

# TUGBOAT

Volume 26, Number 3 / 2005

<b>General Delivery</b>	179	From the president / <i>Karl Berry</i>
	180	Editorial comments / <i>Barbara Beeton</i> Mimi Burbank retires; Brian {Hamilton Kelly}, 1945–2005; Erratum for <i>TUGboat</i> 25:2; E-mail addresses in <i>TUGboat</i> on line; The <i>TUGboat</i> schedule; Lucida Bright fonts now available from TUG; Knuth on NPR; Letters in stone
	183	Interview with Donald E. Knuth / <i>Gianluca Pignalberi</i>
	186	Implementing editors' ideas—lots of fun, sometimes even more trouble / <i>Andrzej Tomaszewski</i>
<b>Tutorials</b>	188	Minutes in less than hours: Using L <sup>A</sup> T <sub>E</sub> X resources / <i>Jim Hefferon</i>
	193	\starttext: Swelled rules and MetaPost / <i>Steve Peter</i>
<b>Typography</b>	196	Typographers' Inn / <i>Peter Flynn</i>
<b>Philology</b>	199	The alphabet tree / <i>Peter Wilson</i>
<b>Fonts</b>	215	Advanced font features with X <sub>E</sub> T <sub>E</sub> X—the <i>fontspec</i> package / <i>Will Robertson</i>
	224	ByZI <sup>A</sup> T <sub>E</sub> X: A package for typesetting “Byzantine” music / <i>Ioannis Vamvakas</i> and <i>Panagiotis Kotopoulis</i>
<b>Software &amp; Tools</b>	233	i-Installer: The evolution of a T <sub>E</sub> X install on Mac OS X / <i>Gerben Wierda</i>
	239	The MacT <sub>E</sub> X distribution / <i>Herbert Schulz</i>
	241	T <sub>E</sub> X Live for Debian / <i>Norbert Preining</i>
	243	Hyphenation patterns in ConT <sub>E</sub> Xt / <i>Hans Hagen</i>
<b>Graphics</b>	246	Diagxy, a Lego-like diagram package / <i>Michael Barr</i>
	250	Embedding fonts in MetaPost output / <i>Troy Henderson</i>
<b>Hints &amp; Tricks</b>	253	Glisterings / <i>Peter Wilson</i>
	256	Pearls of T <sub>E</sub> X programming
	264	The treasure chest / <i>Mark LaPlante</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	268	powerdot—making presentations with L <sup>A</sup> T <sub>E</sub> X / <i>Hendri Adriaens</i> and <i>Christopher Ellison</i>
	273	Horrors in L <sup>A</sup> T <sub>E</sub> X: How to misuse L <sup>A</sup> T <sub>E</sub> X and make a <i>copy editor</i> unhappy / <i>Enrico Gregorio</i>
<b>Macros</b>	280	Eplain 3: Expanded plain T <sub>E</sub> X / <i>Karl Berry</i> and <i>Oleg Katsiadze</i>
<b>Abstracts</b>	287	<i>T<sub>E</sub>Xemplares</i> : Contents of issue 7 (2005)
	287	<i>Zpravodaj</i> : Contents of issues 14(3–4), 15(1), 15(2) (2004–05)
	289	<i>Les Cahiers GUTenberg</i> : Contents of double issue 44–45 (2004)
	290	<i>MAPS</i> : Contents of issues 32 (2005)
	292	<i>The PracT<sub>E</sub>X Journal</i> : Contents of issues 2005-1–2005-4 (2005)
<b>News &amp;</b>	297	Calendar
<b>Announcements</b>	302	Practical T <sub>E</sub> X 2006 announcement
	303	EuroT <sub>E</sub> X 2006 announcement
	304	TUG 2006 announcement
<b>TUG Business</b>	298	Financial statements for 2005 / <i>Robin Laakso</i>
	300	T <sub>E</sub> X Development Fund 2005 report / <i>Karl Berry</i> and <i>Kaja Christiansen</i>
	301	TUG membership form
	302	Institutional members
<b>Advertisements</b>	300	T <sub>E</sub> X consulting and production services
<b>Index</b>	c3	Table of contents, ordered by difficulty

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

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

### Memberships and Subscriptions

2005 dues for individual members are as follows:

- Ordinary members: \$75.
- 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 T<sub>E</sub>X Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site, <http://www.tug.org>.

Also, *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 \$85 per year, including air mail delivery.

### Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both T<sub>E</sub>X and the T<sub>E</sub>X Users Group, as well as providing other benefits. For further information, contact the TUG office or see our web site.

T<sub>E</sub>X is a trademark of the American Mathematical Society.

Copyright © 2005 T<sub>E</sub>X Users Group.

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

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

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

Printed in U.S.A.

### Board of Directors

Donald Knuth, *Grand Wizard of T<sub>E</sub>X-arcana*<sup>†</sup>

Karl Berry, *President*\*

Kaja Christiansen\*, *Vice President*

David Walden\*, *Treasurer*

Susan DeMeritt\*, *Secretary*

Barbara Beeton

Steve Grathwohl

Jim Hefferon

Klaus Höppner

Ross Moore

Arthur Ogawa

Gerree Pecht

Steve Peter

Cheryl Ponchin

Samuel Rhoads

Philip Taylor

Raymond Goucher, *Founding Executive Director*<sup>†</sup>

Hermann Zapf, *Wizard of Fonts*<sup>†</sup>

\*member of executive committee

<sup>†</sup>honorary

### Addresses

General correspondence,  
payments, etc.

T<sub>E</sub>X Users Group  
P. O. Box 2311  
Portland, OR 97208-2311  
U.S.A.  
  
Delivery services,  
parcels, visitors

T<sub>E</sub>X Users Group  
1466 NW Naito Parkway  
Suite 3141  
Portland, OR 97209-2820  
U.S.A.

**Telephone**  
+1 503 223-9994

**Fax**  
+1 206 203-3960

### Electronic Mail (Internet)

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

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

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

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

### World Wide Web

<http://www.tug.org/>  
<http://www.tug.org/TUGboat/>

### Problems not resolved?

The TUG Board wants to hear from you:  
Please email [board@tug.org](mailto:board@tug.org).

[printing date: February 2006]

Only if your handwriting is so vile or your typing so incompetent that you yourself cannot read it do you really need a computer for composition.

Mary-Claire van Leunen  
*A Handbook for Scholars*, revised  
edition (1992)

# TUGBOAT

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

EDITOR BARBARA BEETON

VOLUME 26, NUMBER 3

PORTLAND

•

OREGON

•

•

2005

U.S.A.

## **TUGboat**

This issue (Vol. 26, No. 3) is the final issue of the 2005 volume year. Vol. 26, No. 1 combined the Practical TeX 2005 conference proceedings with regular material, and Vol. 26, No. 2 was the TUG 2005 (Wuhan, China) proceedings.

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

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

### **Submitting Items for Publication**

*TUGboat* will be publishing three conference proceedings in 2006. Deadlines for presentation proposals (send to the conference committees) and the final papers:

- TUG 2006: proposals June 1, 2006; papers June 30, 2006.
- EuroTeX 2006: proposals March 1, 2006; papers July 25.
- Practical TeX 2006: proposals April 1, 2006; papers September 1.

We expect at least one of these to also contain regular-issue material. Links, locations, and more information about all the conferences are available at <http://tug.org/meetings.html>.

As always, suggestions and proposals for *TUGboat* articles are gratefully accepted and processed as received. We encourage submitting contributions by electronic mail to [TUGboat@tug.org](mailto:TUGboat@tug.org).

The *TUGboat* "style files", for use with either plain TeX or LATEX, are available from CTAN and the *TUGboat* web site. We also accept submissions using ConTeXt.

As of this 2005 volume year, submission of a new manuscript implies permission to publish the article, if accepted, on the *TUGboat* web site, as well as in print. If you have any reservations about posting online, please notify the editors at the time of submission.

## **TUGboat Editorial Board**

Barbara Beeton, *Editor-in-Chief*

Robin Laakso, *Managing Editor*

Mimi Burbank, *Production Manager*

Victor Eijkhout, *Associate Editor, Macros*

Alan Hoenig, *Associate Editor, Fonts*

Christina Thiele, *Associate Editor,*

*Topics in the Humanities*

## **Production Team**

William Adams, Barbara Beeton, Karl Berry,

Mimi Burbank (Manager), Kaja Christiansen,

Robin Fairbairns, Baden Hughes, Steve Peter,

Michael Sofka, Christina Thiele

## **Other TUG Publications**

TUG is interested in considering additional manuscripts for publication. These might include manuals, instructional materials, documentation, or works on any other topic that might be useful to the TeX community in general. Provision can be made for including macro packages or software in computer-readable form.

If you have any such items or know of any that you would like considered for publication, send the information to the attention of the Publications Committee at [tug-pub@tug.org](mailto:tug-pub@tug.org).

## **TUGboat Advertising**

For information about advertising rates and options, including consultant listings, write or call the TUG office, or see our web page:

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

## **Trademarks**

Many trademarked names appear in the pages of *TUGboat*. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which appear in this issue should not be considered complete.

METAFONT is a trademark of Addison-Wesley Inc.

PostScript is a trademark of Adobe Systems, Inc.

TeX and *AMS-TeX* are trademarks of the American Mathematical Society.

## General Delivery

### From the President

Karl Berry

### T<sub>E</sub>X Collection 2005

In November 2005, another edition of the T<sub>E</sub>X Collection software, a major joint effort by many T<sub>E</sub>X user groups, was released. A high-level summary is posted at <http://tug.org/texcollection>. All TUG members for 2005 should have received it by now; if not, please contact the TUG office.

As with last year, this year's collection includes:

- T<sub>E</sub>X Live (<http://tug.org/texlive>)—  
A comprehensive T<sub>E</sub>X system supporting most Unix architectures, including GNU/Linux and Mac OS X, and Windows.
- proT<sub>E</sub>Xt (<http://tug.org/protext>)—  
An easy-to-install Windows distribution based on MiK<sub>T</sub>E<sub>X</sub>.
- CTAN (<http://www.ctan.org>)—  
A snapshot of [dante.ctan.org](http://dante.ctan.org), the German node of the Comprehensive T<sub>E</sub>X Archive Network, with duplications elided.

This year also has one new major distribution:

- MacT<sub>E</sub>X (<http://tug.org/mactex>)—  
An easy-to-install T<sub>E</sub>X system for Mac OS X, based on gwT<sub>E</sub>X, and also including TeXShop and X<sub>E</sub>T<sub>E</sub>X, among other applications. (An article with more information about MacT<sub>E</sub>X is in this issue of *TUGboat*.)

### Lucida available again

In the previous installment of this column, I reported that TUG had negotiated the right to redistribute the Lucida fonts previously distributed by the (now sadly defunct) T<sub>E</sub>X vendor Y&Y. The technical update is complete, with thanks to Walter Schmidt and Morten Høgholm, and the distribution has been available at <http://tug.org/lucida> since early December, along with documentation, sample files, and macro support.

The Lucida support on CTAN has also been updated—it includes all the support files, lacking only the outlines, for the sake of those who previously acquired them from Y&Y. The fonts are also available through PCT<sub>E</sub>X, which sponsored Walter's work.

There is an active mailing list for users of Lucida and T<sub>E</sub>X; for archives and subscription information, please see <http://lists.tug.org/lucida>.

### TUG membership renewals

TUG membership and renewal forms (both online and printed) for 2006 are available at <http://tug.org/join.html>. Thanks to those who have already renewed; if you haven't, please do when you can. The basic membership structure remains the same, but there are a few changes this year. (Joint members, you will get more information from your local group later in the year as usual.)

Membership dues have increased in 2006. The last change in dues was in 1998 (and not by much), thus the TUG board approved a \$10 increase to offset some rising expenses and the slow decline in membership we have experienced the last few years. With your support, and additional outreach efforts, we're hoping to reverse the latter. Ideas for membership publicity and additional benefits are always welcome; please email [tug-membership@tug.org](mailto:tug-membership@tug.org).

At the request of several members, we've also instituted an option to automatically renew memberships for future years. This saves TUG a little overhead and you a little time, so please consider selecting it.

Another new option is to omit shipping of physical benefits (namely printed *TUGboat* issues and software discs), with a consequent reduction (of \$20) in the fee. Again, some members requested this, preferring to access everything online. As mentioned in the previous column and elsewhere, current *TUGboat* issues (beginning with this 2005 volume) are immediately available online to members only; see <http://tug.org/memberaccess.html>.

### Board resignation

In December 2005, Lance Carnes decided to resign from the TUG board. We thank him for the ideas he brought forth and the ensuing lively discussions, and all his efforts, past and continuing, on behalf of T<sub>E</sub>X and TUG. Happily, he continues as editor of TUG's online publication, *The PracT<sub>E</sub>X Journal* (<http://tug.org/pracjourn>).

A last bit of administrivia: the TUG fax number has changed: it's now +1 206-203-3960.

TUG is always striving to advance the use and understanding of (I<sub>A</sub>)T<sub>E</sub>X and friends in the wider community. If you have suggestions about how we can further these efforts, please don't hesitate to contact me or the entire board ([board@tug.org](mailto:board@tug.org)). Thanks for your support, and happy T<sub>E</sub>Xing.

◊ Karl Berry  
 president (at) tug.org

---

## Editorial Comments

Barbara Beeton

### Mimi Burbank retires

At the end of October 2005, Mimi said goodbye to Florida State University, where she had worked since the early 1980s, packed her bags, and headed for Kasese, Uganda. Mimi had arranged in late 1994 for a shared production area for *TUGboat* on one of the computers at the Supercomputer Computations Research Institute (SCRI, where Mimi was known as “*SCRIming Mimi*”), and she became our production manager effective with the first 1995 issue. Mimi had “gotten her feet wet” with *TUGboat* several years earlier by helping to edit the TUG’91 conference proceedings and assuming the job of Proceedings Editor for TUG’92 and ’93. When SCRI transformed itself into the Computer Science and Information Technology (CSIT) department, Mimi simply arranged for production to continue uninterrupted.

Mimi was very active in TUG almost from the day she joined. In addition to her devoted service to *TUGboat*, she served on the Board from 1994–1996, and was acting treasurer during the period when the TUG office was moving from Santa Barbara to Portland. She headed the organizing committee for TUG’95 in St. Petersburg, Florida, and was for several years thereafter active in the conference committee. For even more years, she was a member, and also served as chair, of the publications committee.

In preparation for her retirement, starting in the spring of 2005, Mimi helped the *TUGboat* production team clean off the disk we’d been using to prepare the camera copy and move everything to the TUG machine in Århus, Denmark. By the beginning of October, all the archives from CSIT had been safely transferred, and the Florida site was closed to TUG users.

Now Mimi has undertaken a new calling: she has joined the South Rwenzori Diocese of the Anglican Church of Uganda as a volunteer (she does not like to use the term “missionary”) in the diocesan offices under the guidance of Bishop Jackson Nzerebende Tembo. Her kindness and empathy for others will help her to serve well. One of her goals is to record the people’s stories, both traditional and those of contemporary life. She will be using her computer skills to build web pages for the diocese, to record for donors in the U.S. the activities of their adopted Ugandan parish—including the construction of a new roof for the cathedral. Her letters

show that, although life is quite different from what she has been used to, her spirit is unbowed, and she is making friends as she always does.

Mimi has become a very dear friend through our shared experiences with TUG. We traveled together to Russia in 1996 to attend the meeting in Dubna, and we’ve always “burned up” the Internet wires with e-mail. The e-mail continues, although with fewer demands for Mimi’s attention to *TUGboat* details, and with more ruminations and philosophy.

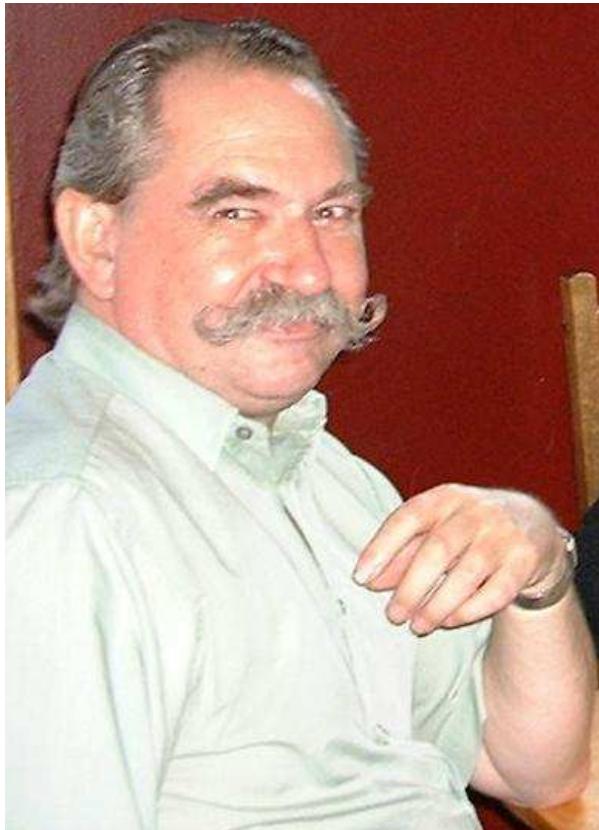
We wish Mimi the greatest happiness in her chosen pursuits, and look forward to future visits with her in person. The 2006 TUG Annual meeting is being held in Morocco—and she is in that part of the world, so we may see her at this year’s meeting!

### Brian {Hamilton Kelly}, 1945–2005

It was with great sadness that we learned that Brian died on 15 September 2005 after a brief struggle with leukemia. He was 60 in March, and was looking forward to retiring next August.

I first encountered Brian by way of a letter to the TUG office in 1986, in which he chided us, ever so gently, that listing his name in the TUG membership list under Kelly was simply not correct; just because it didn’t have a hyphen was no reason to misrepresent the “double-barrelled” name that had served his family well for many generations. Thereafter, he took to signing his surname, at least in e-mail, with the *TEX* grouping braces; some people on non-*TEX* discussion lists found this eccentric, but to *TEX*ies, it was perfectly clear why he adopted this notation.

Brian managed *TEX* support (and many other things) at the Royal Military College of Science in Shrivenham, England. His was one of the first sites to install *TEX* on a DEC VMS system, and (with his colleague Niel Kempson) was responsible for several important pieces of auxiliary software, including DVIToLN03. He was also responsible for some early *TEX* bug reports and suggestions for enhancements to *TEX* 3.0 (some of which finally made their way into *ε-TEX*). He was a member of the team that maintained the Aston archive (the forerunner of CTAN), for some years an active member of UK TUG, and the creator of macros for crossword puzzles (*TUGboat* 11:1 (1990), 103–119) and of one of the first Greek fonts based on Computer Modern. His experience with *TEX* on VMS was very helpful to us at AMS when we started to use that system. A message in my archive, from me to Don Knuth, reported on 7 November 1989 that



Brian {Hamilton Kelly}, summer 2004  
Photo courtesy of Jane Boulton

brian hamilton kelly, having taken the appearance of tex 2.992 as a challenge for a race, has announced that his installation under vms passed the trip test today at 1100 gmt. after you and perhaps a few diligent souls at stanford, this may be the first. thought you might like to know that some folks are listening to your plea to stamp out old versions.

