>
    <point><x>-1</x><y>1</y></point>
    <point><x>0</x><y>0</y></point>
    <point><x>1</x><y>1</y></point>
    <point><x>2</x><y>4</y></point>
</points>
```

Once this XMList has been constructed, it is converted to a string,

```
cPlotData=cPlotData.toXMLString();
```

We then get the rich media annotation object,

```
var annot = this.getAnnotRichMedia(pNum,
    "afRM"+baseName);
```

and send the data to the AcroFLeX graphing widget by way of the scripting bridge,

```
annot.callAS("getPlotData", graph_props, oDR,
    cPlotData);
```

The bridge from Acrobat-to-Flash is the JavaScript callAS method of the rich media annotation. (See [2] for information on the callAS method). The function getPlotData is an ActionScript function defined in the AcroFLeX graphing widget. We pass the function name, "getPlotData", as a string; the properties of the plot graph_props (a JavaScript object); the range and domain specification oDR (a JavaScript object); and finally, the plot data itself, cPlotData.

In order for getPlotData ActionScript function to be recognized by the widget, it must be exposed by the ExternalInterface.addCallback method of ActionScript. Within the source code (MXML) of the AcroFLeX graphing widget, we have

```
private function initApp():void {
      ExternalInterface.addCallback(
        "getPlotData",getPlotData);
};
```

The initApp function is executed when the graphing widget is activated.

The function getPlotData, on the AcroFLeX widget side, interprets the graph_props object to determine what type of plot is to be created (plot points or draw a curve, draw continuous curve or segmented curve, shade graph or not). AcroFLeX maintains an array of length 12 to manage all the curves and plotted points. The function passes the viewing domain and range, the oDR object, to the chart for labeling, and assigns a unique color for the curve or plot. Finally, getPlotData converts the parameter cPlotData to XML,

```
var xmlPlotData:XML = new XML(cPlotData);
```

and populates the chart with the plot data.

3.3 Compatibility with exerquiz

The exerquiz package plays an important role in the acroflex package by providing the parsing routines

for algebraic expressions; however, exerquiz was created to provide quizzing environments for educators. The AcroFLeX graphing screens can be integrated into exerquiz quizzes using support commands provided by the acroflex package. The demo file `afgraph.pdf` contains several examples.

The interested reader is encouraged to view the demo file `afgraph.pdf`, located at the home page of the `acroflex` package.[4]

4 The AcroTEX PDF blog

Following the development of rmannot and acroflex, the *AcroTEX PDF Blog*[5] was created for the PDF and LaTeX communities; the blog covers various topics in PDF, including extensive information on the rich media annotation (RMA) and on the scripting bridge. The reader interested in these topics may read PDF blogs #1–11; these blog articles were written using the rmannot package.

5 Concluding remarks on LaTeX

The rmannot and acroflex packages forge no new ground in LaTeX code; they build on the AeB and AeB Pro bundles. Developing a package has become easier through the years because there are so many basic packages that are available for developers: hyperref, xkeyval, and xcolor, to name a few. In any case, LaTeX is shown to be a fine authoring system for new applications of emerging technologies.

[4] `http://www.math.uakron.edu/~dpstory/acroflex.html`

[5] `http://www.math.uakron.edu/~dpstory/pdfblog.html`

References

[1] Adobe Systems, Inc. *Adobe Supplement to the ISO 32000, BaseLevel 1.7, ExtensionLevel 3*. `http://www.adobe.com/go/pdf_developer`, 2008.

[2] Adobe Systems, Inc. *JavaScript for Acrobat API Reference*. `http://livedocs.adobe.com/acrobat_sdk/9/Acrobat9_HTMLHelp/index.html`, 2008.

[3] Adobe Systems, Inc. *PDF Reference, Sixth Edition, version 1.7, Adobe Portable Document Format*. `http://www.adobe.com/go/pdf_developer`, November 2006.

[4] International Organization for Standardization. *ISO 32000-1:2008, Document management — Portable document format — Part 1: PDF 1.7*. `http://www.iso.com`, 2008.

[5] Alexander Grahn. The movie15 package. Available from CTAN, `macros/latex/contrib/movie15`, 2008.

[6] D. P. Story. AcroTEX eDucation Bundle (AeB). Available from CTAN, `latex/contrib/acrotex`, 2008.

[7] D. P. Story. AeB Pro Family of Software. Available from CTAN, `latex/contrib/aeb_pro`, 2008.

◇ D. P. Story
Department of Mathematics
Northwest Florida State College
Niceville, FL 32578
U.S.A.
`dpstory (at) acrotex dot net`
`http://www.math.uakron.edu/~dpstory/`

Use of the \write18 feature for composing indexes

Claudio Beccari

1 Introduction

Any LaTeX user who has composed an index knows the trouble one gets in when s/he has spent a lot of time collecting the index entries, with the correct formatting, and eventually finishes his/her paper but forgets the last three runs in order to have the correct final index typeset at the end of the paper. The three runs are in order:

1. Run LaTeX or pdfLaTeX with the command `\makeindex` in the preamble; this will collect the index entries, their formatting, and the page numbers where the entries are located in the document; this information gets written down in a file with extension `.idx` that contains all the entries with the related information in the order they were collected.
2. Run the program `makeindex`, possibly with a personalized style file and specific options in order to produce the sorted entry list with their ranking and page numbers. This information gets written into a file with extension `.ind`; this file contains the whole `theindex` environment with each entry marked with an `\item` or `\subitem` or `\subsubitem` prefix, together with the relevant formatting instructions for both the entries and the page numbers.
3. Run again LaTeX or pdfLaTeX to process the final index file, possibly by inputting it by means of a command such as `\input{`⟨*filename*⟩`.ind}`.

These three steps can be executed in just one run if you make use of the command `\write18{`⟨*text*⟩`}`.

Modern TeX distributions contain an executable file and a format file that have been created with special switches in such a manner that the interpreter `pdftex` may produce either a `.dvi` or a `.pdf` output file; it may make use of the commands of the extended `etex` interpreter, or refrain from doing so; it may or may not "shell out" to the system and ask it to perform some actions before resuming its main typesetting task.

Here I will exploit this last feature in order to produce the actual index in one run of (pdf)LaTeX.

2 Auxiliary files

When the command `\makeindex` is executed while the preamble gets processed, a new output stream is opened in order to write the index information into the `.idx` auxiliary file; at the same time, that command activates the various `\index` commands within the document that actually write the information into the auxiliary file. The name of this output stream is assigned to the internal LaTeX kernel control sequence `\@indexfile`.[1]

This file has the specified extension `.idx` and its name is the same as that of the (main) `.tex` file that is being typeset; the name of this file is kept into the TeX control sequence `\jobname`, that may be used by any suitable macro.

Since the index is typically the last thing to be typeset in a document, `\input{`⟨*filename*⟩`.ind}` is likely the last command before the `\end{document}` statement.

So, in order to sort and produce `\jobname.ind` from `\jobname.idx`, we have to do essentially two simple operations:

1. Close the output stream identified by the internal macro `\@indexfile`, and
2. execute the `makeindex` command in order to produce the new updated `.ind` file.

3 The commands

We have to execute "immediately" the following two commands:

```
\immediate\closeout\@indexfile
\immediate\write18{
    makeindex -s style.ist \jobname.idx}
```

just before the `\input{\jobname.ind}` statement or its equivalent. Of course, if no specific index style file `style.ist` is being used, the whole clause `-s style.ist` is omitted.

The `\immediate` TeX primitive instructs the interpreter not to delay the execution of the subsequent command; without it, the TeX commands concerned with input and output streams, including the actual read or write commands, are deferred until a page is shipped out. Obviously `\closeout` closes an output stream, in our case the one specificied by the subsequent macro `\@indexfile`.

The `\write18{`⟨*text*⟩`}` command writes onto the special output stream numbered "18"; this stream is equivalent to writing the ⟨*text*⟩ as a system command line in the command window of the Windows family

Editor's note: Reprinted by permission from ArsTEXnica #8, October 2009.

[1] This is what the LaTeX kernel and the standard classes do. Some nonstandard classes may use other control sequences to hold the information about the index output stream: for example the class memoir by default uses the control sequence `\jobname` extended with the suffix `@idxfile`, but, since memoir may build up several indexes, each one has its own output stream with a different name, that gets specified with the optional argument of the command `\makeindex`.

of operating systems, of an xterm window of the Unix family of operating systems, or of the terminal window of the Mac OS X operating system. When the system command finishes its processing, control returns to the TEX interpreter that goes on from the very point it had suspended its processing in order to have the operating system perform the required task.

The presence of the `@` character in the output index stream name '`\@indexfile`' requires that a `\makeatletter` command be issued before the first statement and a `\makeatother` be issued just after the first statement, thus changing and reverting its category code.

On the other hand, if those two statements are defined with a macro contained in a personal `.sty` file, there is no need to change the category code of `@`. This would also be a good occasion to add useful bells and whistles to the new command, for example, checking if the output stream really exists and may be closed; checking that `\jobname.idx` exists in order to convert it; provide for an optional argument to specify the index style file so as to insert the style file clause in the system call; and warning and error messages could be added so as to let the author know if anything is inconsistent. This is not the proper place to add long stretches of TEX or LATEX code, but any reader should be capable of customizing a new command suitable for his/her needs.

4 The results

By issuing those commands, or the new command defined in the personal `.sty` file, just before giving the command `\input{\jobname.ind}` (or equivalent), the index turns out to contain exacly the index entries and page numbers that were collected during that current (pdf)LATEX run, and there is no need to remember to update the index file after any correction to the source file(s), since the entries and the page numbers are always up to date.

I find these simple macros extremely valuable, and I use them myself in all my (recent) documents, after I realized the possible benefits deriving from the `\write18` command. This facility is not widely used, because it makes it possibile for TEX to execute external applications exposing the typesetter to the risk of executing malicious code; therefore some TEX distributions are distributed with this facility disabled by default.

Nevertheless there is some good news in this regard: TEX Live 2009, and a subsequent release of MiKTEX, will contain a restricted version of the `\write18` command, such that `\write18` will be enabled by default, but only a restricted list of "safe" programs (listed in a configuration file) will be allowed to be executed. And, happily, `makeindex` is one of those safe programs.

I imagine there are many other possible applications of the `\write18` facility: for example, the format transformation of graphic files is already available in the package `epstopdf.sty`; it's up to the users to discover new ones.

◇ Claudio Beccari
`claudio dot beccari at gmail dot com`

Glisterings

Peter Wilson

> Yet in his feverish mind
> He still could find
> The miraging domes of Samarkand
> Glistering through the roiling sand.
>
> *Doubt*

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.

Corrections, suggestions, and contributions will always be welcome.

> There was an old man named Michael Finnegan
> Grew some whiskers on his chinnegan.
> The wind came out and blew them innegan.
> Poor old Michael Finnegan. Beginnagen.
> There was ...
>
> Traditional

1 Repetition

There are occasions when an author wants to repeat some text that occurred in a document at some later place, perhaps in an Appendix.

If the original is plain text then this is simple enough — define a macro holding the text and use it in each place the text is to appear. A variation on this is when there is some boilerplate text that will be used in many documents. If the text is short, then define a macro to hold it. If the text is longer, say a page or two, then put the text into a `tex` file and define a macro that `\input`s that file. I used both these schemes when I was writing class [5] and package [6] files for some ISO documents which tended to have much boilerplate text, both short and long.

Life gets more complicated if you need to repeat something that is automatically numbered by LaTeX, as requested by David Romano [4] in this partial quote from his posting to `texhax`, which was passed on to me by Barbara Beeton:

> *Here is the problem I'm trying to solve. I'd like to have a particular theorem appear twice in a paper, and I'd like its appearance to be identical in both, the numbering in particular, without having to set the section and theorem counters by hand.*

For the following examples I have specified:

```
\newtheorem{again}{Repetitive}[subsection]
\newsavebox{\Tsaved}
\newcounter{savesub}
\newcounter{savethm}
\newcounter{restoresub}
\newcounter{restorethm}
\newcommand*{\fox}{The slow old fox
 readily leaped over the quick brown dog.}
\newcommand*{\party}{All good men came
 to the party.}
\newcommand*{\boring}{Some things get
 terribly boring when repeated too often.}
```

1.1 Box it

Here is one theorem:

Repetitive 1.1.1 *All good men came to the party.*

and the next one is the one that is to be repeated. One way of repeating some typeset text is to save it in a box and then use that box wherever the text is to be repeated. The typeset theorem (1.1.2) below is produced by the following code:

```
\savebox{\Tsaved}{%
\begin{minipage}{\linewidth}
\begin{again}\label{th1}
\boring
\end{again}
\end{minipage}}
\vspace{\topsep}
\noindent\usebox{\Tsaved}
\vspace{\topsep}
```

Repetitive 1.1.2 *Some things get terribly boring when repeated too often.*

I discovered that I had to put a vertical space of `\topsep` before and after the box to match the normal spacing around theorems.

And we can use the box again to obtain a repetition of the theorem:

```
\vspace{\topsep}
\noindent\usebox{\Tsaved}
\vspace{\topsep}
```

Repetitive 1.1.2 *Some things get terribly boring when repeated too often.*

Using this technique a theorem has to be put into a minipage inside the box in order to get the correct line breaking, but minipages don't break across page boundaries. Provided the theorem is short and the stars are aligned then this won't be a problem. On the other hand, for long theorems and normal alignment the repetition will make a page too long.

Here is another theorem (1.1.3) to be repeated. This time, for use later, I save the values of the current `\subsection` and the current `again` theorem counters before typesetting the theorem:

```
\setcounter{savesub}{\value{subsection}}
\setcounter{savethm}{\value{again}}
\begin{again}\label{th2}
\party
\end{again}
```

Repetitive 1.1.3 *All good men came to the party.*

1.2 Saved the numbers

Any `again` theorem in this subsection will normally have a number starting **1.2**.

Having saved the relevant numbers for the first appearance of theorem 1.1.3 we can now typeset it again. The process is:

1. Save the current values of the `\subsection` and the `again` theorem counters.
2. Set the current values of those counters to those for the original `again` theorem.
3. Repeat the theorem.
4. Restore the values of the `\subsection` and the `again` theorem counters.

Here, then, is a repetition of theorem 1.1.3.

```
\setcounter{restoresub}{\value{subsection}}
\setcounter{restorethm}{\value{again}}
\setcounter{subsection}{\value{savesub}}
\setcounter{again}{\value{savethm}}
\begin{again}
\party
\end{again}
\setcounter{section}{\value{restoresub}}
\setcounter{again}{\value{restorethm}}
```

Repetitive 1.1.3 *All good men came to the party.*

Show another theorem here:

```
\begin{again}
This is a new theorem.
\end{again}
```

Repetitive 1.2.1 *This is a new theorem.*

And for the third time display theorem 1.1.2:

```
\vspace{\topsep}
\noindent\usebox{\Tsaved}
\vspace{\topsep}
```

Repetitive 1.1.2 *Some things get terribly boring when repeated too often.*

If the theorem to be repeated is complex then you could either define a macro for it, or put it into a file to be input.