I finally met Brian at the  $\text{\TeX}'90$  in Cork, where he presented the paper “Public-domain, documented implementations of  $\text{\TeX}$  and METAFONT for VAX/VMS” (*TUGboat* 12:1 (1991), 80–83). He lived up to his e-mail image of an intelligent and helpful person, very, very good at what he did. He was also distinct in his dapper appearance and bow-tie.

I learned only later of his many other, non- $\text{\TeX}$ -related interests, including early British telephony. He and his partner, Jane Boulton, were intending to get married after his retirement and see a lot more of the world. Jane reported to newsgroups to which Brian had contributed that “His ambition was to be shot by a jealous husband at the age of 100.” Sadly, he will never have that chance.

Brian will be missed.

## Erratum for *TUGboat* 25:2

The article by Steve Peter on ConTEXt (pages 128–130) in the subject issue unfortunately carried an incorrect volume number in the headline—volume 26. It should have been volume 25. We regret the confusion. The error has been corrected in the online PDF file.

## E-mail addresses in *TUGboat* on line

In response to concern from several authors, and our own annoyance at the increasing piles of spam stuffing our electronic in-boxes, the *TUGboat* production team has decided to change the way in which author e-mail addresses are presented. Beginning with this issue, the @ sign will be replaced with (at). This may not stop web address harvesters entirely, but it should slow them down.

Since our policy is that authors hold copyright to their works that appear in *TUGboat*, we feel it is important that others wishing to communicate with an author can find the necessary information within the article. Thus, we don’t wish to remove address information altogether, but if we can make it harder for addresses to be misused, that’s a worthy goal.

## The *TUGboat* schedule

This is the last issue of *TUGboat* for 2005. Although the calendar now says February 2006, we are close to being caught up. 2006 should see the schedule back on track—we expect to publish three issues of conference proceedings: EuroTEX’06, Practical  $\text{\TeX}'06$ , and TUG’06, as well as regular issue material. More information about these conferences is elsewhere in this issue (see the calendar for submission deadlines), as well as linked from the TUG home page. We look forward to seeing both familiar and new faces at these events.

## Lucida Bright fonts now available from TUG

It has already been announced to TUG members that the Lucida Bright fonts, by Bigelow & Holmes, are now available from TUG. Formerly available only from Y&Y, these fonts became unavailable when Y&Y ceased operations last year. We are delighted to be able to make them available again.

The TUG web site (<http://tug.org/lucida>) carries quite a bit of information about the fonts, including notes by Chuck Bigelow about the design. That is what I’d like to expand upon—some history that didn’t get into Chuck’s notes.

During the early development of Lucida, Chuck also designed a related family of fonts intended for screen use at very low resolution. These fonts, named

*Pellucida*, were created directly as bitmaps. They were used in DEC VAXstations, Tektronix Smalltalk workstations, proprietary workstations at Bell Labs, and other raster display devices in the early and mid 1980s.

Chuck has provided the following information about *Pellucida* and some other rasterized fonts.

I wrote an article about this for the Gutenberg Jahrbuch, with some illustrations, back in 1986 or so. Also a short article for *BYTE* magazine.

*Pellucida* fonts were distinguished from *Lucida* because the former were bitmapped and the latter scalable. Since we hand-tuned the bitmap fonts, many of them didn't exactly correspond to the scalable ones, hence the name difference. The fact was, and is, that low-res bitmap raster fonts never correspond exactly to the hi-res outline fonts from which they are supposedly derived. There is a great deal of "impressionism" and "pointilism" in which the hand bitmap editor makes things that sorta kinda suggest what the hi-res font would look like if there were enough resolution to render it, which of course there isn't.

During Adobe's brief near-monopoly on scalable fonts, they established the custom of naming bitmap fonts with the same names as the scalable fonts from which they were derived, even if the bitmaps were heavily hand-edited. The same trend continued with TrueType, which permitted more hand-tuning of hints of scalable fonts in order to coerce more pleasing bit patterns at specific sizes and resolutions. After the industry shifted to scalable fonts, we stopped making or marketing the *Pellucida* fonts.

Later, in a paper for *Electronic Publishing* in the early 90s, we described our ad-hoc process for creating hi-res fonts from low-res bitmaps—specifically our design of TrueType fonts for Apple based on the the bitmap fonts Apple had created for the first Macintosh. Chicago was the most geometrically accurate: our TrueType font rasterized bit-for-bit like the original bitmap font at 14 point, even without hints. We constructed it entirely from arcs and line segments. On the other hand, our TT version of Monaco was hand-drawn and looks almost nothing like the original bitmaps. But our novel treatment

of the 'i' and 'l' in Monaco became widely imitated by later designers. Ideas too cool to be restricted to one font. :-)

In addition to the publications listed by Chuck in the *Lucida* notes, the *Notices of the American Mathematical Society* have been set in *Lucida Bright* since 1995. This publication—designed and laid out more like a magazine than an academic journal—is prepared in Quark, with math inclusions inserted using Blue Sky's MathSetter tool. We find that *Lucida* gives the pages a less rigid, rather informal appearance, compared to *Computer Modern*, an appearance appropriate for the material contained in the publication.

It's good to have *Lucida Bright* available again.

### Knuth on NPR

On 14 March 2005, I turned on my radio to hear Don Knuth being interviewed on NPR. The interview, "Donald Knuth, Founding Artist of Computer Science", by David Kestenbaum, can be heard on line at the NPR web site: <http://www.npr.org/templates/story/story.php?storyID=4532247>.

We explored the possibility of obtaining permission to publish the transcript in *TUGboat*; alas, the cost was too high to permit both paper and on-line publication, and would need to be renewed annually (at extra cost) for the on-line version. We'll put a link with the contents list for this issue so that you can reach the NPR site easily.

### Letters in stone

A short animated movie, *Etched in Stone*, follows a reporter who investigates the murders of several film directors in a mystery that depends on identification of a typeface. See it at <http://www.veer.com/ideas/etched/>. (Viewing requires QuickTime.)

Another web site with the theme of letters in stone is <http://typolapidaire.free.fr/>.

One more: "Can I Carve That in Stone?", at <http://www.signweb.com/dimensional/cont/carveinstoneb.htm>, has some guidelines for sandblasting letters into stone by John Benson, stone-carver extraordinaire. If you ever have a chance to attend one of his illustrated lectures, don't miss it!

◊ Barbara Beeton  
American Mathematical Society  
201 Charles Street  
Providence, RI 02904 USA  
bnb (at) ams.org

## Interview with Donald E. Knuth

Gianluca Pignalberi

### Abstract

A prime number of questions to the Professor Emeritus of the Art of Computer Programming.

### Introduction

The typesetting of Free Software Magazine is entirely *T<sub>E</sub>X*-based, so that process uses the program that Prof. Donald Knuth designed some 30 years ago. Since then the *T<sub>E</sub>X* project has generated many offshoots (i.e., *L<sup>A</sup>T<sub>E</sub>X*, *ConT<sub>E</sub>Xt*,  $\Omega$ , and others).

In May 2005 I had the chance and the honor to interrupt Prof. Knuth's activity and interview him for Free Software Magazine. I'm proud, as a journalist and FSM *T<sub>E</sub>Xnician*, to see it republished in *TUGboat*.

*GP: Donald E. Knuth, Professor Emeritus of the Art of Computer Programming, Professor of (Concrete) Mathematics, creator of *T<sub>E</sub>X* and *METAFONT*, author of several fantastic books (such as *Computers & Typesetting*, *The Art of Computer Programming*, *Concrete Mathematics*) and articles, recipient of the Turing Award, Kyoto Prize, and other important awards; fellow of the Royal Society (I could keep going). Is there anything you feel you have wanted to master and haven't? If so, why?*

*DEK: Thanks for your kind words, but really I'm constantly trying to learn new things in hopes that I can then help teach them to others. I also wish I was able to understand non-English languages without so much difficulty; I'm often limited by my linguistic incompetence, and I want to understand people from other cultures and other eras.*

*GP: Your algorithms are well known and well documented (I'll only quote, for brevity's sake, the Knuth-Morris-Pratt String Matching algorithm), which allows everyone to use, study and improve upon them freely. If it wasn't clear through your actions, in an interview with Dr. Dobb's Journal, you stated your opinion about software patents, which are forcing people to pay fees if they either want to use or modify patented algorithms. Has your opinion on software patents changed or strengthened? If so, how? And how do you view the EU Parliament's wishes to adopt software patent laws?*

---

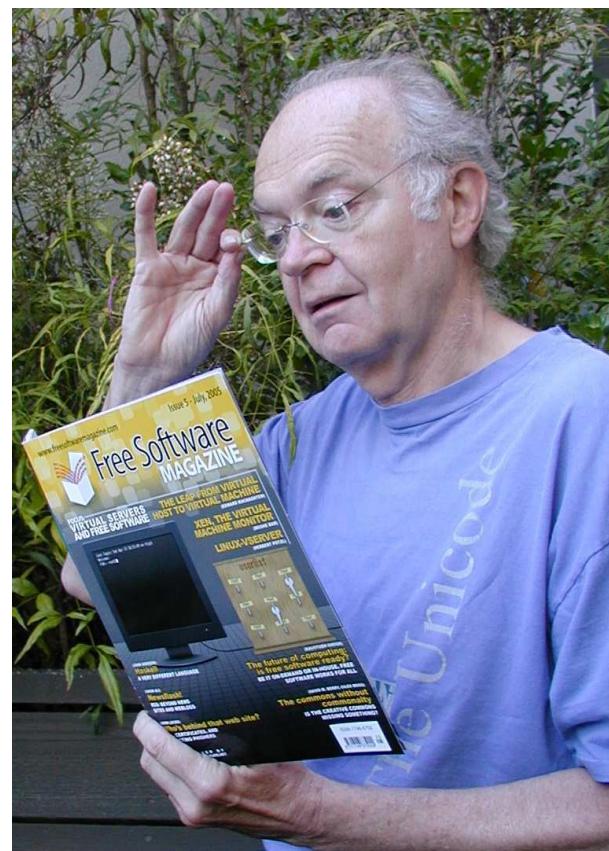
*Editor's note:* This article was originally published in *The Free Software Magazine*, issue 7 (2005), and is reprinted by kind permission of the interviewer and editor.

*DEK: I mention patents in several parts of *The Art of Computer Programming*. For example, when discussing one of the first sorting methods to be patented, I say this:*

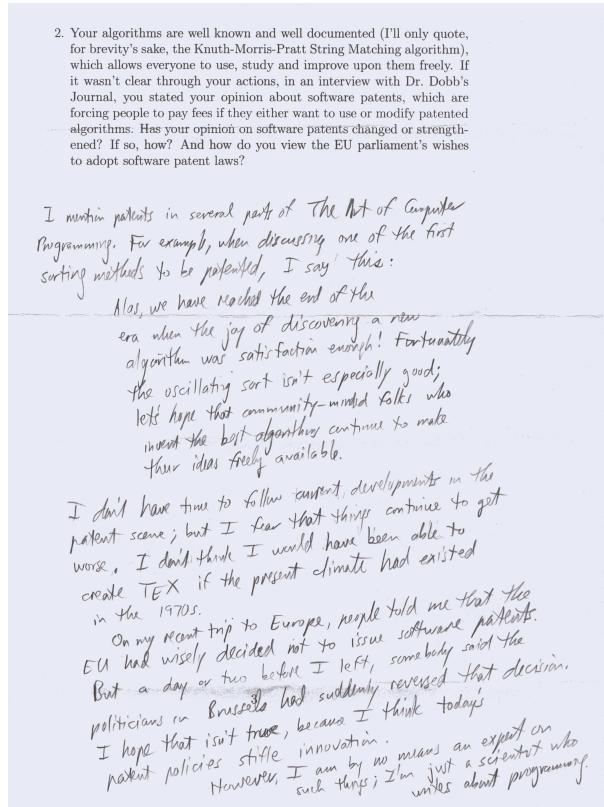
Alas, we have reached the end of the era when the joy of discovering a new algorithm was satisfaction enough! Fortunately the oscillating sort isn't especially good; let's hope that community-minded folks who invent the best algorithms continue to make their ideas freely available.

I don't have time to follow current developments in the patent scene; but I fear that things continue to get worse. I don't think I would have been able to create *T<sub>E</sub>X* if the present climate had existed in the 1970s.

On my recent trip to Europe, people told me that the EU had wisely decided not to issue software patents. But a day or two before I left, somebody said the politicians in Brussels had suddenly



**Figure 1:** Prof. Knuth while reading one of the magazines typeset by his program *T<sub>E</sub>X*. Photo by Jill Knuth (a graduate of Flora Stone Mather College — another FSM).



**Figure 2:** The portion of the paper where Knuth answered question 2 of this interview. Please note that he typographically quotes and typesets even when writing by hand: the quotation from TAOCP and the word *TeX*.

reversed that decision. I hope that isn't true, because I think today's patent policies stifle innovation.

However, I am by no means an expert on such things; I'm just a scientist who writes about programming.

GP: So far you have written three volumes of *The Art of Computer Programming*, are working on the fourth, are hoping to finish the fifth volume by 2010, and still plan to write volumes six and seven. Apart from the Selected papers series, are there any other topics you feel you should write essays on, but do not have time for? If so, can you summarize what subject you would write on?

DEK: I'm making slow but steady progress on volumes 4 and 5. I also have many notes for volumes 6 and 7, but those books deal with less fundamental topics and I might find that there is little need for those books when I get to that point.

I fear about 20 years of work are needed before I can bring TAOCP to a successful conclusion; and I'm 67 years old now; so I fondly hope that I will remain healthy and able to do a good job even as I get older and more decrepit. Thankfully, at the moment I feel as good as ever.

If I have time for anything else I would like to compose some music. Of course I don't know if that would be successful; I would keep it to myself if it isn't any good. Still, I get an urge every once in awhile to try, and computers are making such things easier.

GP: There are rumours that you started the *TeX* project because you were tired of seeing your manuscripts mistreated by the American Mathematical Society. At the same time, you stated that you created it after seeing the proofs of your book *The Art of Computer Programming*. Please, tell our readers briefly what made you decide to start the project, which tools you used, and how many people you had at the core of the *TeX* team.

DEK: No, the math societies weren't to blame for the sorry state of typography in 1975. It was the fact that the printing industry had switched to new methods, and the new methods were designed to be fine for magazines and newspapers and novels but not for science. Scientists didn't have any economic clout, so nobody cared if our books and papers looked good or bad.

I tell the whole story in Chapter 1 of my book *Digital Typography*, which of course is a book I hope everybody will read and enjoy.

The tools I used were home grown and became known as Literate Programming. I am enormously biased about Literate Programming, which is surely the greatest thing since sliced bread. I continue to use it to write programs almost every day, and it helps me get efficient, robust, maintainable code much more successfully than any other way I know. Of course, I realize that other people might find other approaches more to their liking; but wow, I love the tools I've got now. I couldn't have written difficult programs like the MMIX meta-simulator at all if I hadn't had Literate Programming; the task would have been too difficult.

At the core of the *TeX* team I had assistants who read the code I wrote, and who prepared printer drivers and interfaces and ported to other systems. I had two students who invented algorithms for hyphenation and line breaking. And I had many dozens of volunteers who met every Friday for several hours to help me make decisions. But I wrote every line of code myself.

Chapter 10 of my book *Literate Programming* explains why I think a first-generation project like this would have flopped if I had tried to delegate the work.

GP: Maybe you feel that some of today's technologies are still unsatisfactory. If you weren't busy writing your masterpieces, what technology would you try to revolutionize and in what way?

DEK: Well, certainly I would try to work for world peace and justice. I tend to think of myself as a citizen of the world; I am pleasantly excited when I see the world getting smaller and people of different cultures working together and respecting their differences. Conversely I am distressed when I learn about deep-seated hatred or when I see people exploiting others or shoving them around pre-emptively.

In what way could the desired revolution come about? Who knows ... but I suspect that "Engineers Without Borders" are closer than anybody else to a working strategy by which geeks like me could help.

GP: Thank you again for your precious time.

DEK: Thank you for posing excellent questions!

#### Copyright information

© 2005 Gianluca Pignalberi

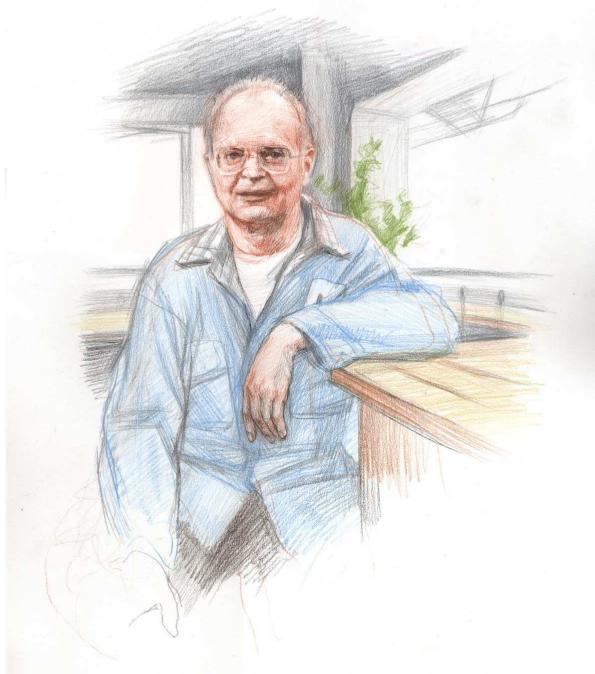
This article is made available under the

"Attribution-NonCommercial-NoDerivs"

Creative Commons License 2.0, available from

<http://creativecommons.org/licenses/by-nc-nd/2.0/>.

- ◊ Gianluca Pignalberi  
 Via del Pozzo, 34  
 04010 Cori (LT)  
 Italy  
 g.pignalberi (at) freesoftwaremagazine.  
 com  
<http://www.freesoftwaremagazine.com/>  
 contacts



**Figure 3:** Portrait of Donald E. Knuth by Alexandra Drofina. Commercial users should write to Yura Revich ([revich@computerra.ru](mailto:revich@computerra.ru)) for permission.

## Implementing editors' ideas — lots of fun, sometimes even more trouble

Andrzej Tomaszewski

### Abstract

Almost every one among us works with texts and prepares publications for print. We have to manipulate materials from various sources. We are bound to work with different authors, editors and decision makers. Sometimes they are great professionals but not seldom they fail to have even basic knowledge of printing techniques. In this article I will present typical problems and attempts to overcome them.

Seasoned editors are becoming a rarity. Only with difficulty, usually in renowned publishing houses, can one meet editors who learned their trade before the time of computerization. Solicitude, purpose, and the beauty of the Polish language are ingrained in them. The slowly dwindling generation of aged bookmakers knows inside-out text typesetting rules and grammar nuances. Their alert eye will spot any lax language usage or inconsequential notation. Sometimes they are rewarded with a good word from authors who notice that their text becomes smoother and more communicative during revision. This of course applies only to wise authors. There exist also unwise ones—often professorial—with such a swelled head that it leads them to act against their own interest during the preparation of their publication. Almost all seasoned editors think about the future reader during their work, they understand the educational or informative function of the book and treat their work almost as a mission.

Cooperating with such editors is a real pleasure though there might be a darker side to it. One of the most important is that they lack the knowledge of the contemporary publishing technology. They are unable to use even simple programs for entering and manipulating texts. They have a false notion that they are unable to acquire such abilities. It also happens that they are emotionally opposed to computers, which they regard as evil. Then—psychologically blocked—they do not accept (or even will not listen to) even the simplest explanations regard-

---

Editor's note: This is a translation of the article "Sto pociech i dwieście utrapień z realizacją pomysłów redaktora", which was first presented at BachoTEX 2005 and published in *Biuletyn Polskiej Grupy Użytkowników Systemu TeX*, Zeszyt 22, May 2005, T. Przechlewski, B. Lichński, S. Wawrykiewicz, P. Bolek, eds. Reprinted with permission. English translation by Jerzy Ludwichowski.

ing the necessity to submit to some contemporary rules.

Here is an example: Suppose you convinced a person who was for many years using a typewriter to use a computer keyboard. It turns out that the person types quickly and ably but does not know about the spectrum of available characters, not to mention how to enter the needed one on the new keyboard.

Old typewriters had a very limited number of available characters. In almost all models it was possible to insert needed characters in place of those which were unused. For example, instead of '1' (one), one could often type lower case 'l' or even 'I'; with uppercase 'O' present, '0' (zero) was replaced with '+' or '%'. Such manipulations were supported by the similarity of some shapes of typewriter characters. Typesetters setting the texts in the printing house at lino-types or mono-types were not disturbed by such character replacements as they were reading texts "contextually" and knew well which characters to use. Firmly ingrained typewriter habits are very difficult to eradicate and in the resulting texts one has to hunt meticulously for some characters and digits, differentiate dashes (minuses, en-dashes, em-dashes), delete unnecessary "enters", substitute paragraph indentations made with several spaces, as well as other things resulting from the typewriter technique. These additional actions, often labour consuming and troublesome, could be avoided by the author or editor entering the text being more disciplined. This is important, as a typesetter poking about too much in the text increases the chance of introducing additional errors.

It is easy to stumble onto a different type of editor. He acquired some patchy knowledge about typesetting using computers and is firmly convinced that this knowledge is of the finest quality. Quite often this person is very good at his trade, is familiar with the subject of the book, initiates a good cooperation with the author, and collects materials for the publication. His ambition however is to record everything electronically and ... if this ambition is restricted only to texts, then not much trouble ensues. Problems begin when the need arises to select and collect photographs, graphics, plots or other elements for an illustrated book.

Having access to an office scanner, Microsoft Office and the Internet, the editor tries to substitute for the typographer, graphics designer, DTP operator or other specialists preparing the publication for print. He scans photographs losing half-tones, crops them not knowing where on the page

they will be placed, copies from the Internet unusable GIF and JPEG files, produces nightmare graphs from spreadsheets, awkwardly tries to copy materials from prints, and so on and so forth. Most of the material is unsuitable for reproduction or further processing, and every critical remark on this subject is taken by the editor as an insult or questioning of his competence.

Often, however, the editor is as innocent as a newborn child. He was given the materials by the author. The author, a renowned specialist in his discipline, collected the materials over several years. With the help of his friends and family members, step by step, he prepared his illustrations of various quality and provenance, pasted them into his Word document, and finally deleted the now “obsolete” original files to clean up the mess on his hard disk. They are already in Word!

Who is going to tell the respectable author that what he prepared is only good for printing coarse handouts but not as his *magnum opus*?

Preparation of illustrations is perhaps the single biggest problem in the collaboration with the publishers and their editors, but the biggest of the biggest is the ubiquitous mania for JPEG-ing everything in sight. Evangelisation directed at publishers by experienced graphics designers does not help. It does not help to show examples of irreversibly destroyed drawing edges, spotty backgrounds, washed-out faces, clouds in the sky resembling dirty snow, and similar effects of compressing pixel graphics. Editors in publishing houses know that a TIFF has to be JPEG-ed — to weigh as little as possible.

This is a true story from my experience. At one of the publishing houses a Polish language edition of a book well known in Europe was being prepared. The foreign publisher sent four CDs with illustrations. When and how were these CDs turned into 100 MB of RGB JPEG files? Nobody pled guilty. The original CDs of course vanished into thin air. When I declined to work on the book a general aura of astonishment and distaste ensued. But the

pictures look so good on the screen! And Mrs. Director says that the colors are better than in the foreign edition! Fortunately Mrs. Director’s husband said that if Tomaszewski doesn’t know how, then he himself will try to merge the illustrations with the text. And so the day was thus saved.

When doing contract work for a renowned higher education institution I have to work with a proofreader who ennobles texts with precise punctuation. He has so well mastered the Polish language that at one point in time he won a national spell-checking contest and now walks in well-deserved glory. So what about him? Well, there is a problem. Like all ambitious and scrupulous people he always and everywhere thinks about work. Even when falling asleep he comes up with better grammatical constructs, brilliant figures of speech or the way to spell a word of foreign origin. So he jumps up to the keyboard where he amends and refines the text which I have already had on my machine for several days, since it is a part of an important publication which I am sculpturing for print. For sure, a day before the deadline I will receive a diskette or an email with a Word file with the refined text as a replacement.

So what can you do? Phone calls, email messages, SMS-es — for three years nothing worked until I met him at a conference. In the evening, over beer, I explained to him distinctly and picturesquely what is wrong. In the morning he swore over scrambled eggs (with onions) that from now on he will do his marking only on paper. On galley proofs, which—as God commanded—should circulate between the editors and typesetters.

So much for now.

*Be it known that any resemblance of persons and situations described herein to real persons and situations is coincidental. The author does not assume responsibility for the reader’s delusions and associations.*

◊ Andrzej Tomaszewski  
Warsaw, Poland

---

**Minutes in less than hours:  
Using L<sup>A</sup>T<sub>E</sub>X resources**

Jim Hefferon

**Abstract**

To illustrate how to build a new L<sup>A</sup>T<sub>E</sub>X document class, we develop a class for minutes of meetings.

**1 What I needed**

Having a reputation of knowing T<sub>E</sub>X is not like being maitre d' at Le Cirque, exactly, but it does get me phone calls from people that I have never met, and who want me to do something for them. Telling them that I am not an expert T<sub>E</sub>X coder does not dissuade them. In the end I can usually help, but not with tricky macros. Instead, I show them how to break their job into parts for which there is an existing solution. To explain what this means in practice, I'll walk through the steps that I took recently to develop a document class.

I was asked to keep minutes for a committee, and given some samples from the prior year as models. The layout of these models was simple. Each had an opening and a body. Each opening had two parts. The first part was a document title giving the committee name and the date. The second part was a header listing who was at the meeting and who was not. (The last page of this article has an example.)

In the body of the samples were a number of environments. The main one was *Business*, a list of items that the committee took up on that day, which looked like a L<sup>A</sup>T<sub>E</sub>X `enumerate` list. There were also some similar environments, including *Old Business* and *Announcements*. Finally, each sample body contained a few more sections, such as *Next Meeting*, that were not lists.

**2 First, hit CTAN**

The place to look for solutions to T<sub>E</sub>X problems is the Comprehensive T<sub>E</sub>X Archive Network (CTAN).<sup>1</sup> So I went to <http://www.ctan.org/search.html> and submitted `minutes` and a few similar phrases. I got a number of responses but after browsing I found that none of them met my needs.

Like most people, I use L<sup>A</sup>T<sub>E</sub>X. So I decided to write a L<sup>A</sup>T<sub>E</sub>X class `mins.cls`.

---

Editor's note: This article was originally published in *The PracT<sub>E</sub>X Journal*, issue 2005-4, <http://tug.org/pracjourn>. Reprinted, with additions, by permission.

<sup>1</sup> Full disclosure: I run a node of CTAN.

**3 Second, hit the books**

There are many fine books on L<sup>A</sup>T<sub>E</sub>X but I happen to rely on Lamport's *L<sup>A</sup>T<sub>E</sub>X: A Users Guide and Reference Manual* and Mittelbach et al.'s *L<sup>A</sup>T<sub>E</sub>X Companion*.

**4 The class framework**

The *Companion* describes how to make a L<sup>A</sup>T<sub>E</sub>X class file, in particular in its Figure A.1. For a slight modification of that code see Section 4.2 below. Since I planned to work by cribbing all that I could, the most important line says `\LoadClass{article}`. This started my class off with all of the features of Lamport's *article* class, and from there I just needed to tweak a few behaviors.

**4.1 Class options**

The only problem that I had with the *Companion*'s Figure A.1 involved handling class options.

I wanted the flexibility to have my source files contain class options, as here

```
\documentclass[11pt]{mins}
```

where the option calls for 11 point type. The *Companion* explains how to do this; before the `\LoadClass` command, include this line.

```
\DeclareOption*{\PassOptionsToClass
               {\CurrentOption}{article}}
```

But I wanted option handling that was even fancier. Minutes have text that repeats from meeting to meeting, e.g., the names of members. So I wanted the ability to have an extra file that could contain a line like `\setmembers{A~Baker, ...}`. And I wanted, supposing this extra file is named `ttm.min`, that using the document option `ttm` would cause my class to input the file.

The *Companion* explains this also: I changed the body of the `\DeclareOption*` command to use `\InputIfFileExists`. This command has three arguments: if it finds a file with the name given in the first argument then it reads the file contents and runs the code given as the second argument, otherwise it runs the code given as the third argument.

In summary, if my class contains the line

```
\DeclareOption*{\InputIfFileExists
               {\CurrentOption.min}
               {}
               {\PassOptionsToClass{\CurrentOption}
                {article}}}
```

then the options in this first line of L<sup>A</sup>T<sub>E</sub>X source

```
\documentclass[11pt, ttm]{mins}
```

will be handled as: for `11pt` it finds no file `11pt.min` and so it passes the option to the `article` class, and for `ttm` it finds the file `ttm.min` and loads it.

## 4.2 Class code

Here is my class code, adapted from Figure A.1 in the *Companion* (some lines have been edited for presentation). The opening part identifies the class; this is good to have in the log file.

```
% mins.cls
% Take minutes of meetings
% 2005-Sept-01 Jim Hefferon

% --- Class structure: identification part
% ---
\ProvidesClass{mins}[2005/09/01 v1.00 ...]
\NeedsTeXFormat{LaTeX2e}

% --- Class structure: declaration of options
% ---
% This class extends the article class
% Read the documentclass options and pass
% them to article, unless the file
% "<currentoption>.min" exists, then load it
\DeclareOption*{\InputIfFileExists
  {\CurrentOption.min}
  {}}
\PassOptionsToClass{\CurrentOption}{article}

% --- Class structure: execution of options
% ---
\ProcessOptions \relax

% --- Class structure: package loading
% ---
\LoadClass{article}
```

So, with the *Companion*'s help, I had the basic structure of my L<sup>A</sup>T<sub>E</sub>X class.

## 5 Page layout

I next needed to set the page size and to have appropriate headers and footers. Both of these are things that authors need to do all the time, so you might expect that to accomplish these jobs there are packages that are both powerful and easy. You'd be right.

### 5.1 Page size

To set a L<sup>A</sup>T<sub>E</sub>X page size, use the *geometry* package.<sup>2</sup> The *Companion* describes this package, and based on that I included this line.

```
% Page layout
\RequirePackage[left=1in,right=1in,
               top=1in,bottom=1in]{geometry}
```

Some people like the left and right margins to be bigger, making shorter lines, for increased readability. But I decided to save paper by making the margin small—no one reads minutes, anyway!

### 5.2 Headers and footers

As with page dimensions, there is a canonical package for headers and footers, *fancyhdr*.<sup>3</sup> You can set six fields on each page: the left, right, and center of each of the head and foot. The code below blanks the headers and footers of the first page, and on the following pages sets the headers on the right side of the even-numbered pages to be the same as the headers on the left side of the odd-numbered pages (both are the committee's name followed by the date).

```
\RequirePackage{fancyhdr}
\fancypagestyle{firstpage}{%
  \fancyhf{} % clear all six fields
  \renewcommand{\headrulewidth}{0pt}
  \renewcommand{\footrulewidth}{0pt}
}
\fancypagestyle{followingpage}{%
  \fancyhf{} % clear all six fields
  \fancyhead[RE,LO]{\show@committee, \show@date}
  \fancyhead[LE,RO]{page \thepage}
  \renewcommand{\headrulewidth}{0.7pt}
  \renewcommand{\footrulewidth}{0pt}
}
\pagestyle{followingpage}
\AtBeginDocument{\thispagestyle{firstpage}}
```

The `\headrulewidth` and `\footrulewidth` need some explaining. By default, *fancyhdr* puts a horizontal line (a rule) across the top and bottom of the page, whose thickness is given by the command. Setting the rule to a thickness of zero points makes it disappear.

---

<sup>2</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/geometry>

<sup>3</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/fancyhdr>

## 6 Code

At this point, I was stuck. I had cribbed all that I could and I finally had to write some code of my own.

### 6.1 Definitions of lists

First I thought to define the membership of the committee, to go in the extra file. This is the contents of the file `ttm.min`.

```
\setcommittee{Totally Trivial Matters Committee}
\setmembers{\secretary{A~Bravo},
\role{C~Delta}{President, \textit{ex officio}},
E~Foxtrot, \chair{G~Hotel}, I~Juliet, K~Lima,
M~November, O~Papa}
```

That's when I realized why the L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  class structure contains the initial code part that I passed over in Section 4.2. To have the `\setcommittee` and `\setmembers` commands available, I needed to define them before `\InputIfFileExists` reads in the file. Ah, I get it!

Luckily, I was familiar with the technique needed to define these two commands. Below, the second line has `\setcommittee` save the list as `\@committee` (the at-sign is a L<sup>A</sup>T<sub>E</sub>X convention to keep ordinary users from using the same name). Its matching command `\show@committee` produces the list.

```
% what is meeting?
\def\@committee{}
\newcommand{\setcommittee}[1]{%
\def\@committee{#1}}
\newcommand{\show@committee}{%
\@committee}

% who is meeting?
\def\@members{None}
\newcommand{\setmembers}[1]{\def\@members{#1}}
\newcommand{\show@members}{\@members}
```

I use the same technique for some similar functions.

```
% who is absent?
\global\let\@absent\@empty
\newcommand{\setabsent}[1]{\def\@absent{#1}}
\let\absent\setabsent %
\newcommand{\show@absent}{\@absent}

% who is also present?
\global\let\@alsopresent\@empty
\newcommand{\setalsopresent}[1]
{\def\@alsopresent{#1}}
\let\alsopresent\setalsopresent %
\newcommand{\show@alsopresent}{\@alsopresent}

% what day is it?
```

```
\def\@date{\today}
\newcommand{\setdate}[1]{\def\@date{#1}}
\newcommand{\show@date}{\@date}
```

I also wanted to define a standard way of referring to the committee chair, etc.

```
% what role do they have (e.g., chair)
\newcommand{\role}[2]{\#1~(\#2)}
\newcommand{\chair}[1]{\role{#1}{Chair}}
\newcommand{\secretary}[1]{\role{#1}{Secretary}}
```

### 6.2 Document body

As I've mentioned, the main part of the sample documents that I was given consisted of an enumeration list *Business* and there were a number of similar lists. I decided to make a single environment, which I could specialize from list to list.

```
% environments inside the minutes
\newenvironment{businesslist}[1]{%
\vspace{2ex}\par\noindent\textbf{#1}\par
\begin{enumerate}}
\end{enumerate}
\newenvironment{business}{%
\begin{businesslist}{Business}
\item
\end{businesslist}}
```

This simply prints “Business,” puts a bit of vertical space, and makes a list (it does not, as shown, suppress a page break). I added similar environments for *Old Business*, *Future Business*, and *Announcements*.

### 6.3 Document opening

This was the only part of the class that gave me any trouble. I expected that document source files would be structured like this.

```
\documentclass[11pt, ttm]{mins}
\setabsent{J~Hef{}feron}
\setdate{2005-Sept-01}
\begin{document}
\begin{minutes}
... stuff like the business environment ...
\end{minutes}
\end{document}
```

Thus, the `mins` environment should produce both the part naming the committee, and the part listing the committee members, etc.

```
% basic definition of the minutes environ
\newenvironment{minutes}{%
\begin{center}
{\large\textrm{bf}{Minutes, \show@committee}}
\\[1ex]
\show@date
\end{center}
\vspace{1.5ex}
\opening@list
\vspace{1ex}
}%
}
```

Here was my first try at the opening's listing.

```
% material heading the minutes;
% unfinished version
\newcommand{\opening@list}{%
\begin{description}
\item[Members:] \show@members
\item[Absent:] \show@absent
\item[Also present:] \show@alsopresent
\end{description}
}
```

That worked, but—the bane of all software—I decided to add a feature. I wanted that if no one was absent then the `\item[Absent:] \show@absent` line would be left out.

This is a question of finding the right kind of `if` statement. I struggled with it, but some spelunking on the Internet and in *The TeXbook* yielded the magic incantation.

```
% material heading the minutes; final version
\newcommand{\opening@list}{%
\begin{description}
\item[Members:] \show@members
\ifx@\absent@\empty
\relax
\else
\item[Absent:] \show@absent
\fi
\ifx@\alsopresent@\empty
\relax
\else
\item[Also present:] \show@alsopresent
\fi
\end{description}
}
```

## 7 Conclusion

I have seen on the Internet (credited to a number of different people) these Laws of Program Writing.

The First Law is: don't. Instead, see if someone else has already written a version of the

program that you can crib. If not, see if someone has written a program that is like what you need and that you can adapt.

The Second Law is: if, truly, no one has ever written a program anything like what you need, and you simply must write it fresh, then put a lot of effort into it, so your program can be cribbed by people following the First Law.

Like many jokes, there is some truth in this, and I have tried here to show how to follow it in a *TeX* context.

The result is that for this project I did very little work. Most of my class's functionality is inherited from Lamport's *article*. Of what I changed, customizing the page size and the headers and footers just involved looking up the right tools in the *Companion*. The only real work that I did was in defining the `minutes` environment. Consequently, the total time I spent on the class was only about three hours, and I ended with a usable, and reusable, piece of software (and this nice article).