```
\newcommand*{\foxy}{%
\begin{again}
\fox
\end{again}}
\foxy
```

Repetitive 1.2.2 *The slow old fox readily leaped over the quick brown dog.*

> A work that aspires, however humbly,
> to the condition of art should carry its
> justification in every line.
>
> *The Nigger of the Narcissus*,
> Joseph Conrad

2 Rectangular text

In an earlier column [7] I discussed how to create paragraph shapes of various kinds. One that I had not considered was sought after by Brad Cooper who asked on `comp.text.tex`:

> *I am trying to do something ... whereby two lines of large text are justified on the right and left without any hyphenation occurring.*

Several solutions were posted and I give some of them below, in alphabetic order of the respondents.

Donald Arseneau [1] said that the solution was to use a 'stretch' `tabular` column type, but that unfortunately there isn't one. Instead he suggested

```
\noindent\begin{tabular}{@{}r@{}}
\hfilneg A SHORT LINE \\
\hfilneg A LITTLE LONGER LINE \\
\hfilneg Donald Arseneau
\end{tabular}
```

which produces:

A SHORT LINE
A LITTLE LONGER LINE
Donald Arseneau

Or, using the array package, like this:

```
\newcolumntype{s}{>{\hfilneg}r}
\enskip\begin{tabular}{s}
A SHORT LINE \\
...
```

where I have used the `\enskip` space, as well as the regular paragraph indent, to set off the text from the left margin:

A SHORT LINE
A LITTLE LONGER LINE
Donald Arseneau

Enrico Gregorio [2] posted the following solution, defining a new `center`-like environment:

```
\newenvironment{stretchcenter}%
{$$\let\\\cr\vbox\bgroup\ialign\bgroup%
```

```
  \unskip##\unskip\cr}%
{\crcr\egroup\egroup$$}
```

The `stretchcenter` environment is used just like the regular `center` environment. The result is:

A SHORT LINE
A LITTLE LONGER LINE
Enrico Gregorio

Dan Luecking [3] posted further solutions. The first simply involves measuring the longest line and putting the others into boxes to match.

```
\newlength\gxx
\settowidth{\gxx}{A LITTLE LONGER LINE}
\noindent\makebox[\gxx][s]{A SHORT LINE}\par
\noindent\mbox{A LITTLE LONGER LINE}\par
\noindent\makebox[\gxx][s]{Dan Luecking}\par
```

This results in:

A SHORT LINE
A LITTLE LONGER LINE
Dan Luecking

That was the kind of method I had thought of but it does require some manual work. Dan also provided a more elegant solution to match the others, as:

```
\halign{#\cr
A SHORT LINE\cr
A LITTLE LONGER LINE\cr
Dan Luecking\cr}
```

which results in no indentation of the text from the left margin:

A SHORT LINE
A LITTLE LONGER LINE
Dan Luecking

To have the text indented, add space (`\quad` in the example below) inside the `\halign` like:

```
\halign{\quad#\cr
A SHORT ...
```

Note that the `\halign` and `\ialign` commands used by Dan and Enrico are usually hidden from LaTeX users but are used by the LaTeX kernel in defining environments like `tabular`, for example.

References

[1] Donald Arseneau. Re: Text filling the line. Post to `comp.text.tex` newsgroup, 24 March 2007.

[2] Enrico Gregorio. Re: Text filling the line. Post to `comp.text.tex` newsgroup, 22 March 2007.

[3] Dan Luecking. Re: Text filling the line. Post to `comp.text.tex` newsgroup, 23 March 2007.

[4] David Romano. Setting counters to output of a `\ref` command. Post to `texhax` mailing list, 8 February 2007.

[5] Peter Wilson. LaTeX for ISO Standards, 2002. `mirror.ctan.org/latex/macros/contrib/isostds/iso`.

[6] Peter Wilson. LaTeX Package Files for ISO 10303, 2002. `mirror.ctan.org/latex/macros/contrib/isostds/iso10303`.

[7] Peter Wilson. Glisterings. *TUGboat*, 28(2):229–232, 2007.

◇ Peter Wilson
18912 8th Ave. SW
Normandy Park, WA 98166
USA
herries dot press (at)
earthlink dot net

Peter Wilson's Herries Press packages

Will Robertson

Abstract

In September 2009 I became the maintainer of the majority of Peter Wilson's LaTeX packages. This short article describes how this came about and what the different packages are.

1 Who is Peter Wilson?

Frequent readers will be familiar with his name as an active member of the LaTeX community with ten *TUGboat* publications in the last eight years. He is most well known for his prodigious memoir class. He also gave the keynote address at TUG 2007.[1]

2 Introduction

Earlier this year I was writing a class file for a local conference and wrote to Peter about a minor feature request in his abstract package. He replied, quickly accepting the modifications I'd suggested, adding at the end 'Would you like to take over the package? I'm slowly retiring from LaTeX (age is calling) and trying to pass things off to others for support.'

Knowing of Peter's wide variety of packages on CTAN, it didn't make sense to me that each package should end up being maintained by whoever next sent a support email. Instead, I presumptuously offered to take on maintainership for the whole lot. After all, Peter is a well-regarded figure in the community and surely his packages don't receive very many bug reports? (As far as I know, this is indeed the case. Ask me again in a few months.) Peter seemed to like this idea and promptly sent me a complete list of his packages. I knew he was prolific, but I didn't quite realise the full extent of what I was taking on. Thirty-two packages ended up with my name in them, which involved a certain amount of tedium for both me updating the contact details for each and for the tireless CTAN administrators responsible for uploading the new versions.

In this report, I'll discuss what it means to be the 'maintainer' for a package and list the range and capabilities of Peter's packages.

3 Author versus maintainer

LaTeX itself and the majority of the third party contributed software for it are free and open source software, licensed under the 'LaTeX Project Public License'[2] (LPPL). The LPPL is similar to other well-known free software licences such as the BSD and Apache licences in that software may be freely distributed and modified, but access to the sources of derived versions is not required (unlike the 'copyleft' licences such as the GNU General Public License). The only restriction on redistribution is that the modified software must be clearly differentiable from the original by *users* of the code (not just people who read the source for fun). The easiest and most fail-safe way of doing this is simply by changing the name of the new package.

The LPPL also contains an interesting component that I have not seen in any other free software licences: the concept of an explicit 'maintainer' for the work who is theoretically responsible for keeping it up-to-date and for receiving bug reports. Usually the author of the software will be the maintainer of the work, but people change and move on and often lose interest in dealing with code they wrote long ago and no longer use. The LPPL formalises the process for new people to come along and adopt old code, especially 'orphaned' works for which the original authors can no longer be contacted.

This idea of explicit maintainership solves a real problem in the long term. In the LaTeX world, CTAN is the first port of call for contributed software; if it has not been uploaded there, it generally won't be available in TeX Live or MiKTeX. When package authors lose interest in their code and abandon their work, it is not clear how fixes or additions to their packages should be handled.

One can't simply upload patched versions of other people's code to CTAN, even if the original author is no longer around. If this were not the case, the CTAN team themselves would have to vet each new 'unofficial' update, in effect acting as *de facto* maintainers for all orphaned code — a preposterous idea considering the amount of work they already do, and certainly out of the question. Having explicit maintainers for the software they administer, the CTAN team can theoretically ensure that someone, somewhere, is responsible for each and every piece of software they (re-)distribute.

Peter himself was maintaining a number of packages for authors pre-dating his own involvement with LaTeX. This puts me in the dubious category of being a 'third generation LaTeX package maintainer'. (I take comfort in knowing that I'm not the only one.)

[1] "Between then and now — A meandering memoir", *TUGboat* 28:3, 2007, http://tug.org/TUGboat/Articles/tb28-3/tb90wilson.pdf, http://river-valley.tv/keynote-address-between-then-and-now-a-meandering-memoir/.

[2] http://www.latex-project.org/lppl/

4 The Herries Press packages

Peter's packages date from at least as far back as 1996 and fall into several rough groups:

- Replacements and better interfaces for functionality in the standard classes.
- Features to ease programming in LaTeX.
- New and assorted document features.

In the remainder of this section are brief descriptions of the packages I am now maintaining, concluding with a short list of Peter's works for which I am *not* responsible. Where other packages exist with similar functionality, I've listed them as well (to the best of my knowledge — no doubt I've forgotten some).

CTAN holds the definitive version of each package, of course. Rather than printing a link for each of these packages to their CTAN location, simply use this URL to access them:[3]

`http://tug.ctan.org/pkg/`⟨*package name*⟩

They are all included in recent (and not so recent) TeX distributions.

Development or pre-release versions of these packages are available at GitHub, where bugs and feature requests may be filed:

`http://wspr.github.com/herries-press/`

The adventurous may even wish to fork the code there in order to suggest code changes, which I will probably accept without too much question.

4.1 Standard class improvements

The standard LaTeX classes (article, book, report) are notoriously inflexible. You would like to change how the abstract appears, say? Then redefine the `abstract` environment. Same thing with figure captions, and document titles, and so on. Sooner or later, someone writes a package that provides a convenient user interface; here are those of Peter's.

abstract Easily customise the `abstract` environment for one- or two-column typesetting.

appendix Provides additional appendixing[4] capabilities.

ccaption Provides many features for customising and extending captions in floating and non-floating environments. See also the caption package.

romannum Change (any combination of) various document counters, such as captions, sections, equations, etc., to use roman numerals.

tocloft Easily customise the table of contents and other 'List of . . . ' sections. See also the titletoc package.

titling Easily customise the document title produced with `\maketitle`.

tocbibind Add (perhaps with customizations) the table of contents, bibliography, index, etc., to the actual table of contents.

tocvsec2 Adjust the relationship between section headings and table of contents listing, and adjust automatic section numbering, mid-document.

For example, remove the number from a group of subsections and suppress their appearance in the table of contents without changing their markup.

4.2 LaTeX programming tools

LaTeX 2ε's programming interface is often limited, offering little more in some areas than plain TeX's 'primitive' functionality. Peter's packages in this area tend to provide abstractions for specific tasks in LaTeX, useful for other class or package authors.

bez123 & multiply Draw generalised bezier curves in LaTeX, and multiply lengths without overflow. See also pict2e and the more ambitious drawing packages PSTricks and pgf/Ti*k*Z.

chngcntr Change the rules for the resetting of counters, such as numbering equations per-chapter or per-document.

chngpage and changepage Tools to locally change the size of the typesetting space and to detect robustly whether a page is even or odd.

N.B. that the two packages perform the same tasks, but changepage is interface-compatible with memoir and should be used for all new code that require these features. chngpage is an older version that is *incompatible* with memoir.

docmfp Extend doc to aid documentation of code in other programming languages. See also xdoc2 and (more recently) gmdoc.

ifmtarg Robust and expandable test for 'emptiness' of a macro argument.

makecmds LaTeX equivalent for `\def` with the syntax of `\newcommand` and `\newenvironment` (i.e., creates or overwrites the definition with equal abandon).

newfile Convenient interface to TeX's file reading and writing commands.

nextpage Extending the family of `\clearpage` commands. (E.g., `\cleartoevenpage`.)

printlen Print lengths of counters in specified units (as opposed to points, the TeX default).

stdclsdv Detect whether the class provides `\chapter`, and other sectional divisions.