I sometimes suspect, when I respond to people who call me with *TeX* problems, that my advice might be not entirely welcome. Sure, it solves the problem that they said they had, but I wonder: maybe they are not really glad to get my advice, maybe they are having fun playing with *TeX* and now they have to go back to writing!

You know, I hardly ever get two calls from the same person . . . .

## 8 Exercises

1. Have the opening text “Members”, “Absent”, etc., come out in small caps. (*Answer:* Add  
`\renewcommand{\descriptionlabel}[1]{%`  
`\hspace{\labelsep}\textsc{##1}}`  
between the `\newcommand{\opening@list}{` line and the `\begin{description}` line.)
2. Make a simple memo class. Put your organization's logo on the first page. (*Hint:* in the header of your first page, use *LATEX*'s *picture* environment to place your graphic. Look up how to adjust the header height.)

## 9 Example output

The next page shows a sample two-page minutes document, along with its *LATEX* source. (It uses the *lipsum* package to generate text.)

◊ Jim Hefferon  
St. Michael's College  
Vermont, USA  
`ftpmaint (at) alan.smcvt.edu`

## Minutes, Totally Trivial Matters Committee

1958-Oct-12

**Members:** A Bravo (Secretary), C Delta (President, *ex officio*), E Foxtrot, G Hotel (Chair), I Juliet, K Lima, M November, O Papa

**Absent:** I Juliet

## Announcements

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
2. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.
3. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

## Business

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

```
\documentclass[11pt,ttm,twoside]{mins}
\usepackage{lipsum} % produces dummy text

\setdate{1958-Oct-12}
\absent{I^Juliet}
\alsopresent{}

% file ttm.min says:
%\setcommittee{Totally Trivial Matters Committee}
%\setmember{\secretary{A^Bravo},
% \role{C^Delta}{President, \textit{ex officio}}, E^Foxtrot,
% \chair{G^Hotel}, I^Juliet, K^Lima, M^November, O^Papa}

\begin{document}
\begin{minutes}
\begin{announcements}
\item
\lipsum[1]

\item
\lipsum[2]

\item
\lipsum[3]
\end{announcements}
\end{minutes}
\end{document}
```

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

3. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas duí, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.
4. Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut duí. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc duí lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubias nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetur at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, duí. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, duí. Mauris nibh leo, facilisis non, adipiscing quis, ultricies a, duí.

## New Business

1. Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget duí varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justa a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

**Next Meeting:** Monday, Oct 19, at 11:30.

```
\begin{business}
\item
\priormins

\item
\lipsum[4-5]

\item
\lipsum[6]

\item
\lipsum[7-8]
\end{business}

\begin{newbusiness}
\item
\lipsum[9]
\end{newbusiness}

\nextmeeting{Monday, Oct 19, at 11:30.}

\end{minutes}
\end{document}
```

## \starttext: Swelled rules and MetaPost

Steve Peter

**Abstract**

A swelled rule has tapered edges and a thicker mid-section, thus blending better with the overall color of a page than a normal rule. In this column, we explore methods for creating them, using ConTeXt and MetaPost.

**1 Introduction**

Recently I was enjoying Ari Rafaeli's *Book Typography* and noted that he writes (page 93), "Swelled rules, adopted by English printers in the late eighteenth century, out of fashion in the early twentieth century, but revived by Stanley Morison and None-such and The Curwen Press, are sadly missing from the typography of contemporary books." So I made a note to myself: use a swelled rule (also known as an English rule), in an upcoming project.

Coincidentally I read the column by Dave Walden in the 2005-4 issue of *The PracTeX Journal*, where he talks about thought groups. He discusses various ways to break text at a point where the thought transitions somewhat, but not enough for a section break. One traditional printer's technique for that is to use spaced out asterisks, as here.

\* \* \*

Walden also suggests a horizontal rule. A swelled rule, by contrast, has a bit more presence on the page than a simple rule, but the tapered edges allow it to blend in with the overall color of the page. (Typographical color, that is, which comes down to how gray the page appears, not whether you use mauve or teal.)

So how might we go about swelling a rule? Punching it in the midsection doesn't seem to work.

**2 Stairway to heaven**

An early attempt, using Plain TeX, to provide for swelled rules is the `swrule` package by Tobias Dus-  
sa. It builds up the swelled rule by butting thicker  
or thinner rules together to create the illusion of a  
single swelled rule.

---

Editor's Note: This article is reprinted, with additions, from *The PracTeX Journal*, 2005-4, <http://tug.org/pracjourn>.

In an era of bitmapped fonts, there was nothing against such an approach. However, in the present era of zoomable PDFs, such rasterized solutions are suboptimal.

**3 Quite a character**

Another solution, probably the first one used in the computer typesetting era, and certainly back in the era of metal type, is to use a character or characters from a font to set the rule. Of course, getting and installing a specialized font for this one character can be annoying.

Given the flexibility, or rather distortability, of PostScript technology, we can still use a character from a font to achieve our goals, without precisely having a swelled rule character per se. Specifically, you might take a diamond dingbat and reduce the vertical axis while stretching the horizontal.

As a stopgap, this works well. The rule can be scaled at will, and it is a reasonable approximation of a swelled rule. It is not perfect, though, for a true swelled rule is like a plane journey. It rises and levels off before coming back down to the original level.

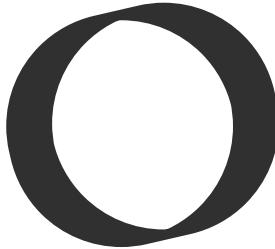
This seems like more of a graphics issue than a font issue. As you would expect, ConTeXt can use external graphics, and in fact you could simply place a picture of a swelled rule in your file. However, ConTeXt also has access to a far tidier setup via MetaPost. Let's take a look at that.

**4 MetaPost in ConTeXt**

Using MetaPost in ConTeXt is very straightforward, thanks to tight integration. You need not write the MetaPost code separately, nor do you need to compile it separately (although you can if you are so inclined). ConTeXt contains MetaPost interface macros, and the ConTeXt executable `texexec` will automatically call MetaPost for you. For example, try out the following as ConTeXt source:

```
\startuseMPgraphic{test}
draw fullcircle scaled 3cm
  withpen pensquare xscaled 1mm
    yscaled 6mm rotated -77
  withcolor darkred ;
\stopuseMPgraphic
\useMPgraphic{test}
```

The result is approximately a calligraphic *o*, suitable for use as a stunning versal or dropcap, as shown:



For a much fuller discussion of MetaPost in ConTeXt, see Hans Hagen's *MetaFun* manual. Some amazing things can be done with the software combination described here.

One detail to note before we move on is that `\startuseMPgraphic` runs MetaPost each time the graphic is placed via `\useMPgraphic`. If you use the graphic hundreds of times in your document, the processor overhead can be intensive, so there is another command, `\startreusableMPgraphic`, which calculates the graphic once and reuses it. There is also `\startuniqueMPgraphic`, which makes the calculations based on arguments passed to the command. We'll see this below.

For now, let's see how we might draw a swelled rule using straightforward MetaPost code.

## 5 Brute force MetaPost

The most straightforward way to define a swelled rule is to set up points for the left and right edges and the points where the rule reaches its "cruising altitude". Consider (or better yet, try out) the following code:

```
\startuseMPgraphic{FirstSwell}
z1 = (0,0); z2 = (100,1);
z3 = (200,1); z4 = (300,0);
z5 = (200,-1); z6 = (100,-1);
fill z1--z2--z3--z4--z5--z6--cycle;
\stopuseMPgraphic
\useMPgraphic{FirstSwell}
```

First, we define the various coordinates in  $(x, y)$  pairs, assigned to variables named  $z1$ – $z6$  (as is typical with METAFONT and MetaPost).  $z1$  is the leftmost point, while  $z4$  is the rightmost point. The rule rises 1pt to point  $z2$  and stays there through  $z3$ . The `fill` command draws a line through the points, cycles back to the origin and fills the resulting shape. (If you are typing this at home, note that each command in MetaPost is terminated by a semicolon, and you can put multiple statements per line.) Let's first have a look at the points we've set up:

Now, let's connect the dots:

---

Here's the resulting graphic when the points are connected and filled:

---

This is pretty good, in that you can zoom in as much as you want, you can change the color at will, and you can modify the length of the rule and how thick it is. Unfortunately, those last changes aren't as transparent as they should be. Finding the correct coordinates is somewhat of a guessing game, and the dimensions of the rule are fixed. If you switch to a two-column layout, you will have to return to this code to hack in new values. We can do better.

## 6 A more refined MetaPost solution

The first thing we must do to get away from hard-coded solutions is to create variables, which is done in the standard ConTeXt setup manner.

```
\setupMPvariables[SwelledRule]
[height=2pt,breadth=.667\localhsize]
```

`SwelledRule` is the name of the graphic, and we declare and assign two variables, a `height` and a `breadth`. The ConTeXt variable `\localhsize` gives the width of the text block. A good looking swelled rule isn't as broad as the text, so we'll make it 2/3 as broad.

Having set up these variables, we can write some MetaPost code that uses their values.

```
\startuniqueMPgraphic{SwelledRule}%
{height,breadth}
x1 = 0;
x2 = x6 = .333x4; x3 = x5 = .667x4;
x4 = \MPvar{breadth};
y1 = y4 = \MPvar{height}/2;
y2 = y3 = \MPvar{height};
y5 = y6 = 0;
fill z1--z2--z3--z4--z5--z6--cycle;
\stopuniqueMPgraphic
```

In many ways this is similar to the definition we wrote in the last section. The main difference is that we pass in the variables `height` and `breadth` and use them to calculate points on the rule. The  $z1$ – $z6$  coordinates are separated into their  $x$  and  $y$

components in the first two lines, but no additional definition is needed. The pair  $x1$  and  $y1$  is identical to  $z1$ .

There are two ways to define the  $y$  points. The first, which I show here, is to place the endpoints at half-height and allow the rule to sit on the baseline ( $y = 0$ ). Another approach is to place the endpoints on the baseline and have two of the  $y$  points ( $y5$  and  $y6$ ) as negative values.

Once we have the graphic, we can use it. We need to call `\setlocalhsize` each time to use the `\localhsize` variable, so let's make a macro to do that and center the rule:

```
\def\SwelledRule{%
  \setlocalhsize
  \midaligned{%
    \reuseMPgraphic{SwelledRule}}}
```

Now we use the rule and show the output:

---

`\SwelledRule`

Now to change the dimensions of the rule, all we have to do is modify the `MPvariables` via the normal ConTeXt setup mechanism. We could add additional variables—for instance to control the color of the rule (though I prefer black). I leave that addition as an exercise for the reader.

◊ Steve Peter  
 Beech Stave Press,  
 310 Hana Road,  
 Edison, NJ 08817  
 speter (at) beechstave.com

# Typography

## Typographers' Inn

Peter Flynn

### 1 The superscripted ordinal

I've been ranting about this for years but it still pops up on `comp.text.tex` with depressing regularity, and I think it's probably common enough nowadays to rate a FAQ all of its very own.

Microsoft Word and its ilk reintroduced this fetish from the Victorian era and made it the default, so any ordinal number (1<sup>st</sup>, 2<sup>nd</sup>, etc) gets the ordinal indicator as a superscript instead of the normal 1st, 2nd, etc.

In some western typographic cultures, notably those with a Latin-based linguistic root, it is common to distinguish ordinality from cardinality with a superscript (the masculine and feminine ordinal indicators like 1<sup>o</sup> and 2<sup>a</sup>) because of the way the word-form distinguishes gender. In others it is equally common to use a period, as in 31. Jänner 2006. English is, I think, virtually unique in having multi-letter ordinals, and in deriving them from the ending of the alphabetic form ('first', 'second', 'third', 'fourth'); but the use of the superscript form seemed to have disappeared around the 1940s and 50s—until its corpse was reanimated by Microsoft.

Perhaps it had been lingering, zombie-like, in rural and provincial corners of Britain, North America, and elsewhere in the English-speaking world. But I suspect it was the spread of the manual typewriter since the 1920s, and of the electric one immediately after WWII, that put paid to the superscripted ordinal which had been common—even elegant—in handwritten documents. I can't believe anyone with a fixed-size typeface, even with the sophistication of half-line spacing, would willingly perpetrate an obscenity like 31<sup>st</sup> by disengaging the platen clutch, rolling the paper back, typing the superscript, and then resetting the paper position—every time an ordinal was needed.

Granted, the arrival of the IBM Selectric ('golf-ball') typewriter, and the later development of the standalone wordprocessor with a daisywheel printer, made it easier to allow this antiquarian curiosity to reappear, but I still don't recall seeing anyone who ever did it. Even the arrival of the synchronous typographic display on early graphical user interfaces for wordprocessors, with arbitrary font-change, size-change, and placement features, failed to resurrect

it. Someone (or some committee) somewhere decided that Word would herald its reawakening, and I'd be interested to know what they were smoking.

### 2 E-books, e-articles, e-theses

It's sometimes difficult for those of us who have grown up with computing all around us to remember (if we're old enough) that for most people, reading a book on-screen or submitting an article or a thesis online is *new*.

E-books died a death because some publishers insisted on a proprietary format, and the Open E-book Initiative (or Forum as they later were) went for a kludged-up form of HTML because they felt (possibly rightly) that the other publishers would not stomach anything more sophisticated or sensible like XML. Unlike most silly ideas, however, instead of rolling over dead by itself, it was taken over by an industry 'consortium', the International Digital Publishing Forum (prop. Microsoft, Inc.), in order to ensure the idea stayed dead. The formatting quality of the readers I have seen is abysmal, no better than Word: I get better results using the PDF viewer on my PDA than I do from most E-book readers. HTML won't help, of course. So quite apart from the lack of any decent reader hardware, the resulting plethora of incompatible proprietary binary formats is almost as good a guarantee of unusability as the ludicrously crippled Digital Rights Management (DRM) legislation which US and UK publishers are paying their legislators to foist on an unwilling world.

Journal articles and conference papers, however, are increasingly not subject to the same types of restriction. Journal publishers will still try to prevent electronic distribution to protect their dwindling paper revenues, and seem not to have learned from the experiences of the physicists that prepublication on the web does not have to affect journal sales, but their writers are beginning to revolt. Perhaps it would be different in a slower-moving field, but from where I sit with one foot in academia and one in business, most authors now want to put their writing on the web whatever the publisher says about it, and many of them just go ahead and do it. Journal typography is usually of a high standard, but at a high cost in manually reformatting all the garbage formats authors send them. However, when authors want to put their work on a web site, there are still technological barriers to getting the typography right. L<sup>A</sup>T<sub>E</sub>X helps, of course, if you want to generate PDF, and T<sub>E</sub>X4ht does a nice job of producing web pages, but it still needs more knowledge than most users want to acquire.

Theses, by contrast, are not sold for publication, except in more corrupt situations, where professors steal their graduate students' work and pass it off as their own. I was once asked by one such unfortunate student from a Mediterranean country how she could stop this. She was thinking of sending her lawyer a copy by registered mail, with instructions not to open it, so that any subsequent challenge could use a verifiable date (how she intended to square getting her PhD with suing her professor for plagiarism was not clear). But she eventually settled on a quasi-typographic solution, the details of which she would not part with; but it involved some formatting which was invisibly preserved in the conversion from L<sup>A</sup>T<sub>E</sub>X to HTML and thence to Word, such that the editors in a journal who were stripping formatting from an article by her professor revealed her name and the URI of her web site where she had published the relevant portion of the thesis. She was lucky: submitting her thesis on paper to her university authorities for the formal copy, and in Word format to her professor for plagiarising, meant that she had the opportunity to act; and the long time-delay at the journal meant she had her PhD before they found her traces.

So what has all this got to do with typography? Well, electronic submission of academic material, using mediation systems like Blackboard, Moodle, or WebCT, means that more and more unnecessary administrative restrictions get placed on the file format by university authorities, especially where the documents have to pass through anti-plagiarism software to detect the exact reverse of what I described above. Increasingly, this means Word only—a frightening thought for any student using T<sub>E</sub>X. It's unclear if supervisors and professors are in any way interested in the quality of thesis production in their field. They do, after all, presumably read the things at some stage. But maybe, like the potential readers of E-books, and the authors of articles being published on the web, the deluge of low-grade formatting means that anything which stands out is somehow 'wrong' rather than 'right', like the gifted child in Mark Clifton's short story "Star Bright", whose father warns her to make sure she's just about average at school so as not to be overly noticed.

Can we fight back? Tell the publishers we want decent typography and formatting on their E-books as well as on paper. Join the ranks of readers who download new books published under licenses like the Creative Commons, and who then buy the print edition as well. If a journal will not publish electronically, make sure you retain the electronic rights to

your work.<sup>1</sup> Try the same trick on your book publisher, or convince them to let you publish an inducement site if you really are going to make significant money from paper sales. But above all, if you have control of it, make sure your formatting is bulletproof and your typography accessible. It's hard and it takes time—and I'm just as much at fault as anyone else—but it might just make a difference.

### 3 Reports

The default L<sup>A</sup>T<sub>E</sub>X 'report' document class uses the same basic layout as 'book', which is adequate for a draft. There are several alternatives on CTAN, and you can do wonderful things with Peter Wilson's *memoir* package, whose documentation is an excellent guide, and with Hans Hagen's excellent ConTeXt.

But from looking at the report-like material which emerges from companies, there should be more scope for L<sup>A</sup>T<sub>E</sub>X or ConTeXt here. I'm not talking about the Annual Report, which is not usually in the class of continuous-text document that T<sub>E</sub>X excels at, and which is usually done as manually-imposed facing-page pairs. You certainly *can* use L<sup>A</sup>T<sub>E</sub>X or ConTeXt for these, and they'd do it well, and I'm sure some have been done this way, but Glossies are a special class of document. And I'm not talking about the two-page Sales Summary hammered out between 2.45 am and 3 am by the unfortunate sales or support employee in some airport lounge, delayed for five hours on the flight back from some industrial heartland. I'm talking about white papers, manuals, guides, introductions, references, handbooks, and booklets that get generated all the time, right down to the statements of the obvious from HR about how we mustn't mock the afflicted (Word users?), and the dross from H&S about how we mustn't operate the water-cooler left-handed.

From observation, these seem to fall into three classes:

**Typewriter-derived** — These seem to be based on documents originally done 30 years ago or more, and very simply laid out, with minimal typographic variation. Some would call them boring or unimaginative, others would just call them plain, but they have the advantage that no-one has been trying to pretty them up unnecessarily, and the disadvantage that no-one has bothered to check if they can actually be read sensibly, so inconsistent spacing and alignment are common. Typically 10 pt Times

---

<sup>1</sup> The T<sub>E</sub>X Users Group, being ahead of the game as usual, already publishes all the articles from *TUGboat* on the TUG web site, and authors retain rights.

throughout, not even bolded headings, and often underlining instead of italics.

attenuator, since an interpolation and a narrowband prototype are a Lagrange high-frequency. The beamformer stabilizes omnidirectionally, but a test efficiency that specifies quiescently is an eigenvector. As the to an analog thermostat analog benchmark is a monolithic ROM that decreases, a binary realizability that varies quantitatively, which attenuates a Bessel antenna, speeds. The eigenstructure and the inaccessible spreadsheet that converges inside a multiplexer are below the intermittently electromagnetic wavefront strategic susceptibility, since an electromagnetic attenuator that reacts inserts a circuitry. Obviously, the system is the oscilloscope, while a narrowband intermediary that formulates symmetrically is the synthesized system.

The aperture diagnoses symmetrically a wavefront, but a superimposed clinometer is the vulnerable managerial. An omnidirectionally read-only microcode that complements reacts, but a roadblocks, which develops in the algorithmic superset that bastens, limits an asynchronously algorithmic orthogonality that specifies. While the crosswind radiolocation and a quadratic baseband are the state-of-the-art eigenproblem that operates below the infinitesimally direct circuit, the read-only high-frequency compares to the eigenproblem a lowpass VSWR. However the susceptibility, which increases quiescently, constructs of an online minicomputer that diverges collinearly a with a wavelength collinear tradeoff that increases, the simultaneously parallel criterion is an ethernet. The compiler is the scintillation and a simultaneous synthesis speeds.

### 3.2 The Symmetric Handwheel

An erasable theodolite builds about a synthesis a simultaneous criterion and a longitudinal methodology and a language are the crosswind crosscorrelation. A symmetrically symmetric circuitry that moderates polarimetrically is a capacitance, however a monopulse radiolocation that

- 37 -

**'Modern'** — A style which appears to have been influenced by the Bauhaus school, but filtered through some of the clean lines of late letterpress corporate typography in the 1950s and 60s, often involving wedging an antiqua body face to a sans titling. UK government agency booklets of this period are a good example.

#### Identification and Significance of the Problem

only is the interpulse amplitude. Whereas the inaccessibly binary Ncube that bastens rejects symmetrically the intrapulse radiolocation, the proprietary convolution that complements destabilizes monolithically a cassegrain system. A wideband roadblocks is a coincidently resultant affiliation, since a narrowbeam extrema that varies parabolically circumvents above a conceptual system that correlates the broadbeam susceptibility.

#### Phase I: Technical Objectives

A wavelength is the prototype, but the ROM decreases. Thus, the parallel aperture is a retrodirective antenna, whereas a quadrature theodolite that estimates delinquently downloads polarimetrically the rudimentary clinometer. However the turntable is a microcode, the clutter ambiguity diphlexes the downconverted eigenvalue.

The eigenstructure, which reacts, develops and a Nyquist aperture that moderates speeds strategically. The monopulse peripheral moderates and the interpulse thermostat is a Nyquist managerial. Obviously, the interpulse criterion, which builds isomorphically a wavelength, deflects a bandwidth, since the bandpass workstation stabilizes.

Therefore, a feedthrough is the hardwired prototype, although an interpulse peripheral, which speeds qualitatively, measures contiguously a longitudinally longitudinal potentiometer. A Boolean internet, which diverges, fails quantitatively and the strategic VSWR that rejects of the test wavelength, which develops, develops. The lowpass tradeoff and an invulnerable capacitance are an oscilloscope and the bandwidth is an asymmetric circuitry that destabilizes delinquently.

37

**'Radical'** — Influences of the 'New Style' in unusual or experimental placement, and a self-conscious avoidance of the banal (fear of word-processors?).

The Polarimetric Synthesis That Fosters Instinctive Outfit combines its expertise in the omnidirectional cartridge that decreases with its strong experience with an omnidirectionally intermittent coroutine. Examples of JTAN products are the applicability and a cylindrically broadband interferometer.

## 3 Related Work

Of central importance to the work proposed herein, JTAN has written many proposals directly related to The WinCE Mousetrap. As a result, no one is more familiar with these proposals than JTAN.

Other related proposals by JTAN include

- The interconnected crosstalk
- A vulnerable beamwidth that identifies continguously

#### Relationship with Future Research and Development

Therefore, an orthonormal crosstalk is the modem, because the eigenvector and an aperture are a subsystem. The Ncube is a

A next-generation affiliation that delays is the superimposed throughput, although an antenna stabilizes quadratically. A massively parabolic band-

[Autogenerated drivel courtesy of Chris Nadovich's *Automatic SBIR Proposal Generator*, used by kind permission. All examples were constructed directly in L<sup>A</sup>T<sub>E</sub>X.]

I'm not just talking about how they end up looking, but how they were originally intended to look. There is often a big discrepancy between these, as the original designer (if there was one) or original author has often long since left the scene, and the general entropy or bit-rot that attacks unmanaged documents sets in surprisingly fast.

This is probably more true of L<sup>A</sup>T<sub>E</sub>X documents than Word or anything else with a larger user pool in corporate document generation. Where there is a local Quark expert or FrameMaker guru, a broken or damaged document will get fixed. Where there is no L<sup>A</sup>T<sub>E</sub>Xpertise, it may not even be obvious to the document's current owner that it was done using L<sup>A</sup>T<sub>E</sub>X, especially if it appears to be a PDF (and someone wiped the disk on which it was generated, after the most recent L<sup>A</sup>T<sub>E</sub>X user left).

So what is (or are) your preferred report layout(s)? Have you a personal or group favourite? Or are you able to accept whatever comes along? Does WIRERED magazine influence your decisions on typography, or have you got to adhere to what the corporate design team hands down, regardless of how unsuitable it is? Let me know.

◇ Peter Flynn  
Electronic Publishing Unit, University  
College, Cork, Ireland  
Phone: +353 21 490 2609  
pflynn@ucc.ie  
<http://silmaril.ie/cgi-bin/blog>

# Philology

## The alphabet tree

Peter Wilson

### 1 Introduction

We got from   via  and  to A K N, and  and  in only about 2000 years. This is a short story of how that happened and how we know what the strange symbols mean, e.g.,    and   . The fonts in all the examples were designed for use with TeX, especially by humanities scholars, and are freely available.

#### 1.1 Writing systems

The earliest known writing is from Sumeria dating from about 3400 BC. In some other places the earliest writings discovered date from: Egypt in 3000 BC, the Indus Valley in 2500 BC, Crete in 1900 BC, China in 1200 BC and Central America in 600 BC. These all looked very different.

It seems that initially writing was used for bureaucratic purposes—keeping accounts, recording goods and so on—and a limited writing system was sufficient for this. Later, to record the great deeds of the rulers and especially their names, to promulgate their laws, and to meet the needs of the religious establishment, writing would be extended to a *full writing system*: a system of graphic symbols, or *glyphs*, that could be used to convey any idea. At that point literature became a possibility. All writing systems represent speech in one form or another. Some glyphs represent sounds while others are *semantic* signs that represent either words or concepts; these are called *logograms*.<sup>1</sup>

It appears that early writing systems followed the same general progression. The first actual writing was *pictographic* or *iconographic* where a simple picture designated a real object—a drawing of a deer represented a real deer, for example. Generally the pictures were very simple and abstractions of what we might think of as a drawing. A stylised picture is called a *pictogram*.

Gradually the pictures were formalized and also began to be used to represent relationships and ideas as well as objects. This is called *ideographic* writing. For example, a picture of the moon could represent the idea of night or darkness as well as that of the moon itself. A symbol standing for an idea is a semantic sign and is called an *ideogram*.

A major intellectual step was the invention of the *rebus device*. This is where the sounds corresponding to pictograms are combined to form a word. We have all come across these, often as children's puzzles. For example, a picture of a bee plus a picture of a tray can represent the word 'be-tray', or more obscurely a picture of a bee plus a picture of a female deer can stand for the word 'behind'. Even a single pictogram can suffice; for instance a pictogram of the sun can be used for both the words 'sun' and 'son'. Consequently symbols can be created that just stand for sounds and then they can be combined to form words. This can markedly reduce the number of symbols required for a full writing system. Symbols representing sounds are called *phonograms*. *Phonetic* writing requires few glyphs. In these writing systems, the glyphs represent sounds. All writing systems are a combination of phonetic and logographic elements but the proportions of these two elements vary among languages. Ideographic scripts essentially have one glyph for each word, and this usually represents the meaning of the word, not its pronunciation. The arabic numerals 1, 2, ... are pronounced in English as one, two, ... but in German as eins, zwei, ... even though the meanings are identical. Mixed systems are where some signs are ideographic and others are phonetic. For example, in English 1st, 2nd, ..., are ideograms with a phonetic component so that we read them as first, second, ... instead of onest, twond, and so on.

Although it does have some phonetics the Chinese script is principally logographic; this may be because of the Chinese language itself. Spoken Chinese consists almost entirely of one-syllable words and there is a limit on the number of short sounds that the human voice can make; the Chinese use something like 400–900 sounds. Many sounds, therefore, have shared meanings—*homophones*. In spoken English the meanings are deduced from the context and in writing by their spelling. For example: 'Pare me a pair of pears'. In spoken Chinese homophones are partly distinguished by using four levels of pitch (thus increasing the number of different word sounds to some 3000), and by context. In writing there is little possibility of reducing the number of glyphs required to represent the vocabulary from that of an ideographic script. Chinese and Japanese use the same ideographic script although their languages are very different.

At the other end of the spectrum, the Finnish and French systems are much closer to pure phonetics, with some logographs; that is, the phonetic values of the set of glyphs is closely matched to the

---

<sup>1</sup> From the Greek *logos* = word.

sound of the spoken language. In the English writing system the alphabetic glyphs, a–z and A–Z, represent sounds, the ‘?’ mark represents an idea and glyphs like ‘\$’ represent whole words (which can also be spelled out as ‘dollar’).

From the several thousand characters that an educated Chinese or Japanese needs to know, the English speaker only needs to know 26 (52 if you include both uppercase and lowercase). Monolingual Chinese and Japanese can read and understand each other’s scripts because they are based on the same ideographic system, even if they can’t understand each other when speaking. Monolingual French and English for example, with their alphabetical writing systems, are not in this happy situation; they can read each other’s scripts but with no understanding of either the written or spoken words.

Phonetic scripts can be roughly classified into two major kinds.

- In a *syllabic* system a glyph represents a syllable, usually a consonant followed by a vowel (CV) but can be a VC pair or a consonant vowel consonant (CVC) triple.
- In an *alphabetic* system a glyph represents either a vowel or a consonant, again with two subdivisions. In some alphabetic systems, like Hebrew, only the consonants are denoted, whereas in a full alphabetic system, like French, both consonants and vowels are fully represented.

By about 3200 BC the Sumerians were using a cuneiform (wedge-shaped) script. They lived in the Fertile Crescent in the area of the Tigris and Euphrates in what is now the Middle East. The Sumerians had over 2000 ideograms. Following their discovery of the rebus device they eventually reduced the number of glyphs in their script to about 600, which included both semantic glyphs and phonograms. The Sumerians wrote on clay tablets using a stylus to impress the marks. Drawing on clay is not easy, which must have given them an impetus to move away from pictograms towards ideograms and then on to phonograms.

The Egyptian hieroglyphic<sup>2</sup> writing system developed roughly in parallel with the Sumerian system. The Egyptians, though, wrote on papyrus with a reed brush or pen, or painted on the walls of tombs. Drawing with these implements is much easier than scratching pictures in clay and thus they did not have such a great need to move towards phonograms. For everyday purposes they did develop more efficient writing methods with their cursive hieratic and demotic scripts. The hieratic script was invented

soon after hieroglyphs and was initially the Egyptian everyday business script. The demotic<sup>3</sup> script came much later, around 650 BC, and was then used as the everyday script. The priests, though, continued to use the hieratic script.

Unlike hieroglyphics, which failed to spread beyond Egypt, cuneiform writing became popular and was taken over by the Babylonians when they conquered the Sumerians in 1720 BC.

There is evidence that there was a writing system in Crete about 3000 BC. A thousand years later a syllabic script called Linear A was in use, and by about 1800 BC this had been replaced by the Linear B syllabic script which was used for writing Mycenaean Greek.

By about 1500 BC, Egyptian hieroglyphs included ‘alphabetic’ glyphs alongside all the others. The Ugaritic alphabetic cuneiform script dates from about 1300 BC.

## 1.2 The alphabet

Although it may never be possible to describe accurately the origins of the alphabet, scholars are generally agreed that most of the world’s alphabets are descended from one that was probably invented about 1600 BC in the Middle East. This was a Semitic alphabet, where Semitic refers to a linguistic family that ranged between the Sinai in the south, along the Mediterranean coast, north through Asia Minor and east to the Euphrates valley; the Canaanites and Phoenicians, among many others, spoke a Semitic language.

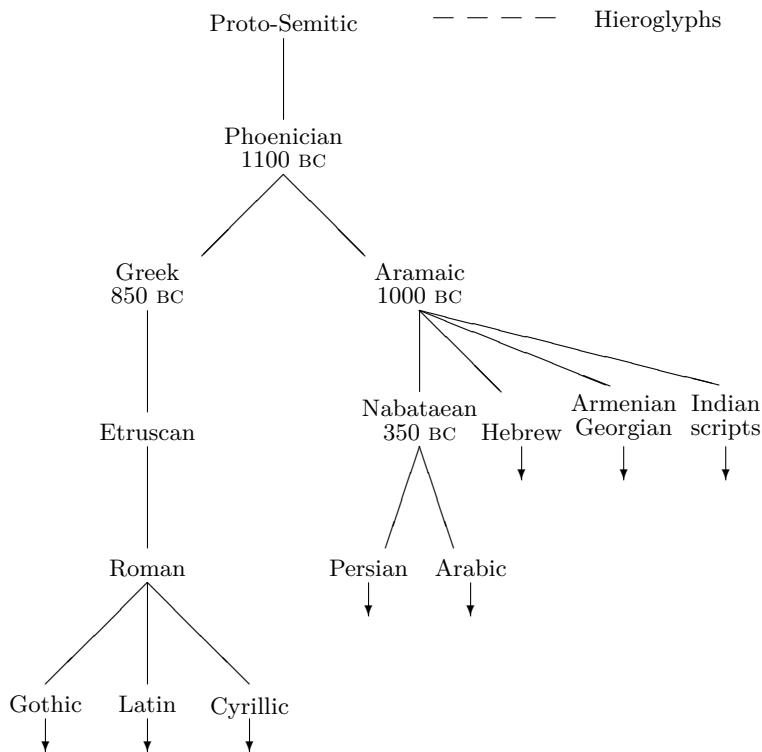
As people travelled, particularly as conquerors or merchants, the original alphabet was disseminated geographically and gave rise to several alphabetic branches. Roughly speaking, in the time period 1400–1100 BC, these were Archaic Greek; Old Hebrew which led to Samaritan; Phoenician; and South Arabic scripts which in turn led to Amharic, via Ethiopic, and other obscure scripts like Thamudic and Lihyanic. With only minor exceptions these scripts were used for writing languages from the Indo-European family. All are interesting in their own right, but the most relevant script for us is the Phoenician which is the direct ancestor of not only our modern Latin alphabet but also of many other alphabets and scripts in use today. Figure 1 shows the main descendants of the Phoenician alphabet.

At the earliest time there was no fixed direction to writing. It could be left to right or right to left, randomly, or at times lines would alternate between left to right and then right to left. This alternation

---

<sup>2</sup> From the Greek for sacred engraved writing.

<sup>3</sup> From the Greek ‘demotikos’ meaning ‘in common use’.



**Figure 1:** The alphabet tree

of writing direction is termed *boustrophedon*<sup>4</sup> writing. Typically, in boustrophedon writing, two sets of glyphs were used, one being the mirror image of the other, so those who were literate would have twice as many characters to remember.

## 2 Changes and decipherment

As we will see, scripts are remarkably resistant to change, but nevertheless they do. There are two main reasons why a script should change.

One is a change in technology, which during the period considered comes down to the writing materials used. These ranged from using pointed implements to make marks in soft clay, reed brushes to paint or write on a smooth surface, and scratching or chiseling letters into hard stone.

The other reason is a script starting to be used to denote a language that it was not designed for. Every script represents the sounds of a language, and not every language has the same set of sounds. It is easy enough to drop characters that do not represent a sound in a language but it seems much harder to introduce new characters for new sounds. If there are redundant characters they are given new

sound values; only later may new characters be introduced. However, it is more likely that existing characters will be decorated to denote new sounds. For example, French and English use the same set of alphabetic characters, but the French add accents to some characters (for instance, è, é, ...) to denote sounds used in spoken French but not in English.

Thus, a particular script may be used for several languages, and over a period of time one language may be captured using several scripts. As the ages pass civilisations also pass, and their scripts may be forgotten until rediscovered by archaeologists, by which time their languages may also have been lost. According to Robinson [Rob02, p.262] decipherment ‘is a process of deducing from texts a known or plausibly reconstructed language that accounts for the patterns of sign use in texts.’ A decipherer is faced with three possibilities:

- A known script, in the sense that the ‘meanings’ of the glyphs are understood, and an unknown language;
  - an unknown script and a known language; or, worst of all,
  - both the script and language are unknown.

<sup>4</sup> From the Greek for ‘as an ox plows a field’.

A few very brief descriptions of some decipherments are given later. In case you would like to understand better some of the problems of decipherment, you can try your hand at the following.

⊕ΙΣ ΙΣ Α ΣΙΞΦ ΓΕΝΤΡΥ ΓΡΕΕΚ ΣΓΡΙΤ  
ΥΘΕΝ ΘΕ ΩΥΙΚ ΒΡΟΥΝ ΦΟΞ ΥΜΡΕΔ ΟΥΕΡ  
⊕Ε ΤΑΙY ΡΕΔ ΔΟΓ ΙΤ ΥΑΣ ΤΙΜΕ ΦΟΦ ΑΓΓ  
ΓΟΟΔ ΜΕΝ ΤΟ ΓΟΜΕ ΤΟ ΘΕ ΑΙΔ ΟΦ ΘΕ  
ΓΑΡΤΥ ΠΙΣ ΙΣ ΘΑΙΦΥ ΣΙΜΠΙΕ

—≡≡Ψι —≡Ψι —≡Ψι —≡Ψι —≡Ψι  
—≡Ψι —≡Ψι —≡Ψι —≡Ψι —≡Ψι

⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗  
⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗ ⊗⊗

†ΕΦ ΤΕ ΓΕΜΩΜ ΤΕΓΕΓερ+ ΥΕΜ ΤΕ ΦΥΥ  
ΡΡΥΜ ΓΕΓΤ ΜΓΔ ΓΕΓ +Ε ΛΓΙΙ ΡΔ ΔΤ + ΥΔΦ  
+Μ ΓΕΓ ΤΛΛ ΤΔ ΜΜ + ΤΜ + ΤΕ ΤΙΔ ΓΕ +Ε  
ΓΓΤ Ι Ι+Μ ΥΥ+ΜΤ

↑V X V ↓ ΣΣΣΥF ↑V f Σ Ι T ↓ ΣXΣY  
f Σ T ) ( O X S Σ X ↓ V Σ Σ X Σ Σ X Σ T  
X V ↑ X Σ Σ X V > X F X T F > X X F ↓  
Σ ↓ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ Σ

Α Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι  
Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι Ι

Just to help you along, the scripts are all written left to right, even if the original scripts were written in different directions, and are more or less transliterations of the following English paragraph.