[3] Having said this, each package name is hyperlinked if you're reading this electronically.

[4] I think Peter invented this word.

4.3 New and assorted features

The final section contains packages that provide document authors with features not offered by standard LaTeX classes or packages.

anonchap Makes `\chapter` typeset like `\section`. (E.g., for converting a book chapter into an article without changing the sectioning markup.)

booklet Typeset documents arranged on paper to be folded into booklets.

combine Combine multiple entire LaTeX documents into a single output file.

epigraph Add quotation-like material at the beginning/end of sections or chapters.

fonttable Visualise a font's glyph repertoire.

hanging Typeset paragraphs with hanging indents, and enable hanging punctuation using active characters.

For hanging punctuation, the microtype package for pdfTeX is recommended instead (although I may, in time, update hanging to work without active characters in XeTeX, using the latter's `\interchartoks` feature).

hyphenat Control hyphenation: turn it off entirely or allow the use of analphabetic symbols in hyphenated words.[5]

layouts Visualise the design of the page layout.

midpage An environment to vertically centre its contents in the text block.

needspace Reserve a certain amount of space on a page when you want to insert some material without breaking it over pages; if it cannot fit it will be forced it to the next page if necessary (ending the current page prematurely).

pagenote Typeset end notes per chapter or per document. See also the endnotes package.

verse Typeset verse material. See also poemscol.

vertbars Place vertical bars in the margin of paragraph text. Based on lineno, with the same caveats. See also the changebar package.

xtab Extensions and improvements to the package supertabular for multipage tables.

4.4 Classes and packages that I do not maintain

memoir Arguably Peter's single most influential contribution to LaTeX; memoir is a complete replacement for the standard document classes. It incorporates many of the packages mentioned above and contains a suite of new functionality, all with a consistent interface for creating new document designs. Now maintained by Lars Madsen. See also the KOMA-Script classes.

memdesign Originally the first half of the memoir manual. While this document is titled 'A Few Notes on Book Design', it's actually an excellent primer on typesetting and typography in general. Essential reading.

ledmac, ledpar, and ledarab For typesetting critical editions, based on plain TeX code 'edmac' and others. Now maintained by Vafa Khalighi.

expressg METAPOST package for drawing diagrams that consist of boxes, lines, and annotations.

iso and iso10303 LaTeX packages and classes for typesetting ISO standards. Possibly out of date with respect to the current typesetting standards.

isorot Rotate document elements and paragraph text. Perhaps I should maintain this one as well; it seems to be of more general interest than the other 'iso' packages.

[5] Good spot for a hyphen, there, hey?

5 Conclusion

I think it's important for members of open source communities to pass down their work as they start to retire from the field. Having a succession of maintainers allows bugs to be fixed and removes any confusion about how updates to their work should be named and distributed.

Would I continue to take on maintainership of yet more packages? Generally speaking, yes, provided the workload doesn't increase too much. Of course, we can't continue working with older and older packages indefinitely; at some stage new solutions to old problems will be created that supersedes the old work. (Cue my current involvement with the LaTeX3 Project.) In which case, old bugs in old code don't really need to be fixed.

But as long as Peter's packages continue to be useful (and most of them certainly are), I believe they deserve at least enough attention to keep them ticking along smoothly.

◇ Will Robertson
University of Adelaide, Australia
will dot robertson (at)
latex-project dot org

Biuletyn GUST 25–26 (2008–2009)

Editor's note: *Biuletyn GUST* is the publication of GUST, the Polish language TEX user group. Their web site is `http://www.gust.org.pl`.

Biuletyn GUST 25 (2008)

Ulrik Vieth, Zrozumieć estetykę składu matematyki [Understanding the æsthetic of math typesetting]; pp. 5–21

One of the core strengths of TEX is the ability to typeset math to a very high level of æsthetic standards. However, this level of quality not only depends on TEX alone, but relies on close interaction between sophisticated algorithms (built into the TEX engine) and the fine-tuning of metric information (built into math fonts), which is not so well understood.

At a previous conference Bogusław Jackowski presented a paper, "Appendix G Illuminated", in which he translated the formal description of TEX's algorithms for math typesetting into a visual representation, illustrating the mathematical and geometric relations between the various font metric parameters. While this helps to improve the understanding, it doesn't resolve the question of how to determine good values of font metric parameters when designing a new font.

In this paper, we analyze the values of these parameters in existing fonts and draw some conclusions about the underlying design principles. In the end, we hope to obtain a recipe for how to determine good values of font metric parameters based on simple design parameters such as the x-height or rule thickness.

Jonathan Kew, Co nowego w świecie XeTEX-a? [What's new in the XeTEX world?]; pp. 22–27

This presentation will review the current state of the XeTEX engine and associated packages, with an emphasis on features that have been added or updated over the past year, and will be included in the TEX Live 2008 release. These include updated OpenType support, synthetic font styles, the polyglossia macro package, updated math font support, built-in source/PDF synchronization support, and more.

We will also look at how XeTEX fits in with other TEX engines and with the wider software world. XeTEX is not the only extension of TEX providing Unicode and modern font support, but it is based on a quite different approach from others, both old (Omega) and new (LuaTEX). Each has both strengths and weaknesses, which users should consider when choosing the appropriate tool for their particular needs.

Karel Píška, Testy fontów TEX Gyre (wiosna 2008) [Tests of the TEX Gyre fonts (Spring 2008)]; pp. 28–33

The contribution presents the results of verification of previous and newest versions of the TEX Gyre fonts, shows examples of various testing techniques, overviews suggestions and bugs; many of them, reported earlier, have already been fixed in the recent releases. "Synoptic tables" have been produced to list a full glyph repertoire to check it for correctness of shapes, completeness, consistency between regular, bold, italic, and bold italic faces, etc. We will also discuss the status and the future of the Cyrillic and Greek parts.

Mateusz Kmiecik, Od Logo do Metaposta [From Logo to MetaPost]; pp. 34–40

The Logo language (turtle graphics) is recommended for teaching of informatics at secondary schools in Poland. It is a quite primitive language so young people quickly hit the limits when programming more advanced pictures. Could MetaPost be used to this end?

Piotr Krakowiak and Tomasz Łuczak, Dragonia Magazine – kulisy redakcji [The Dragonia Magazine backstage]; pp. 41–43

The Dragonia Magazine is a monthly free e-zine published since 2006. It is devoted to GNU/Linux and free software. Most of the issues were typeset with Scribus, one with InDesign but the latest are typeset with LATEX. We will show the abilities and restrictions of typesetting of a periodical of several dozen pages with Scribus, the reasons behind the migration to typesetting with LATEX and the results — or what was gained and what was lost.

Jacek Kmiecik, Dostosowanie LATEX-a do konkretnych potrzeb [The tuning of LATEX to one's purposes]; pp. 44–52

A few of the simplest ways for tuning the canonical LATEX macros to one's own typographical requirements will be presented.

Jonathan Kew, TEXworks: obniżenie progu dostępności [TEXworks: Lowering the barrier to entry]; pp. 53–59

Published in *TUGboat* 29:3.

Grzegorz Murzynowski, Styl bibliograficzny pl64.bst [A bibliography style: 'pl64.bst']; pp. 59–60

I generated the `pl64n` BibTEX bibliography style ('n' for `natbib`) with `makebst` and then fine tuned-some details by hand to make the style as compliant

with the spirit of Polish standards as possible, making my own choices where not determined by the standards or not available with `makebst`.

Grzegorz Murzynowski, O składaniu listów E. Szarzyńskiego trojako: estetycznie, XeTeX-owo, troszkę hackersko [Threefold on typesetting of E. Szarzyński's letters: æsthetic-, XeTeX-, and a little hacker-wise]; pp. 61–66

The "102 listy" by E. Szarzyński is a series of letters which he wrote to himself just before committing suicide. Fairy tales on colorful papers with calligraphic handwriting have been bound into a book which by being richly adorned reminds one of medieval incunabula. Preparing it for print we decided on one color (black) ink and instead of reproducing the adornments to provide their descriptions which gives our edition a rather conceptual character.

I intend to report on fonts we used and typographical conventions which aim at a clear separation of the editorial level, which describes the original, from Szarzyński's text, at the same time trying to reflect the conventions used by the author.

The engine I am using since at least half a year is XeTeX, currently at version 0.997. I am using the availability of the system OpenType fonts, the possibilities to easily declare the features available through fontspec and the new conveniences offered by XeTeX like the pseudo-feature "slant". I will describe how XeTeX serves the æsthetic of a book.

I intend to give the audience a few words on the basic typesetting parameters, such as `\tolerance` and `\emergencystretch` in the context of their role for the appearance of the work.

If time allows, I will present a few tricks, such as typesetting of a fragment longer than a page in such a way that all "t" characters are typeset as crosses (as in Szarzyński's original).

Jean-Michel Hufflen, XSLT 2.0 vs. XSLT 1.0; pp. 67–77

This article focuses on the new features introduced by version 2.0 of XSLT, the language of transformations used for XML texts. We show why these new features — groups of XML subtrees, functions, interface with schemas — ease the development of some applications. Some examples, related to bibliography management, will be demonstrated.

Marcin Woliński, bayerancki – mój pierwszy font [bayerancki — my first font]; p. 78

The subject of this talk is my first attempt at creating a font. The font is named "bayerancki" and is (loosely) based on Herbert Bayer's "Universal" from 1925. The font has been prepared with MetaType1.

Reinhard Kotucha, Siep Kroonenberg, and Norbert Preining, Nowy instalator dla TeX Live [A new installer for TeX Live]; pp. 79–81

TeX Live has a new package infrastructure, primarily developed by Norbert Preining, and inspired by the Debian/Linux packaging system.

We shall present a new TeX Live installer, based on the new package infrastructure. It includes a text based as well a graphical user interface. Among other new features, installing TeX Live from the internet is now possible. It should work on all platforms supported by TeX Live.

Jean-Michel Hufflen, Jeszcze raz o porządku leksykograficznym wg. osób [Revisiting lexicographic order relations on personal names]; pp. 82–90

Published in *TUGboat* 29:1 as "Managing order relations in MlBibTeX".

Grzegorz Murzynowski, Warsztat: Krótki kurs niepowtarzania się, czyli: Napisz klasę lub pakiet. [A short course in "Don't Repeat Yourself" or "Write a class or package"]; pp. 91–95

We will elaborate on what Marcin [Woliński] mentions rather than describes in the MWCLS documentation, i.e., we will demonstrate the writing of our own document class based on MWCLS or on standard classes:

- the required commands
- loading of the base class (`\LoadClass`)
- loading of the desired packages (`\RequirePackage`)
- declaring of options and passing them to the base class and packages (`\DeclareOption`, `\PassOptionsToPackage/Class`)
- if time allows, options of the key-value form

Documenting the class at write-time (literate programming) with the gmdoc package:

- where to put the class written and what should be done for TeX to see it.
- what are the 'at' signs, `\edef` and other basic TeX commands for; or *The TeXbook* and the LaTeX source as enlightening.
- See how others do it or read the `.dtx` instead of the `.cls` and `.sty` files.

Placing macros in the package:

- the difference between a package and a class
- when to write a package and when a class.

Why so? The DRY rule (Don't Repeat Yourself (Twice Is Too Much)).

Hans Hagen, OpenType: zbyt otwarty? [OpenType: Too open?]; pp. 96–98

One advantage of the need to provide support for the fonts that are used in the Oriental TeX project

(the primary funding project of LuaTEX code development) is that we need to support OpenType fonts with more than average features in ConTEXt (needed by Oriental TEX). In this talk I will reflect on the possible use, abuse and/or misuse of OpenType fonts, the complications in implementing the standard(s), and the consequence for TEX users.

Hans Hagen, Biblioteka MetaPost [The MetaPost Library]; pp. 99–113
Published in *TUGboat* 28:3.

Hans Hagen and Taco Hoekwater, Lua z TEX-em i ConTEXt-em [The luafication of TEX and ConTEXt]; pp. 114–123
Published in *TUGboat* 29:2.

Hans Hagen, Taco Hoekwater, and Volker rw Schaa, Font Euler na nowo: wspólpraca z Hermanem Zapfem [Reshaping Euler: A collaboration with Hermann Zapf]; pp. 124–128
Published in *TUGboat* 29:2.

Jean-Michel Hufflen, TEX-owe narzędzia składu dla języków Dalekiego Wschodu [Asian TEX-like typesetting engines]; pp. 129–131

In order to extend MlBibTEX to languages of the Far East, we are experimenting with TEX engines for them — e.g., pTEX — after attending the first Asian TEX conference. We give a demonstration of that.

Paweł Jackowski, TEX: śliczności i dziwności [TEX: Beauties and oddities]; pp. 140–144

See the TEX Pearls web page: `http://www.gust.org.pl/projects/pearls`.

***Biuletyn GUST* 26 (2009)**

Ulrik Vieth, Matematyka OpenType iluminowana [OpenType Math Illuminated]; pp. 7–16
Published in *TUGboat* 30:1.

Jean-Michel Hufflen, Wprowadzenie do XQuery [Introduction to XQuery]; pp. 17—25

XQuery is a query language for data stored in XML form. It can be used to search such documents and arrange the result, as an XML structure or simple text (possibly suitable for a TEX engine). Like XSLT 2.0, it is based on XPath 2.0. We propose an introduction to XQuery, and some comparisons with XSLT to allow readers to discern the applications for which XQuery is suitable.

Jean-Michel Hufflen, Jak jest zorganizowana dokumentacja MlBibTEXa [How MlBibTEX's documentation is organised]; pp. 26–30

MlBibTEX's documentation is planned to be multilingual — that is, written in several languages — and to be able to share as many examples as possible. Different people can write translations of the orginal English documentation in parallel. In addition, we show how the translations of this documentation can be updated if need be. This documentation can be used as printed text or an on-line document. The features for managing this documentation can be reused for other programs. In a first part, we explain our requirements in detail. Then we show how they are implemented.

Marek Ryćko, Projektowanie programowalnych aspektów oprogramowania typograficznego [Designing programmable aspects of typographic software]; pp. 31–33

It has been 32 years from the time when Donald Knuth started to design and write a typographic system, consisting of a program for computer typesetting, and a program for creating character shapes. Those programs are still in use now and a lot of effort is put into creating their contemporary versions.

One of the recent achievements in programming in the typographic area is a new version of the MetaPost program, for creating vector graphics. The effort went into separating the program code into a library, called MPlib, and a small program, MetaPost, that uses this library.

In the talk I plan to look closely at the current implementation of MPlib (version 1.110) from the software designer point of view. I will show, in a step-by-step style, a possible way of thinking in designing a new program, that will include a functionality of MetaPost/MPlib, but will be created using programming techniques and tools that were not available 32 years ago. The design will be object-oriented and based on a very high-level programming language.

The result of the design process, carried on in the presented way, will be transparent program code, easily understandable, extendable, and easy to use as a part of other programs and systems.

Philip Taylor, Preprocesor parshape [A Parshape Pre-processor]; pp. 34–36

`\parshape` is a powerful but potentially verbose primitive that provides the hooks necessary for wrapping text around graphics or other figures. When the outline of the inserted figure is rectangular, considerable simplification can be accomplished by providing a `\parshape` pre-processor in TEX.

Jean-Michel Hufflen, Skład wielokierunkowy w XSL-FO [Multi-directional typesetting in XSL-FO]; pp. 37–40

XSL-FO is an XML format that aims to describe high-quality print output. This article complements the introduction to XSL-FO given at EuroBachoTEX

2007. We show how XSL-FO allows users to typeset texts belonging to different writing systems: from left to right, from right to left, etc. We compare this implementation to TEX-like typesetting engines, e.g., TEX--XET.

Ryszard Kubiak, Tworzenie dokumentacji oprogramowania dla użytkowników w LATEXu i Hyperlatexu [Creating software documentation for users with LATEX and Hyperlatex]; pp. 41–46

A LATEX-based notation is shown in the article as a language for writing source versions of software manuals for its users. Books or brochures as PDF files can be prepared simply by using TEX and their HTML version can be generated from the same source files by calling Hyperlatex from the Emacs editor. The author describes his experience in using this technology.

Chris Rowley, Składanie poza \box-em: zaczyn dyskusji [Typesetting outside the \box: A discussion item]; pp. 47–49

There are a some fundamental problems in basing the future of high-quality, highly-automated typographic software on extending the monolithic and programmatic paradigm of current TEX-related software developments.

It is therefore time to take seriously other paradigms and strategically different software architecture, moving on to a development path that will attract and exploit both the typographic and programming expertise of TEX gurus and also other sources of high-quality software design for document processing.

Paweł Jackowski, TEX: śliczności i dziwności [TEX: Beauties and oddities]; pp. 59–65

See the TEX Pearls web page: `http://www.gust.org.pl/projects/pearls`.

[Received from Tomasz Przechlewski]

Die TEXnische Komödie 2009/3

Editor's note: *Die TEXnische Komödie* is the journal of DANTE e.V., the German-language TEX user group (`http://www.dante.de`).

Jürgen Fenn, News regarding LATEX export from OpenOffice.org

This article presents news regarding the LATEX export from the free office suite OpenOffice.org 3.1 (`Writer2LaTeX`, `Calc2LaTeX`).

Christine Römer, Linguistic examples put in perspective