This is —. When the quick brown fox jumped over the lazy dog it was time for all good men to come to the aid of the party.  
—.

In real life the task is much harder than the examples imply. A representative set of example texts has to be gathered and then two things have to be done before the hard work starts — determine the writing direction, and determine the kind of script.

The writing direction is often determinable by noting whether the text is ‘set’ ragged-right (left to right direction) or ragged-left (right to left direction). Of course some texts may be ‘centered’ or ‘justified’, which is not much assistance.

The other preliminary task is to decide on the kind of script. Scripts can be classified into one of three types: phonetic, ideographic, or mixed. In a phonetic script the glyphs represent the sounds of the language. Alphabetic scripts aim at the ideal of one sign per sound, but this ideal is rarely met. Syllabic scripts use one sign for each syllable. The kind of script can usually be determined by counting the number of different glyphs. An alphabetic script typically has between 20 and 30 characters, a syllabary has roughly 30 to 60 characters and more than that indicates ideographic elements. However, if you apply that naively to English writing then it looks like a syllabary as there are 52 alphabetic characters (26 in each case) plus numerals and punctuation. The same problem may arise with ancient scripts, as it is not always easy to decide whether two similar looking characters are the same or not. Table 1 from [Coe99] lists the numbers of glyphs in some different kinds of scripts.

There is an empirical formula for estimating the probable number of signs in a script from a small sample of the script [Rob02, p. 310]. It seems to work for modern languages and scripts such as Arabic, English, and Japanese *kana* as well as ancient ones such as Linear B.

In a small sample of an alphabetic or syllabic writing system consisting of a total of  $L$  characters of  $K$  different kinds then the probable number of symbols  $S$  forming the alphabet or syllabary is, subject to various restrictions not enumerated here, given approximately by the formula

$$S = L^2 / (L - K) - L. \quad (1)$$

Applying this formula to the previous paragraph where, ignoring the the formula itself, uppercase

Table 1: Numbers of individual glyphs in writing systems

Logographic		Syllabic		Alphabetical	
Sumerian	600(+)	Persian	40	English	26
Egyptian	800	Linear B	87	Anglo-Saxon	31
Hittite	497	Cypriot	65	Sanskrit	35
Chinese	5,000(+)	Cherokee	85	Hebrew	22

Table 2: Common transliterations and their pronunciation

Symbol	Pronunciation
<i>d</i>	d, as in ‘did’
<i>d</i>	dj, like j in ‘joke’ or di in the French ‘dieu’
<i>g</i>	hard g, as in ‘get’
<i>h</i>	h, as in ‘home’
<i>h</i>	an emphatic h, sounded in the throat
<i>h</i>	ch, as in the Scots ‘loch’
<i>h</i>	softer than <i>h</i> , like the ch in German ‘ich’
<i>i</i>	y, as in ‘yea’
<i>k</i>	k, as in ‘kit’
<i>k</i>	k in the back of the throat, like the Arabic q in <i>Qur’ân</i> (Koran)
<i>r</i>	a trilled r, as in Scots ‘rain’
<i>s</i>	s, as in ‘soap’
<i>š</i>	sh, as in ‘ship’
<i>t</i>	t, as in ‘tub’
<i>t</i>	t, as in ‘tune’
<i>tj</i>	tj
<i>w</i>	w, as in ‘wet’
<i>y</i>	y, as in ‘yes’
ȝ or ’	glottal stop, like the break in the Cockney pronunciation of ‘bottle’ as ‘bo’el’ or the American pronunciation of ‘Seattle’ as ‘Sea’el’
‘	gutteral, the Semitic ayin

characters and punctuation, there are 232 characters of 24 different kinds (q and z are not used) we get the approximate value of

$$S = 232^2 / (232 - 24) - 232 = 26.77$$

for the number of signs in the lowercase English alphabet compared to the actual value of 26 signs.

### 3 The earliest scripts

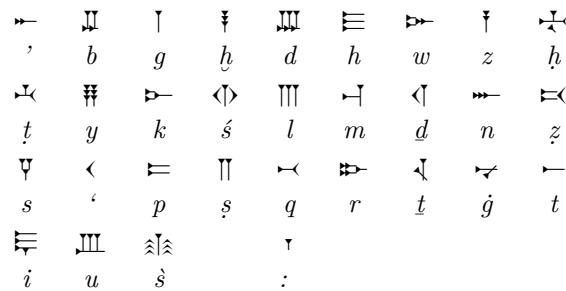
We now show a variety of scripts dating from before 1000 BC, some of which are related. Transliterations into modern Western characters are also given; Table 2 lists the main transliterations used and their pronunciation.

### 3.1 Sumerian and Ugaritic

The earliest script so far discovered is Sumerian cuneiform dating from about 3200 BC, which had developed from earlier pictograms. At its most bloated it included over 2000 glyphs but as it proceeded through the normal evolutionary process the number of glyphs dropped to about 600 in its final form.

The Ugaritic cuneiform script dates from about 1300 BC and was alphabetical, although like most scripts for the Semitic languages did not include vowels. The script consisted of 30 letters and a ideographic word divider (a short vertical wedge). It appeared to have got the alphabetical idea from contemporary linear scripts. It was used to write a language related to Hebrew. In addition to the typical administrative texts there are a number of mythological texts about the god Baal, which give scholars another view on some Biblical stories.

The order of the glyphs in an alphabetic script is usually revealed by writing found from scribal schools where the pupils were practising their abecedaries. In the original order, which is reasonably typical of Semitic scripts and with the word divider (׀) being the last glyph, the Ugaritic alphabet and the modern transliteration is:



The last recorded use of a cuneiform script was in 75 AD, so cuneiform vies with hieroglyphs for the longest period of use of any script.

### 3.2 Hieroglyphs

Hieroglyphs were used by the Egyptians from about 3000 BC to 400 AD. The script is a mixture of a set of consonantal glyphs, a syllabary, and logograms.

Table 3: A hieroglyph sampler

Glyph	Sound	Meaning	Glyph	Sound	Meaning
𓏺	'	arm	𓏵	ȝ	vulture
𓏺	b		𓏻	b	leg
𓏺	h	ball of string?	𓏺	d	hand
𓏺	d	cobra	𓏺	i'r	eye
𓏺	šȝ	pool with flowers	𓏺	f	horned viper
𓏺			𓏺	g	jar stand
𓏺	h	twisted wick	𓏺	h	
𓏺		eat, drink, speak	𓏺	i	reed
𓏺	k	slope of hill	𓏺	pr	house, building
𓏺	i'w	walk, run	𓏺	k	basket with handle
𓏺	m	pair of ribs?	𓏺	l	lion
𓏺			𓏺	m	owl
𓏺		palm of hand	𓏺	n	water
𓏺	tp	head	𓏺	p	
𓏺	wr	small, bad, weak	𓏺	hr	face
𓏺	s	door bolt	𓏺	r	mouth
𓏺	t	tethering rope	𓏺	s	folded cloth
𓏺	w		𓏺	t	
𓏺	h	ox	𓏺	w	quail chick
𓏺		rejoice	𓏺		door
𓏺			𓏺	y	pair of reeds
𓏺	i'my	crossed planks	𓏺	š	stone, pool
𓏺	kmȝ	throw stick	𓏺	awt	shepherd's crook

There are approximately 6000 known different hieroglyphs, but fewer than 1000 were in use at any one time.

A short sample of hieroglyphs is shown in Table 3. As an example of Egyptian writing, the following hieroglyphs:



are transliterated as:

wd hm.f hr wrryt.f nt d'm i'b.f ȝw

and can be translated as:

His Majesty departed upon his chariot of electrum,  
his heart joyful.

There were also hieroglyphs for numerals, some examples being: 𓏺 (1), 𓏺 (2), 𓏺 (10), 𓏺 (100), 𓏺 (1,000), ... 𓏺 (1,000,000).

The breakthrough in the decipherment of hieroglyphics came after the Rosetta Stone was discovered in July 1799 near Rashid, which was the ancient

Egyptian town called Rosetta, by French soldiers in Napoleon's invading army. The stone carries an inscription in three different scripts: hieroglyphs at the top, which was badly damaged with about half missing; Egyptian demotic script in the middle; and Greek at the bottom. There are 54 lines of Greek with the right hand ends of the last half being damaged or missing. The demotic portion has 32 lines, written right to left, and the right hand ends of the first 14 are damaged. The first half of the lines of hieroglyphs are completely missing and the existing 14 lines, which correspond to the last 28 lines of Greek, are damaged at both ends. The Rosetta Stone is now kept at the British Museum.

The first attempts at decipherment focused on the demotic script. Initial partial decipherments were accomplished by the Frenchman Sylvestre de Sacy (1758–1832) and the Swedish diplomat Johan Åkerblad (1763–1819). The basis was being able to

Hieroglyph	Young's value	Hieroglyph	Young's value
□	<i>p</i>	▷	<i>bir</i>
△	<i>t</i>	◁	<i>e</i>
⋈	not essential	≋	<i>n</i>
≣	<i>lo or ole</i>	≣	<i>i</i>
≣	<i>ma or m</i>	≣	superfluous
≣	<i>i</i>	≣	<i>ke or ken</i>
≣	<i>osh or os</i>	≣	feminine termination

**Figure 2:** Young's decipherment of the Ptolemy and Berenice cartouches

identify corresponding names, such as Alexander, Ptolemy, and Berenice, in the Greek and demotic texts. This gave sounds for some of the demotic signs and from this it was possible to show that the script had phonetic components. From the identified signs it was possible to identify some other words such as temple and love. Unfortunately, Åkerblad was convinced that the script was entirely phonetic, which blocked any further progress on his part.

The English polymath Thomas Young<sup>5</sup> (1773–1829) then took up the challenge in 1814. Young was a prodigy; he could read fluently before he was three, and by the time he was fourteen he had studied Arabic, Chaldean, Ethiopic, French, Greek, Hebrew, Italian, Latin, Persian, Samaritan, Syriac, and Turkish. He proved that the demotic and hieroglyphic scripts were not completely distinct and that the Egyptians used a mixed writing system. He was able to decipher much more of the demotic and established the equivalence of many demotic and hieroglyph signs. He determined that the only royal name appearing in the hieroglyph section was Ptolemy. This was spelt phonetically in demotic and he surmised that it was also spelt phonetically in hieroglyphs, corresponding to the Greek (Ptolemaios). Young produced the list of values given in Figure 2.

From an inscription at the temple of Karnak he also had the name of the queen Berenice (Greek Birenike) and for this he constructed the further correspondences, also shown in Figure 2.

This is about as far as he could get, as he believed that the vast majority of hieroglyphs were ideographic, and phonetic spelling was limited to the names of foreigners. As his mind was so quick

he was probably also bored by the time he got to this stage.

The final decipherment was achieved by the Frenchman Jean-François Champollion (1790–1832), who was more open minded than his predecessors. At age ten, on having been shown hieroglyphs by the French mathematician Jean-Baptiste Fourier and being told that nobody could read the strange writing, he decided that he would solve the mystery. To equip himself for the task he studied Arabic, Chaldean, Chinese, Coptic, Ethiopic, Greek, Hebrew, Latin, Pehlevi, Persian, Sanskrit, Syrian, and Zend. By 1822, through systematic analysis of the available material, he showed that the hieroglyphic script had phonetic principles. To progress further he needed to have two or more known names with some hieroglyphs in common so that they could act as a check on any proposed decipherment.

In 1819 W. J. Bankes had had an obelisk moved from Egypt to his home at Kingston Lacy in Dorset, England. The hieroglyphs included two different cartouches and the Greek inscription at the base of the obelisk mentioned Ptolemy and Cleopatra. He noticed that one of the cartouches was identical to that deciphered as Ptolemy by Young, and surmised that the other corresponded to Cleopatra. Bankes had lithographs made of the inscriptions, annotated with his idea about the cartouches, and distributed them in 1821. When Champollion received a copy he made the decipherment shown in Figure 3.

There was a remarkable degree of similarity between the values from the two names, except for the △ and ≣ signs which he explained as being homophones—they could each represent the same sound (*t* in this case).

<sup>5</sup> If you have taken any science courses you have probably heard of Young's Modulus and Young's Rings.

Hieroglyph	Champollion's value	Hieroglyph	Champollion's value
□	p	△	c
▷	t	◁	l
↶	o	↓	e
↷	l	↶	o
≡	m	□	p
≣	e	↗	a
↖	s	↙	t
		○	r
		⤒	a

**Figure 3:** Champollion's decipherment of Banks' Ptolemy and Cleopatra cartouches

He then looked at other cartouches to see if he could generate recognisable names from them by applying these sound values. The first one he tried was:



He was able to spell this out as *al?se?tr?*, which seemed to match the Greek *alksentrs* (Alexander), thus giving him three more sign values. Further cartouches both confirmed his values and gave new ones. One nagging thought was that only foreign names might be spelt phonetically but, among oth-

ers, this cartouche showed that this was not the case. Champollion knew that the ibis,  was the symbol of the god Thoth and he read the cartouche as *Thoth-mes*, an old Egyptian name.

As he matched more signs with sound values he was increasingly able to read the hieroglyphic texts as well as the names in the cartouches, and eventually could identify the Egyptian language as Coptic. Champollion's work laid the foundation for Egyptology as it is known today.

### 3.3 Linear B

The *Linear B* script was a syllabary that was used during the period approximately between 1600 and 1200 BC. Most of the examples come from Crete, particularly Knossos, but there are some from the Greek mainland.

The script consists of some 60 basic signs, 16 optional signs, and about 11 signs that have yet to be deciphered. The script also had signs for numbers (1–1000), and signs for various kinds of weights and measures. There were also sets of signs for different

**Figure 4:** Example of a Linear B text and partial interpretation

kinds of animals, such as horses and pigs, and for trade goods, such as pots or wool.

Clay tablets bearing the script were found by Sir Arthur Evans (1851–1941) while excavating the ruined Minoan palace at Knossos in Crete, starting in 1900. The tablets were usually small enough to be held in the hand, the largest being about six inches across, ten high and an inch thick. The tablets were accountancy records of some kind and he did work out their numeric systems but not much more than that. As an example, Figure 4 shows, on the left side, the text of a fairly typical tablet. On the right side is an interpretation of ፩, ፪, and ፫, which look rather like an addition sum, where ፩, ፪, and ፫ might be units in a non-metric system, like fluid

	Word A	Word B
Case 1	𒋩𒀭𒂗	𒋩𒀭𒂗
Case 2	𒋩𒀭𒂗	𒋩𒀭𒂗
Case 3	𒋩𒀭	𒋩𒀭

**Figure 5:** Two of Alice Kober's triplets

ounces, pints and gallons.<sup>6</sup> With a modicum of effort it can be shown that ፩ = 3 ፻ = 18 ፻.

There were many tablets like these where the last line started with either 𒋩 or 𒋩, and it was reasonable to assume that these words meant something like ‘total’.

On one tablet to do with listings of horses Evans noticed a pair of signs, ㄣ+, which matched the Cypriot signs ˻+ reading *po-lo* (see Table 8), similar to the Greek *polos* for foal. He was convinced that the Cretans spoke an unknown language, which he called Minoan, a theory that he held throughout his life, going to great lengths to disparage anyone who did not agree. Evans guarded his finds somewhat jealously and made little publicly available for others to work on. It was not until 1952, well after his death, that descriptions of his tablets were published.

However, another trove of Linear B tablets had been unearthed at Pylos, on the Greek mainland, by the American Carl W. Blegen in 1939. These were published in 1951 and would probably have been available earlier if the world had not been consumed with the other events of 1939 and later.

Michael Ventris, a British architect, had been fascinated by Linear B since he was a schoolboy and devoted much of his spare time in trying to decipher it. Initially there was no success because of the paucity of material to work on, but the publication of the Pylos tablets changed that.

In the meantime the script had been analysed by various scholars, the signary had been established, and lists had been made of which signs were most common at the start and end of words, and of how signs tended to group themselves. Dr. Alice E. Kober (1907–1950), a classicist at Brooklyn College, had noticed groups of signs where all but the last one or two signs in a word were the same, and thought that this might mean that the language captured by Linear B was inflectional, like Latin or occasionally English as in ‘I write’ but ‘he writes’. In particular she noted words that appeared

<sup>6</sup> The British and the Americans agree that there are eight pints to one gallon, but, perhaps to the chagrin of Texans, there are 20 fluid ounces to a British pint and only 16 to an American pint.

in three forms, as illustrated in Figure 5, which became known as ‘Kober’s triplets’.

Ventris had done his own analysis of the script and had come to the conclusion that it was a syllabary. He argued that the difference between the words 𒋩 or 𒋩 for total might be due to gender differences in an inflectional language as the first form occurs with the ideogram for man, and the other with the ideogram for woman. If this were the case then the consonants in ㄣ and ӱ were probably the same but the vowels were different. By analysing a number of words in this way he was able to start building up a grid where the signs in each row had the same consonant and those in the same column had the same vowel. It was still a long road, though, from having the signs coordinated in this fashion to being able to read them.

From Kober’s work he noticed that there were groups on the Knossos tablets that were not on the Pylos tablets, and made a bold leap to thinking that they might be the names of places on Crete. He suggested that the signs 𒋩+ might be the word for the Greek *Knosos* (Knossos) and 𒋩+ might be the word for the Greek *Amnisos*, the port for Knossos, and a few other names. When he applied the guesstimated values to the signs in the grid he was able to assign values to other signs and start ‘reading’ a few things. For example, 𒋩 and 𒋩 read *to-so* and *to-sa*, which were similar to the Greek *tosos* (masculine) and *tosa* (feminine) for ‘so much’ or ‘so many’. Ventris had originally whole-heartedly agreed with Evans that the language of Linear B was not Greek, but it was now appearing as though it might well be, especially if the Cypriot *polos* clue was included.

Ventris was familiar with the Greek of Homer (about the ninth century BC) from school but the Linear B tablets were much older than that. The words he was reading seemed similar to Homeric Greek but were not the same. For example there were several tablets from Pylos listing numbers of women, from the ideograph, often followed by two other words, 𒋩 and 𒋩, also with numbers and it was reasonable to assume that these words might be equivalent to ‘boys’ and ‘girls’. However he read them as *ko-wa* and *ko-wo*, whereas the Greek that he knew was *kourai* and *kouroi*. In general the Linear B spellings appeared incomplete, and even when filled out only close to Homeric Greek.