There are several packages available for the customary continuous numbering of German examples as well as for mono- and multilingual interlinear glosses. This article presents the TIPA package, the gb4e package including its extension cgloss4e and the new package expex which has connections with PSTricks. The covington package is also mentioned. The advantage of the expex package is that the user can customise the layout of the examples to his needs. As it uses TEX commands, it is more flexible than the existing LATEX packages, but presumably not very convenient for beginners. None of the considered packages are suitable for including interlinear translations with non-Latin characters.

Adelheid Grob, Easy typesetting of lists with easylist

LATEX provides the environments `itemize` and `enumerate` for typesetting lists. A nesting depth of four is allowed, but sometimes it is not easy to prevent errors when having a depth of three or four. The package easylist from Paul Isambert allows any depth and easy modifications of list parameters.

Rolf Niepraschk, Formatting table columns — all bets are off

This article presents a universal way to format table columns.

Rolf Niepraschk, Automatic tables

This article describes one possibility to produce lines in LATEX tables automatically.

[Received from Herbert Voß.]

Zpravodaj 19(1–2) (2009)

Editor's note: *Zpravodaj* is the journal of CSTUG, the TEX user group oriented mainly but not only to the Czech and Slovak languages (`http://www.cstug.cz`).

Pavel Stříž, Úvodník [Opening Letter from the Editorial Board]; p. 1

Subjekty dlouhodobě podporující Zpravodaj [Parties supporting the journal on a long-term basis]; p. 2

Jiří Veselý, Několik poznámek k historii CSTUG. Trocha neformálních vzpomínek [Historical notes on the Czechoslovak TEX Users Group (CSTUG)]; pp. 3–6

The report gives a long history of the Czechoslovak TEX Users Group from its early beginnings.

Let's go back in time to 1977 (TEX beginnings), 1986 (typesetting documents with heavy math in Central Europe at that time), 1989 (paper preparations of CSTUG), 1990 (CSTUG was born), 1991 (the first Zpravodaj journal, issue No. 1/1991, was published), 1992 (EuroTEX organised by Czechoslovaks), 1996 (Knuth lecturing at Masaryk University in Brno, Czech Republic) and back to the present.

Robert Maříк, Oslí okénka v PDF [The fancytooltips package]; pp. 7–13

The article presents the fancytooltips package which includes a wide range of material in PDF tooltips. The package was inspired by the cooltooltips package. In contrast to cooltooltips, fancytooltips allows the inclusion of tooltips which contain arbitrary TEX material or a series of TEX materials (animated graphics) from an external PDF file. To see the tooltips, you have to open the files in Adobe Reader. The links and JavaScripts are inserted using the eforms package from the AcroTEX bundle (acrotex). The conversion into HTML format is also discussed. The package is available at `www.ctan.org`.

Norbert Preining, TEX Live 2008 a správa inštalácie TEX Live [TEX Live 2008 and the TEX Live Manager, Slovak version; translated from English by Ján Buša]; pp. 14–36

TEX Live 2008 has been released recently, and the DVDs are ready to go gold. This is the first release of TEX Live shipping the TEX Live Manager.

Besides taking over some of the tasks from `texconfig` (which has never been available for Windows) it finally brings many new features to the TEX Live world, most importantly the option for dynamic updates.

This article presents the new TEX Live installer, the TEX Live Manager, and at the end lists other changes in TEX Live 2008.

Pavel Stříž, Vybrané partie knižní sazby aneb Píšeme divadelní hru [Selected problems from book typesetting: The case of stageplay typesetting]; pp. 37–89

This article summarizes the author's experience of typesetting a drama playbook. The paper itself is not meant to be a package description because there is no new package. Readers and potential playwrights are encouraged to think about what they would like to typeset and then to get on with their paper. In this extended version, the author demonstrates about a dozen partial problems concerned with stageplay typesetting and presents his solutions.

Marcel Svitalský, Google Summer of Code – představení [Google Summer of Code: An Introduction]; p. 90

The report is a translation of the first paragraphs of the wiki page about Google Summer of Code, `http://en.wikipedia.org/wiki/GSoC/`.

Marcel Svitalský, Google Summer of Code 2008 a TUG [Google Summer of Code 2008 and TUG, English version: Karl Berry]; pp. 91–93

The report presents application information and code sample sources of three projects which were accomplished by students working with the TEX Users Group for Google Summer of Code 2008, `http://tug.org/gsoc/2008.html`.

Marcel Svitalský, Google Summer of Code 2009 a TUG [Google Summer of Code 2009 and TUG, English version: Karl Berry]; pp. 94–101

The report is a Czech translation of the web page presenting ideas for the TEX Users Group for Google Summer of Code 2009, `http://tug.org/gsoc/`.

Marcel Svitalský, Projekt Dublin Core metadata interface [The Dublin Core Metadata Interface Project, English version: Matthew Leingang & Peter Flynn]; pp. 102–106

The report brings a Czech translation of the web page `http://tug.org/gsoc/dublincore.html`, describing one of the projects proposed for the GSoC 2009 with TUG as a mentoring organisation. The goal of the project was the implementation of the Dublin Core Abstract Model in TEX.

Nové a staronové knihy [New and older books]; pp. 107–111

Donald E. Knuth: Umění programování. Základní algoritmy [The Art of Computer Programming, Volume 1, Czech translation].

Pavel Kočička, Filip Blažek: Praktická typografie [Practical Typography, in Czech].

František Štorm: Eseje o typografii [Essays on Typography, in Czech].

Helmut Kopka, Patrick W. Daly: LATEX. Kompletní průvodce [A Guide to LATEX, Czech translation].

Petr Olšák: TEXbook naruby [TEXbook Inside Out, in Czech].

Petr Olšák: Typografický systém TEX [Typesetting System TEX, in Czech].

Jiří Rybička: LATEX pro začátečníky [LATEX for Beginners, in Czech].

Redakční poznámky a pokyny autorům [Notices and instructions for the authors, versions in Czech and English]; pp. 112–116

[Received from Pavel Striz]

TUG 2009 abstracts

Editor's note: Many of the conference presentations are available at `http://www.river-valley.tv` in video form, thanks to Kaveh Bazargan and River Valley Technologies.

Jin-Hwan Cho

Extended DVI formats and DVIasm

DVIasm is a TEX utility program that is designed for editing DVI files with three steps: disassembling, editing, and assembling.

The first stage of DVIasm [1] supported the standard DVI file format as in DVItype and DTL, but in a much more flexible way than those utilities. In the second stage [2], DVIasm made it possible to handle two-byte characters, CJK and Unicode characters. The extended DVI formats generated by Omega, Japanese pTEX were all supported, as well as ordinary LATEX packages with subfont scheme such as CJK-LATEX and Korean *ko*.TEX. The final stage of DVIasm confronts the two advanced TEX engines, LuaTEX and XETEX, both of which can handle OpenType and TrueType files in a direct way without TFM files.

In this talk we introduce DVIasm with a few interesting applications to the TEX world, and discuss how DVIasm handles the extended DVI formats generated by LuaTEX and XETEX.

[1] Jin-Hwan Cho, *Hacking DVI files: Birth of DVIasm*, *The PracTEX Journal* (2007), no. 1, and *TUGboat* **28** (2007), no. 2, 210–217. `http://tug.org/TUGboat/Articles/tb28-2/tb89cho.pdf`.

[2] Jin-Hwan Cho, *Handling Two-Byte Characters with DVIasm*, The Asian Journal of TEX **2** (2008), no. 1, 63–68. `http://ajt.ktug.kr/assets/2008/5/1/0201cho.pdf`.

Paulo Ney de Souza

The MSP TEX production system: Restoration of TUGboat

Mathematical Sciences Publishers will begin to offer *TUGboat* to its subscribers (primarily academic libraries) soon. This talk will discuss the challenges in bringing all thirty volumes of *TUGboat* into MSP's electronic delivery system via its web site, `http://www.mathscipub.org`.

Idris Hamid

Dynamic Arabic: Towards the philosopher's stone of Arabic-script typography

We discuss the present status of the Oriental TEX project, particularly the problem of Arabic-script microtypography. This includes glyph substitution and hz parameterization.

Idris Hamid

Arabic typography: Past, present, and TEX

Accommodating the classical Arabic script to print typography has always been a challenge. In this talk we go over some of the history of this effort — including TEX-based solutions — with a view to providing a backdrop to the Oriental TEX Project and its progress.

Morten Høgholm

Consolidation of expl3

The expl3 language used as the foundation of LATEX3 has gone through a consolidation phase where almost each and every concept has been questioned, taken apart and put back, sometimes in the same form as it was and sometimes in radically different forms. We will go through some of the most interesting changes and highlight the areas where special effort has been made to ensure simple and natural interfaces.

Morten Høgholm

Next steps for breqn

The next version of the breqn package for automatic line breaking of displayed equations is underway. We will discuss different areas of math typesetting: some things breqn handles well, some areas have room for improvement, and some areas are simply not covered. We will spend some time talking about the technical challenges posed by these requirements.

Chris Rowley

TEX-free LATEX, an overview & Standards for LATEX documents and processors or *Whither LATEX? (The language)*

The LATEX is the message. As some of you will be aware, and all should be, LATEX code, possibly with some variations, extensions or simplifications, has for a long time been used, raw and unprocessed, as a lingua franca for communicating mathematics via text files in computers. [I have even seen it used on napkins and coffee tables.]

This has led to a proliferation of LATEX-like input systems for mathematical information and this in turn produced a reluctance by users of maths notation to adopt any other type of input. However, much of this math input is not intended (primarily) to ever be input to a TEX machine. (It may get swallowed by a TEX-like system after, for example, some copy/paste actions.)