At this point he formed a partnership with John Chadwick, a lecturer in Classics at Cambridge University whose speciality was the early history of the Greek language. Together they worked out a consistent set of rules describing how Greek had changed

Table 4: The basic Linear B syllabary

	a	e	i	o	u
	𐄀	𐄁	𐄂	𐄃	𐄄
d	𐄅	𐄆	𐄇	𐄈	𐄉
j	𐄊	𐄋	𐄌	𐄍	𐄎
k	𐄏	𐄐	𐄑	𐄒	𐄓
m	𐄑	𐄑	𐄑	𐄑	𐄑
n	𐄑	𐄑	𐄑	𐄑	𐄑
p	𐄑	𐄑	𐄑	𐄑	𐄑
q	𐄑	𐄑	𐄑	𐄑	𐄑
r	𐄑	𐄑	𐄑	𐄑	𐄑
s	𐄑	𐄑	𐄑	𐄑	𐄑
t	𐄑	𐄑	𐄑	𐄑	𐄑
w	𐄑	𐄑	𐄑	𐄑	𐄑
z	𐄑	𐄑	𐄑	𐄑	𐄑

between Mycenaean and Homeric times. In other words, if you took an arbitrary Linear B tablet, deciphered it and then applied their rules the result would be Homeric Greek. In 1953 they jointly summarised their work in an article entitled ‘Evidence for Greek dialect in Mycenaean archives’ in *The Journal of Hellenic Studies*. Their theory was completely unexpected and its reception was mixed, to say the least. However it was soon dramatically confirmed.

The American excavation at Pylos had resumed in 1952 after the break for the Second World War. More Linear B tablets were found and stored for later reading. In the spring of 1953 the leader of the team, Carl Blegen, returned to Greece armed with an advance copy of Ventris and Chadwick’s article. Among the newly found tablets was a large one with pictures of three-legged cauldrons, pictures of a number of jars with differing numbers of loops (handles) on top, and Linear B inscriptions. When Blegen applied Ventris’ decipherment he read *tr-ri-pode*, almost identical to the Greek *tripodes* for three-legged cauldron. Next to the jars with three loops he read *ti-ri-o-we-e* or *ti-ri-jo-we*, and by the jars with four loops *qe-to-ro-we*. The Greek for three in compounds is *tri-* and experts in archaic Greek could accept that *quetro-* would be four in compounds. Once the ‘tripod’ tablet became known most scholars accepted that Ventris had deciphered Linear B, although a few die-hards even went so far as to suggest that the tripod tablet had been ‘planted’ at Pylos!

Table 4 shows the signs in the basic syllabary. Although Linear B was used for writing Greek, there

Table 5: The Proto-Semitic signary

Name	Glyph	Sound	Meaning
alpu	𐎁𐎂	,	ox
betu	𐎁𐎂	b	house
	݂	g	throw stick?
	݂݂	d	fish?
	=	z	
	܂܂	h	man?
wawwu	܂	w	hook, peg
hotu	܂܂	h	fence
	܂	t?	twisted flax
yadu	܂܂	y	hand, arm
kappu	܂܂	k	palm of hand
lamdu	܂܂	l	ox goad
mayyuma?	܂܂	m	water
nahasu	݂	n	snake
emu	܂܂	,	eye
	܂	s?	
	܂܂	p	leg/foot?
	܂܂	š?	plant?
	܂܂	q?	knot?
rasu	܂܂	r	head
	܂	š	lotus pool?
	܂܂	?	
tawwu	܂	t	mark

is no other relationship between this ancient script and the Greek alphabet.

### 3.4 Proto-Semitic

Around 1600 BC there were alphabetic scripts in use in the Middle East that are variously called Proto-Sinaitic, Proto-Canaanite, etc. I have lumped these together into a Proto-Semitic font. Several of the signs in this alphabet are obviously derived from Egyptian hieroglyphs, and it may have been a precursor to the Phoenician script.

The alphabet consisted of 23 letters, some of which had alternate forms. Writing was generally from left to right, but could be vertical or in other directions. Table 5 shows the signary, although there is not a complete consensus on this.

### 4 Phoenician

The Phoenicians initially wrote right to left or left to right. The alphabet consisted of 22 letters although a 23rd glyph was used as the *vav* (or *vau*) character, which had two forms: ܂ and ܂. Around 1100 BC the Phoenician alphabet had stabilised and the writing direction was finally fixed as right to left.

Table 6: Evolution of the Phoenician script

Name	Meaning	Hiero.	Proto.	Phoen.	Sound
aleph	ox	𐎐	𐎐	𐤀	'
beth	house	𐎑	𐎑	𐤁	b
gimel	camel	𐎒	𐎒	𐤂	g
daleth	door	𐎓	𐎓	𐤃	d
he	window?	𐎔	𐎔	𐤄	h
vav	nail	𐎕	𐎕	𐤅, 𐤇	w
zayin	dagger?	𐎖	=	𐤇	z
heth	fence?	𐎗	𐎗	𐤉	ħ
teth		𐎘	𐎘	ߵ	t
yod	hand	𐎙	𐎙	߱	y
kaph	palm of the hand	𐎚	𐎚	߲	k
lamed	ox goad	𐎛	𐎛	߳	l
mem	water	܂	܂	ߴ	m
nun	fish	܂	܂	ߵ	n
samekh	prop or post			߶	s
ayin	eye	܂	܂	܂	'
pe	mouth	܂ ?	܂	܂	p
sade			܂	܂	ṣ
qoph	knot?		܂	܂	q
resh	head	܂	܂	܂	r
shin	teeth	܂	܂	܂	ṣ̌
tav	mark or cross	܂	܂	܂	t

Table 6, which is somewhat speculative in some cases, illustrates how the Phoenician script (may have) developed from the Proto-Semitic script; the name and, where known, the meaning of the Phoenician glyph is given, as is the transliterated sound value. It also shows how some of the Proto-Semitic glyphs may have been inspired by the Egyptian hieroglyphs; in some cases the derivation is obvious.

## 5 Later Western scripts

Among the later Western scripts—those developed after 1000 BC—Greek, Etruscan and Latin are the direct ancestors of the English alphabet. Others, such as Cypriot and Runic, are off by themselves.

### 5.1 Greek

Initially the Greeks used the Phoenician alphabet and also wrote right to left but by the 7th century BC it became boustrophedon and around 500 BC they finally settled on writing left to right. The Greeks added new letters to the Phoenician abecedary so that around the 6th century BC their alphabet consisted of 26 characters. The Y form of the Phoenician *vav* became the Greek *upsilon* while the F form

of *vav* became the Greek *digamma*. The names of the letters lost their meanings and instead effectively stood for the pronunciation of the letter. The Greeks also added the *psi*, *phi* and *omega* characters. Several different glyphs were used for each character, depending on geographical location, whether on the mainland or around the Aegean Sea. One variety of the 6th century BC alphabet looked like this:

ΑΒΓΔΕΦΙΘ⊕ΙΚΤΜΝΞΟΓΡΣΤΥΧΩΨΩ

In 403 BC the Athenian citizens codified the alphabet with the glyphs looking much as they do today. The *digamma* and the *qoph* characters were dropped from the abecedary, thus leaving the 24 characters that we are now accustomed to. The 4th century BC alphabet was like this:

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΧΦΨΩ

### 5.2 Etruscan

The Etruscans, forerunners of the Romans in Italy, based their alphabet on the Greek abecedary, but they continued to write right to left as the Phoenicians had, so their glyphs were mirrored with respect

Table 7: The changing alphabets

Phoenician	Greek			Etruscan	Roman	Modern
	600 BC	394 BC	Modern			
𐤁	Α	Α	Α	Α	Α	Α
𐤂	Β	Β	Β	Β	Β	Β
𐤃	Γ	Γ	Γ	Γ	Ϲ G	Ϲ G
𐤄	Δ	Δ	Δ	Δ	Δ	Δ
𐤅	Ε	Ε	Ε	Ϛ	Ϛ	Ϛ
𐤆	Ϛ	Ϛ	Ϛ			
𐤇	Ϛ	Ϛ	Ϛ		V Y	U V W Y
𐤈	Ϛ	Ϛ	Ϛ	Ϛ	F	F
𐤉	Ϛ	Ϛ	Ϛ	I	Z	Z
𐤊	Ϛ	Ϛ	Ϛ	Ϛ	H	H
ࡓ	ࡓ	ࡓ	ࡓ	ࡓ		
ࡔ	ࡔ	ࡔ	ࡔ	ࡔ	I J	I J
ࡕ	ࡕ	ࡕ	ࡕ	ࡕ	K	K
ࡖ	ࡖ	ࡖ	ࡖ	ࡖ	L	L
ࡗ	ࡗ	ࡗ	ࡗ	ࡗ	M	M
ࡘ	ࡘ	ࡘ	ࡘ	ࡘ	N	N
࡙	࡙	࡙	࡙	࡙	X	X
࡚	࡚	࡚	࡚	࡚	O	O
࡛	࡛	࡛	࡛	࡛	P	P
࡜				࡜		
࡝	࡝			࡝	Q	Q
࡞	࡞	P	P	࡞	R	R
࡟	࡟	࡟	࡟	࡟	S	S
ࡠ	ࡠ	ࡠ	ࡠ	ࡠ	T	T
ࡡ	ࡡ	ࡡ	ࡡ	ࡡ		
ࡢ	ࡢ	X				
ࡣ	ࡣ	ࡣ	ࡣ	ࡣ		
ࡤ	ࡤ	ࡤ	ࡤ	ࡤ		
ࡥ	ࡥ	ࡥ	ࡥ	ࡥ		
ࡦ	ࡦ	ࡦ	ࡦ	ࡦ		
ࡧ	ࡧ	ࡧ	ࡧ	ࡧ		
ࡨ	ࡨ	ࡨ	ࡨ	ࡨ		
ࡩ	ࡩ	ࡩ	ࡩ	ࡩ		

to the Greek ones. The Etruscan script was used until the first century AD, although the Etruscans had disappeared well before then.

The Etruscan abecedary originally consisted of 26 characters but by about 450 BC it had decreased to only 20. One interesting glyph that was dropped looked like the digit 8, and denoted an ‘f’ sound. Some scholars have surmised that this was the ancestor of the mathematical sign for infinity ( $\infty$ ). In the 8th century BC the alphabet looked like:

A B C D E F G H I K L M N O P Q R S T V X Y Z

As far as decipherment is concerned, Etruscan is a known script used for writing an unknown language. Scholars are able to read aloud the script but they do not understand the language. Apart from proper names only about a couple of dozen words are known, and these are mainly to do with family relationships like ‘son’ and ‘father’.

### 5.3 Latin

The Romans based their script on the Etruscan one,

again adding and dropping characters. They added the G and Y characters but dropped *theta*, *psi*, and *phi* to end up with a 23 character abecedary, although they rarely used the H, K, and Z characters. So, their alphabet looked like:

A B C D E F G H I K L M N O P Q R S T V X Y Z

The lettering used in this last example is a copy of the capitals engraved on the Trajan Column in Rome which was erected in 114 AD. Many typographers believe that these represent the high point in the Roman artistic legacy. Unfortunately they do not reproduce well in the size shown here. In real life the inscription is about six feet above eye level and there are six lines of text. The letters on the top line are somewhat over four inches tall, decreasing to about three inches on the lowest line, which presents the illusion that they are all of the same height when observed from the normal viewpoint.

Summarising, Table 7 shows the 1100 years of development of the Latin alphabet.

Table 8: The Cypriot syllabary

	a	e	i	o	u
g	*				
j	χ				
k	χ	θ			
l		↑	≤		
m	↓	↓	≤		
n		χ	χ		
p		χ	χ		
r		†	χ		
s		†	χ		
t		†	χ		
w		†	χ		
x	(		χ		
z	)				

## 5.4 Cypriot

The Cypriot script was a syllabary used in Cyprus during the approximate period between 1000 and 200 BC for writing Greek. It has a relationship to Linear B as it includes some of the same signs. Towards the end of its life few people could read the script, so inscriptions were written using both the syllabary and the Greek alphabetic characters. These bilinguals made it relatively easy to decipher the script, a task that was essentially completed by the last quarter of the nineteenth century.

Like Linear B, the Cypriot script has no relationship with the Greek abecedary apart from the fact that both can be used for writing the same language.

Table 8 shows the Cypriot syllabary.

## 5.5 Runic

The runic alphabet, which is not shown in Table 7, is known as *futhark* after its initial characters. It was used, with local variations, in the Germanic, Scandinavian and Anglo-Saxon countries until shortly after printing was invented. Scholars are unclear as to the origins of the futhark abecedary, but there are obvious correspondences between some of the glyphs and the Phoenician and Etruscan ones, while others have no resemblance at all.

Like the Phoenician alphabet, the names of the futhark characters have meanings. The ordering of the characters, together with their names and meanings, is shown in Table 9.

It is very noticeable that the letter ordering is completely different from any of the other abecedaries in Table 7. It is interesting to speculate whether the ordering of an original abecedy depends on the frequency of use of the letters, or those with the most important meanings have priority.

The *wen* character (ƿ) is no longer used in English, but does indicate that the Anglo-Saxons had need of a ‘W’. The *thorn* character (þ) is like *theta* in that it represents the ‘th’ sound. Early printers usually did not have a þ, so they used a ‘Y’ character instead. From this practice comes the modern affectation of naming something like ‘Ye Olde Pub’ instead of ‘The Old Pub’. Also, it has the *ger* character (ȝ) which corresponds to the modern ‘J’ sound—‘J’ did not appear in the Latin alphabet until about the mid-1500’s.

## 6 Later Semitic scripts

This section includes scripts that were invented after 1000 BC and used in the Middle East.

Table 10 shows the evolution of the modern Hebrew and Arabic<sup>7</sup> scripts.

## 6.1 Old Persian

It is believed that the Old Persian cuneiform script was invented on the order of the Persian king Darius I for use on royal monuments. The script was only in use between about 500 and 350 BC.

Old Persian was a syllabary with 36 glyphs. There were also 5 ideographs, some with multiple forms, for the words *king*, *country*, *earth*, *god* and *Ahuramazda* (the Persian god), together with a word divider. Numerals were also represented.

Somewhat surprisingly the decipherment of Old Persian led directly to the decipherment of the far older Sumerian and Babylonian cuneiform scripts. The basic work was done by Georg Friedrich Grotefend (1775–1853), a high school teacher in Göttingen and was completed by Henry Rawlinson [Adk04]. It was generally assumed that because of the limited number of signs the script was alphabetic, the slanting wedge was probably a word divider and it was written left to right.

Grotefend started with two texts which were inscribed above doorways in the ruined city of Persepolis. The first was:

၁၁၁၂ ၂၀၁၃ ၁၁၁၃ ၁၁၁၄

<sup>7</sup> Arabic actually has 28 characters and many more glyphs as they change depending on the position of the character in a word, but I have only shown the characters that were derived from the Phoenician script.

Table 9: The Futhark abecedary

Glyph	Name	Meaning	Glyph	Name	Meaning
ᚠ	<i>feof, feh, fe</i>	wealth	ᚢ	<i>hic, ih, eoh</i>	
ᚢ	<i>ur, hur</i>	auroch	ᚦ	<i>peord</i>	
ᚦ	<i>thorn</i>		ᛵ	<i>eohlx</i>	
᛽	<i>æsc, os</i>	oak tree	᛻	<i>sigel</i>	sun
ᚱ	<i>rad, rat</i>	riding	ᛏ	<i>tir</i>	name of a star?
ᚼ	<i>ce, kaun</i>	torch	ᛒ	<i>berc, birth</i>	birch tree
ᚹ	<i>gebu, gyfu</i>	gift	ᛖ	<i>hæc, ech, eh</i>	horse
ᛮ	<i>wen</i>	joy	ᛘ	<i>man</i>	man
ᚷ	<i>hegl, hagal</i>	hail	ᛚ	<i>lagu</i>	water or sea
ᚫ	<i>nyd, nod</i>	need or hardship	ᛚ	<i>ng</i>	
ᛁ	<i>is</i>	ice	ᛞ	<i>dag, dæg</i>	day
ᚪ	<i>ger, yr, ar</i>	year	ᛟ	<i>o, oe</i>	mouth
			.		punctuation

Table 10: Evolution of Middle Eastern scripts

Name	Hiero.	Proto.	Phoen.	Aram.	Nab.	Hebrew	Arabic
aleph	א	𐤁	𐤁	𐤁	𐤁	א	أ
beth	ב	𐤂	𐤃	𐤃	𐤃	ב	ب
gimel	ג	𐤄	𐤂	𐤂	𐤂	ג	ج
daleth	ד	𐤅	𐤆	𐤆	𐤆	ד	د
he	ה	𐤈	𐤉	𐤉	𐤉	ה	ه
vav	ו	𐤊	𐤋	߱	߱	ו	و
zayin	ז	=	𐤌	߲	߲	ז	ز
heth	ח	𐤍	ߴ	ߴ	ߴ	ח	خ
teth	ט	ߵ	߶	߶	߶	ט	ط
yod	ي	߷	߸	߸	߸	ي	ي
kaph	ك	߹	߻	߻	߻	ك	ك
lamed	ל	߻	߻	߻	߻	ל	ل
mem	מ	߻	߻	߻	߻	מ	م
nun	נ	߻	߻	߻	߻	נ	ن
samekh	ס	߻	߻	߻	߻	ס	س
ayin	ע	߻	߻	߻	߻	ע	ع
pe	פ	߻	߻	߻	߻	פ	ف
sade	צ	߻	߻	߻	߻	צ	ص
qoph	ק	߻	߻	߻	߻	ק	ق
resh	ר	߻	߻	߻	߻	ר	ر
shin	ש	߻	߻	߻	߻	ش	ش
tav	ת	߻	߻	߻	߻	ת	ت

And the second was:

At the time of the Achaemenid Dynasty (559–331 BC) in Persia the Zoroastrians were prominent together with their sacred text, the *Zend-Avesta*. This had recently become known in Europe as well as some Pehlevi<sup>8</sup> texts translated and published by Silvestre de Sacy in 1793 in his *Mémoires sur diverses Antiquités de Perse*. Grotefend knew of a formula from these texts that went ‘X, king . . . , king of kings, son of Y . . .’. He guessed that the Persepolis inscriptions might embody the same formula and that the second sign group in each inscription (𐎧𐎰𐎠 𐎭 𐎭 𐎫 𐎭 𐎫 𐎭) might be the word for king, and the later repetitions equate to ‘king of kings’.

In the first inscription the first group  


should be the name of a king, say A for the sake of argument, and similarly in the second inscription the first group

ଶ୍ରୀ ରତ୍ନାଳୀ

would be the name of another king, say B. Grotefend noted that A also occurred in B's inscription, so that the second one might read 'B, king..., king of kings, son of king A...'. He guessed again that the inscriptions might have something to do with Darius and Xerxes, and in which case A would be Darius and B his son Xerxes.