More recently, systems are being developed to produce whole LaTeX-encoded documents that are to be processed by systems such as OMDoc or LaTeXML and so will not necessarily ever pass through a TeX-like engine. Systems such as PlasTeX also belong in this category, despite using TeX as a helper utility in their implementation.

A very recent discovery surprised me more than a little: that many systems in the maths world are not only able to produce LaTeX output (e.g., computer algebra packages) but, currently at least, have LaTeX maths as their only or primary output! This is because: it is wanted or preferred by mathematicians; it is widely accepted by other mathematical software; or simply that nothing else is known to be available for a consistent and familiar encoding of maths notation.

A more sophisticated reason put forward for the increasing ubiquity of LaTeX is that if you are looking for a user-friendly and flexible editor for structured documents, then there are no rivals to the various environments available for the production and editing of LaTeX documents (such as `auctex+(x)emacs`).

Standards. What *standards?* It would be possible to make an exhaustive list of everything that is allowed to appear in a Standard Basic LaTeX document. But that would be both tedious, uncheckable and ignored.

It is currently much easier to pin down which parts of the LaTeX language are accepted by the various non-TeX-like processors of LaTeX (from the first part). Also, there are corpora that can be automatically studied to produce definitions of the subsets actually used by various communities.

Amongst those who handle mathematics in computers there is a growing demand to analyse these de facto standards, at least for LaTeX-math, and to produce reference standards in this area. These would be used to compare systems and communities and make recommendations for usage. This could possibly lead to some more formal standards and, most importantly, extension mechanisms so that, for example, general-purpose parsers can be used to read such code.

TUG Institutional Members

American Mathematical Society,
Providence, *Rhode Island*

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

Banca d'Italia,
Roma, *Italy*

Center for Computing Sciences,
Bowie, *Maryland*

Certicom Corp.,
Mississauga, *Ontario*, *Canada*

CSTUG, *Praha*, *Czech Republic*

Florida State University,
School of Computational Science and Information Technology,
Tallahassee, *Florida*

IBM Corporation,
T J Watson Research Center,
Yorktown, *New York*

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

MacKichan Software, Inc.,
Washington/New Mexico, *USA*

Marquette University,
Department of Mathematics, Statistics and Computer Science,
Milwaukee, *Wisconsin*

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

Moravian College,
Department of Mathematics and Computer Science,
Bethlehem, *Pennsylvania*

MOSEK ApS,
Copenhagen, *Denmark*

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

Princeton University,
Department of Mathematics,
Princeton, *New Jersey*

Springer-Verlag Heidelberg,
Heidelberg, *Germany*

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

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

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

University of Delaware,
Computing and Network Services,
Newark, *Delaware*

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

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

RIVER VALLEY

TECHNOLOGIES

TUG 2010: TEX's 2^5 anniversary

and the 31st annual meeting of the TEX Users Group

Sir Francis Drake Hotel, Union Square

San Francisco, California

June 28–30, 2010

http://www.tug.org/tug2010/

With guests of honor

Donald E. Knuth

and many other members of

the Stanford TEX project

Calendar

2009

Sep 6 – Nov 23 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. San Francisco Public Library, San Francisco, California. Sites and dates are listed at palimpsest.stanford.edu/byorg/gbw

Sep 15 – 18 ACM Symposium on Document Engineering, Munich, Germany. www.documentengineering.org

Sep 25 "A Short History of Printing": lecture by Frank Romano, Museum of Printing, North Andover, Massachusetts. www.museumofprinting.org/txp/Events

Sep 30 – Oct 1 XML-in-Practice 2009, Arlington, Virginia. www.idealliance.org/conferences_and_events

Oct 16 – 18 American Printing History Association's 34th annual conference, "The Book Beautiful", Newport, Rhode Island. www.printinghistory.org/htm/conference/2009/CFP-2009.htm

Oct 16 – 18 The Seventh International Conference on the Book, University of Edinburgh, Scotland. booksandpublishing.com/conference-2009

Oct 17 GuIT meeting 2009 (Gruppo utilizzatori Italiani di TEX), Pisa, Italy. www.guit.sssup.it/guitmeeting/2009

Oct 26 – 30 Association Typographique Internationale (ATypI) annual conference, "The Heart of the Letter", Mexico City. www.atypi.org

Oct 26 – Nov 13 "Late letterpress: The work of Desmond Jeffrey", exhibition, with a talk on Oct 27, St Bride Library, London, England. stbride.org/events

Oct 29 – 31 Guild of Book Workers, Standards of Excellence Annual Seminar, San Francisco, California. palimpsest.stanford.edu/byorg/gbw

Nov 21 Journée GUTenberg & Assemblée générale, Centre FIAP, Paris, France. www.gutenberg.eu.org/manifestations

Dec 7 – Feb 19 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. Allen Library, University of Washington, Seattle. See September 6 entry for additional information.

2010

Mar 5 – Apr 22 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. University of Utah Marriott Library, Salt Lake City. See 6 September 2009 entry for additional information.

May 3 – Jun 25 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. Denver Public Library, Denver, Colorado. See 6 September 2009 entry for additional information.

TUG 2010
San Francisco, California.

Jun 28 – 30 The 31st annual meeting of the TEX Users Group — TEX's 2^5 Anniversary. tug.org/tug2010

Jul 26 – 30 SIGGRAPH 2010, Los Angeles, California. www.siggraph.org/conference/s2010

Aug 17 – 20 TypeCon 2010: Los Angeles, California. www.typecon.com

Aug 17 – 21 SHARP 2010, "Book Culture from Below", Society for the History of Authorship, Reading & Publishing, Helsinki, Finland. www.helsinki.fi/sharp2010

Oct 16 – 18 The Eighth International Conference on the Book, University of St. Gallen, St. Gallen, Switzerland. booksandpublishing.com/conference-2010

Status as of 15 September 2009

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 206 203-3960, e-mail: 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.

Other calendars of typographic interest are linked from tug.org/calendar.html.

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.
To place a larger ad in *TUGboat*, please see `http://tug.org/TUGboat/advertising.html`.

Dangerous Curve
PO Box 532281
Los Angeles, CA 90053
+1 213-617-8483
Email: `typesetting (at) dangerouscurve.org`
Web: `http://dangerouscurve.org/tex.html`
We are your macro specialists for TEX or LATEX fine typography specs beyond those of the average LATEX macro package. If you use XETEX, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical TEX and LATEX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a TEX book.

Martinez, Mercè Aicart
Tarragona 102 4^o 2^a
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) ono.com`
Web: `http://www.edilatex.com`
We provide, at reasonable low cost, TEX and LATEX typesetting services to authors or publishers worldwide. We have been in business since the beginning of 1990. For more information visit our web site.

Peter, Steve
New Jersey, USA
+1 732 287-5392
Email: `speter (at) mac.com`
Specializing in foreign language, linguistic, and technical typesetting using TEX, LATEX, and ConTEXt, I have typeset books for Oxford University Press, Routledge, and Kluwer, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. I have extensive experience in editing, proofreading, and writing documentation. I also tweak and design fonts. I have an MA in Linguistics from Harvard University and live in the New York metro area.

Shanmugam, R.
No. 38/1 (New No. 65), Veerapandian Nagar, Ist St.
Choolaimedu, Chennai-600094, Tamilnadu, India
+91 9841061058
Email: `rshanmugam (at) yahoo.com`
As a Consultant I provide consultation, technical training, and full service support to the individuals, authors, corporates, typesetters, publishers, organizations, institutions, etc. and I also support to leading BPO/KPO/ITES/Publishing companies in implementing latest technologies with high level of automation in the field of Typesetting/Prepress/Composition, ePublishing, XML2PAGE, WEBTechnology, DataConversion, Digitization, Cross-media publishing, etc. I have sound knowledge in creating Macros/Styles/Templates/Scripts and Conversions with automation using latest softwares in industry.

Sievers, Martin
Im Treff 8, 54296 Trier, Germany
+49 651 81009-780
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`
As a mathematician I offer TEX and LATEX services and consulting for the whole academic sector and everybody looking for a high-quality output. From setting up entire book projects to last-minute help, from creating citation styles to typesetting your math, tables or graphics — just contact me with information on your project.

Veytsman, Boris
46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`
TEX and LATEX consulting, training and seminars. Integration with databases, automated document preparation, custom LATEX packages, conversions and much more. I have about fourteen years of experience in TEX and twenty-seven years 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.

Introductory

Intermediate

Intermediate Plus

Advanced

Contents of publications from other TEX groups

Reports and notices

TUGboat

Volume 30, Number 2 / 2009
TUG 2009 Conference Proceedings

Introductory

Intermediate

Intermediate Plus

Advanced

Contents of publications from other TEX groups

Reports and notices