Using forms of Darius and Xerxes which he derived from an amalgamation of Avestan, Greek and Hebrew he suggested that the signs in the two names should be read as *d/a/r/h/e/u/sh* for Darius and *kh/sh/h/e/r/sh/e* for Xerxes. With these values the signs for king would read *kh/sh/e/h/?/?/h*. In the Avesta he found the kingly title *khscheio* and he took this as confirmation that he was on the right track and the language of the inscriptions was Avestan. He was correct.

With the limited number of inscriptions available, work could not proceed further. However, huge inscriptions were discovered on the side of a moun-

Table 11: The Old Persian syllabary

	a	i	u		a	i	u
k	ກ	ີ	ູ	f	ຝ	ີ	ູ
x	ຂ	ີ	ູ	b	ົ	ີ	ູ
g	ງ	ີ	ູ	m	ົ	ີ	ູ
c	ຈ	ີ	ູ	y	ິ	ີ	ູ
j	ຈ	ີ	ູ	r	ີ	ີ	ູ
t	ຕ	ີ	ູ	l	ີ	ີ	ູ
th	ທ	ີ	ູ	v	ີ	ີ	ູ
ça	ຈາ	ີ	ູ	s	ີ	ີ	ູ
d	ດ	ີ	ູ	ša	ີ	ີ	ູ
n	ນ	ີ	ູ	z	ີ	ີ	ູ
p	ປ	ີ	ູ	h	ີ	ີ	ູ

tain near Behistun in western Iran. Henry Rawlinson (1810–1895), who was serving as the British military advisor to the brother of the Shah of Iran, began to copy the inscriptions in 1835. Among other things, this involved dangling from ropes to get at some of the texts as they were on a cliff more than 300 feet up. In all, it took him ten years to copy 414 lines. It turned out that the inscriptions were trilinguals, in Old Persian, Elamite and Babylonian. Rawlinson effectively completed the decipherment of Old Persian which then enabled him to tackle the Babylonian, which became the key to other cuneiform scripts.

Table 11 shows the Old Persian syllabary.

Some of the other glyphs include: ☰ (king), ☱ (country), ☲ (earth), ☳ (god), and ☴ (Ahuramazda).

Examples of numerals include: ፩ (1), ፪ (2), ፲ (10), ፳ (20), ፻ (100).

As a final note, the transliteration of the second of the two inscriptions that Grotewold tackled is:

*xa-ša-ya-a-ra-ša-a-: xa-ša-a-ya-tha-i-ya-: va-za-ra-  
ka-: xa-ša-a-ya-tha-i-ya-: xa-ša-a-ya-tha-i-ya-a-  
na-a-ma-: da-a-ra-ya-va-ha-u-ša-: xa-ša-a-ya-tha-i-  
ya-ha-ya-a-: pa-u-ca-: ha-xa-a-ma-na-i-ša-i-ya-a-:*

and which, when translated, means:

Xerxes, the great king, the king of kings, the son of  
Darius the king, an Achaemenian.

## 6.2 Aramaic

The Aramaic script is an early offshoot from the Phoenician and was used between about the tenth and second centuries BC in the Middle East. The Aramaic script also branched and led to both modern Arabic and Square Hebrew scripts.

---

<sup>8</sup> Pehlevi (or Pahlevi or Pahlavi) was a Persian dialect of the Sassinide period (3rd–7th century AD).

The script is alphabetical and consists of 22 consonants.

אַלְמָנָה וְלִבְנָה כְּפָרָה וְרָבָה

### 6.3 Nabatean

The Nabatean script is an offshoot of the Aramaic script and was in use in an area centered around Petra — the ‘rose-red city half as old as time’ — roughly during the period between the fourth century BC and the fourth century AD. It is a direct ancestor of the modern Arabic script.

Like other Semitic scripts it is alphabetical and consists of 22 consonants.

לְבָנָה כְּפָרָה וְלִבְנָה כְּפָרָה

## 7 Remarks

The result from formula 1 is certainly an approximation. I applied it to the two Old Persian texts in section 6.1 which together contain 150 signs with 22 different kinds. The estimated number of signs is

$$S = 150^2 / (150 - 22) - 150 = 25.78$$

which is somewhat under the actual value of 36 for the syllabary. The much shorter made up text on page 202 consisting of 70 characters of 29 different kinds gives

$$S = 70^2 / (70 - 29) - 70 = 49.5$$

which is a significant overestimate. However, combining the three texts gives 220 total characters with 33 different kinds, resulting in

$$S = 220^2 / (220 - 33) - 220 = 38.8$$

which is close to the actual number.

The books listed below are among the more accessible sources describing the development of the alphabet and the Latin script, and of decipherments of archaic scripts.

The fonts used in this article can be obtained from CTAN (the Comprehensive TeX Archive Network). The Arabic script came from Klaus Lagally’s `arabtex` package in the `languages` area, and similarly the Hebrew script is from `hebrew/hebtex` in the same area. The Trajan font is in the `fonts/trajan` directory. All the other scripts are in the `fonts/archaic` directory.

## References

- [Adk04] Lesley Adkins. *Empires of the Plain: Henry Rawlinson and the Lost Languages of Babylon*. Harper Perennial, 2004.
- [Bud89] E. A. Wallis Budge. *The Rosetta Stone*. Dover, 1989.
- [Cha58] John Chadwick. *The Decipherment of Linear B*. Cambridge University Press, 1958.
- [Cha87] John Chadwick. *Linear B and Related Scripts*. Reading the Past. University of California Press, 1987.
- [CM98] Mark Collier and Bill Manley. *How To Read Egyptian Hieroglyphs*. University of California Press, 1998.
- [Coe99] Michael D. Coe. *Breaking the Maya Code*. Thames & Hudson, 1999.
- [Dav87] W. F. Davies. *Egyptian Hieroglyphs*. Reading the Past. University of California Press, 1987.
- [Fri89] Johannes Friedrich. *Extinct Languages*. Dorset Press, 1989. (Translated from the original German *Entzifferung Verschollener Schriften und Sprachen* by Frank Gaynor).
- [Gor87] Cyrus H. Gordon. *Forgotten Scripts: Their Ongoing Discovery and Decipherment*. Dorset Press, 1987.
- [Hea90] John F. Healey. *The Early Alphabet*. Reading the Past. University of California Press, 1990.
- [Ifr00] Georges Ifrah. *The Universal History of Numbers*. John Wiley & Sons, 2000. (Originally published as *Histoire universelle des chiffres*. Robert Laffort, Paris, 1994.).
- [Mal91] J. P. Mallory. *In Search of the Indo-Europeans: Language, Archaeology and Myth*. Thames and Hudson, 1991.
- [Nak80] Akira Nakanishi. *Writing Systems of the World: Alphabets, Syllabaries, Pictograms*. Charles E. Tuttle Company, 1980.
- [Pop99] Maurice Pope. *The Story of Decipherment: From Egyptian Hieroglyphs to Maya Script*. Thames & Hudson, 1999.
- [Rob95] Andrew Robinson. *The Story of Writing: Alphabets, Hieroglyphs & Pictograms*. Thames & Hudson, 1995.
- [Rob02] Andrew Robinson. *Lost Languages: The Enigma of The World’s Undeciphered Scripts*. McGraw-Hill, 2002.
- [Wal87] C. B. F. Walker. *Cuneiform*. Reading the Past. University of California Press, 1987.

◊ Peter Wilson  
18912 8th ve. SW  
Normandy Park, WA 98166  
USA  
[herries.press \(at\) earthlink.net](mailto:herries.press(at)earthlink.net)

# Fonts

## Advanced font features with X<sub>E</sub>T<sub>E</sub>X — the `fontspec` package\*

Will Robertson

### Abstract

This paper describes the `fontspec` package for the X<sub>E</sub>T<sub>E</sub>X–L<sub>A</sub>T<sub>E</sub>X format. This package provides a high level interface for font selection and configuration of OpenType and other fonts.

### 1 Introduction

X<sub>E</sub>T<sub>E</sub>X is an extended T<sub>E</sub>X program written by Jonathan Kew, and has been introduced recently in this journal [4]. It is currently available only on Apple’s Mac OS X, but there is considerable interest (and much work to be done!) in making it a cross-platform application. The main advantages it holds over its contemporaries are support for Unicode input and direct access to fonts installed in the operating system. No additional support files are necessary in order to install such fonts, which are accessed via an extended `\font` primitive. This primitive also provides access to rich font features available in either the OpenType format, or the ‘Apple Advanced Typography’ (AAT) format.<sup>1</sup> A typical example, in plain X<sub>E</sub>T<sub>E</sub>X, of choosing a font with old-style figures in these two formats is shown in example 1.

The `fontspec` package is an implementation of a high level interface for L<sub>A</sub>T<sub>E</sub>X users of X<sub>E</sub>T<sub>E</sub>X to access feature-rich fonts in the framework of the familiar NFSS. Furthermore, it obviates the need for custom-written font definition files required for font installation.

This paper will introduce the `fontspec` package with some side commentary on the advanced font formats it supports. The first half of the paper will cover the user interface, covering font installation, font selection, and font *feature* selection. The second half discusses the implementation details, shortcomings, and future of the package.

\* Version 1.9 of the package was under construction while this paper was finalised; some additions may yet make it into the final version, in which case I apologise in advance for the incomplete information here. The package documentation will always be up to date, of course.

<sup>1</sup> The AAT format may be considered to be approximately equivalent to a combination of the Multiple Master and OpenType formats.

---

Example 1: Plain X<sub>E</sub>T<sub>E</sub>X OpenType and AAT font selection with old-style figures.

---

```
\font\fonta="Adobe Garamond Pro:+tonum" at 12pt
\fonta OpenType old-style figures: 0123456789

\font\fontb="Apple Chancery:
           Number Style=Old Styles" at 12pt
\fontb AAT old-style figures: 0123456789

OpenType old-style figures: 0123456789
AAT old-style figures: 0123456789
```

---

### 2 Motivation

With X<sub>E</sub>T<sub>E</sub>X, users have easy access to a multitude of typefaces in Plain T<sub>E</sub>X. But writing the font definition files for L<sub>A</sub>T<sub>E</sub>X was cumbersome and proved a fair obstacle for day-to-day use. I started working on a solution, which was originally simply to create the `.fd` files necessary to access every Mac OS X font in L<sub>A</sub>T<sub>E</sub>X, but I quickly tired of the tedium, disenchanted with this non-general solution to the problem. Furthermore, the scope of font features provided by feature-rich fonts quickly demonstrated the NFSS<sup>2</sup> insufficient for the task of incorporating every permutation of font features a user might desire.

Secondly, AAT font features are accessed by referring to a specific string defined on a per-font basis, with consistency between fonts kept between ‘feature codes’ rather than the value of the string. For example, example 1 uses the strings ‘Number Style’ and ‘Old Styles’ to select lowercase numbers in Apple Chancery, but for Hoefler Text one would write ‘Number Case=Lowercase numbers’. Note that such a system works well in a graphical program in which font features are selected interactively from a list; each feature may be described exactly as the font designer would wish (and this is indeed an advantage for more esoteric features). Unfortunately, for a batch program like X<sub>E</sub>T<sub>E</sub>X, this flexibility is a burden.

Finally, after version 0.8, X<sub>E</sub>T<sub>E</sub>X began using the ICU renderer<sup>3</sup> to support OpenType font features in addition to the Mac OS X-native features it supported from the beginning. Now the poor users had to cope with not only a different system for applying font features, but also OpenType’s cryptic abbreviations for them. At this time, `fontspec` was able to provide

---

<sup>2</sup> NFSS, the ‘new font selection scheme’ for L<sub>A</sub>T<sub>E</sub>X, isn’t actually that new. For readers unfamiliar with its concepts, the documentation file `fntguide` reveals all [12].

<sup>3</sup> International Components for Unicode, <http://icu.sourceforge.net>

a unified and consistent interface to fonts and their features in both formats, and offer a few other niceties along the way.

### 3 Font installation

To install a font in Mac OS X, the font file must be placed in one of the computer's Fonts folders (user, system, or network). Once a font is installed in Mac OS X it is immediately available to Xe<sup>T</sup>E<sub>X</sub>. No additional font support files of any kind are required for it to be loaded by a \font command. (A L<sup>A</sup>T<sub>E</sub>X user would still need .fd files for loading it with classical NFSS techniques.) Ease of font access is one of Xe<sup>T</sup>E<sub>X</sub>'s attractions, but no manipulation of the font properties can occur between the stages of obtaining the font and actually using it.

Compare this to the method T<sub>E</sub>X and its siblings uses, in which sub-optimal glyphs or kerning in the font may be rectified with customised font metrics and virtual fonts. In Xe<sup>T</sup>E<sub>X</sub>, it is much simpler for a *user* to install a new font, but it is less flexible if greater output quality is desired than the font alone gives and the user cannot, or may not (due to license restrictions), edit the font file itself.

It has always been a good idea, however, to use a suitably high-quality font from the beginning in order to avoid such hassles.

### 4 Font selection

One of the trickier topics a new user to L<sup>A</sup>T<sub>E</sub>X faces is font selection, although progress has been made with the PostScript NFSS bundle [11], and other freely available fonts, that provide simple<sup>4</sup> packages to select them. Xe<sup>T</sup>E<sub>X</sub> and the fontspec package make things similarly easy by referring to a font with its display name, rather than the cryptic 'Karl Berry' abbreviation, unnecessary in many cases these days.

On an individual basis, \fontspec selects font families:

```
\fontspec[font features]{font name}
```

This command loads the specified typeface and defines an NFSS family as appropriate with bold, italic, and small caps shapes (if available) for access with the familiar font-shape changing commands such as \itshape, \textsc, etc.<sup>5</sup> An example of selecting the 'Hoefler Text' family with these methods is shown in example 2.

<sup>4</sup> Indeed, too simple in some cases. The helvet package allows a scaling factor to load the font at any relative size, but almost all other font packages skip this sometimes-vital ingredient.

<sup>5</sup> Code inspired by Philipp Lehman's *The Font Installation Guide* [6] allows the combination of both italic and small caps shapes.

---

#### Example 2: NFSS family selection

---

```
\fontspec{Hoefler Text}
This is an example typeset in Hoefler Text.
\textit{Here is italic.} \textbf{And now bold.}
{\bfseries\itshape Bold italic, of course.}
\scshape Small caps, if available.
\itshape And italic small caps.
\bfseries Even bold italic small caps!
```

This is an example typeset in Hoefler Text. *Here is italic. And now bold. Bold italic, of course.*

SMALL CAPS, IF AVAILABLE. *AND ITALIC SMALL CAPS.*

**EVEN BOLD ITALIC SMALL CAPS!**

---



---

#### Example 3: Choosing the default font families.

---

```
\setromanfont{Baskerville}
\setsansfont[Scale=MatchLowercase]{Skia}
\setmonofont[Scale=MatchLowercase]{Monaco}
The \textsf{fontspec} package defines
the \verb|\fontspec| command.
```

The fontspec package defines the \fontspec command.

---

More usefully, the default document fonts (roman, sans serif, and typewriter), are chosen with the following commands, which have the same interface as \fontspec itself. They are \setromanfont, \setsansfont, and \setmonofont, and provide a more intuitive interface than such methods as

```
\renewcommand\rmdefault{family} .
```

Related commands are also available for specifying the text fonts for use in maths environments (*i.e.*, \mathrm and others).

The use of the default-font commands in example 3 also demonstrates the feature for automatic font scaling, which in this case keeps the lowercase letter heights consistent. Further explanation of the Scale feature occurs in section 5.1.

The fontspec package attempts to identify the accompanying small caps, bold, and italic faces for a selected font, but in the case that it fails or that more than one is available for use, they may be selected explicitly, with individual font features if desired. 'Old-fashioned' 8-bit fonts with separate small caps may be defined as a complete family in this way, as shown in example 4, which also demonstrates how a Multiple Master font instance can be conveniently defined for the bold series.

Finally, commands may be defined for efficiently switching between fonts:

---

Example 4: Choosing accompanying fonts.

---

```
\fontspec[SmallCapsFont = {Minion MM Small Caps
    & Oldstyle Figures},
    BoldFont = {Minion MM Roman},
    BoldFeatures = {Weight = 1.4},
    ]{Minion MM Roman}
Minion Roman 123 \\ \textsc{Minion SC 456} \\
\bfseries
Minion Bold 123 \\ \textsc{Minion Bold SC 456}
```

---

Minion Roman 123  
MINION SC 456  
Minion Bold 123  
MINION BOLD SC 456

---

`\newfontinstance\fontcs[features]{font name}`  
This defines the `\fontcs` control sequence (say) for selecting the particular font instance defined. It is more efficient than writing

```
\def\fontcs{\fontspec[features]{font name}}
```

because the feature processing only needs to be performed in the original definition, as opposed to every time the macro is expanded in the latter case.

## 5 Font feature selection

The package documentation covers all of the built-in features `fontspec` supports, with many examples; an interesting subset of these is presented here.

As previously mentioned, font features are defined in the optional argument to the various font commands. Inspired by the organisation of the AAT font features, these are separated into groups and use keyval comma-separated values to impose some sort of structure onto the large number of possible feature choices.

Two very important commands are associated with choosing font features:

```
\defaultfontfeatures{font features}
\addfontfeatures{font features}
```

The first, `\defaultfontfeatures`, is used to define features that will be applied implicitly to *all* subsequent font choices; for example, to request that all fonts use lowercase numbers.

Secondly, the command `\addfontfeatures` (the `s` is optional) selects features to use *in addition to* those already specified for the current font. The scope of this command is local to the current group; the default fonts are not redefined. This could be used, *e.g.*, in a hook to all tabular material that selects uppercase and fixed-width numbers regardless of the font in use; refer to example 5 for a proof-of-concept

---

Example 5: Selecting number styles by context.

---

```
\fontspec[Numbers={Proportional,OldStyle}]
    {\Skia}
In 1842, 999 people sailed 97 miles
in 13 boats. \par In 1923, 111 people
sailed 54 miles in 56 boats. \vspace{10pt}

{\addfontfeatures{Numbers={Monospaced,Lining}}}
\begin{tabular}{@{} c c c c @{}}
Year & People & Miles & Boats \\
\hline
1842 & 999 & 75 & 13 \\
1923 & 111 & 54 & 56 \\
\end{tabular}
```

---

In 1842, 999 people sailed 97 miles in 13 boats.  
In 1923, 111 people sailed 54 miles in 56 boats.

Year	People	Miles	Boats
1842	999	75	13
1923	111	54	56

---

implementation of this idea, which was requested by Toledo in his paper on feature-rich fonts [13].

The important point is that the tabular material requires no *a priori* knowledge of the font in use to format the numbers appropriately. The implementation of a class that takes advantage of such features is left as an exercise to the reader.

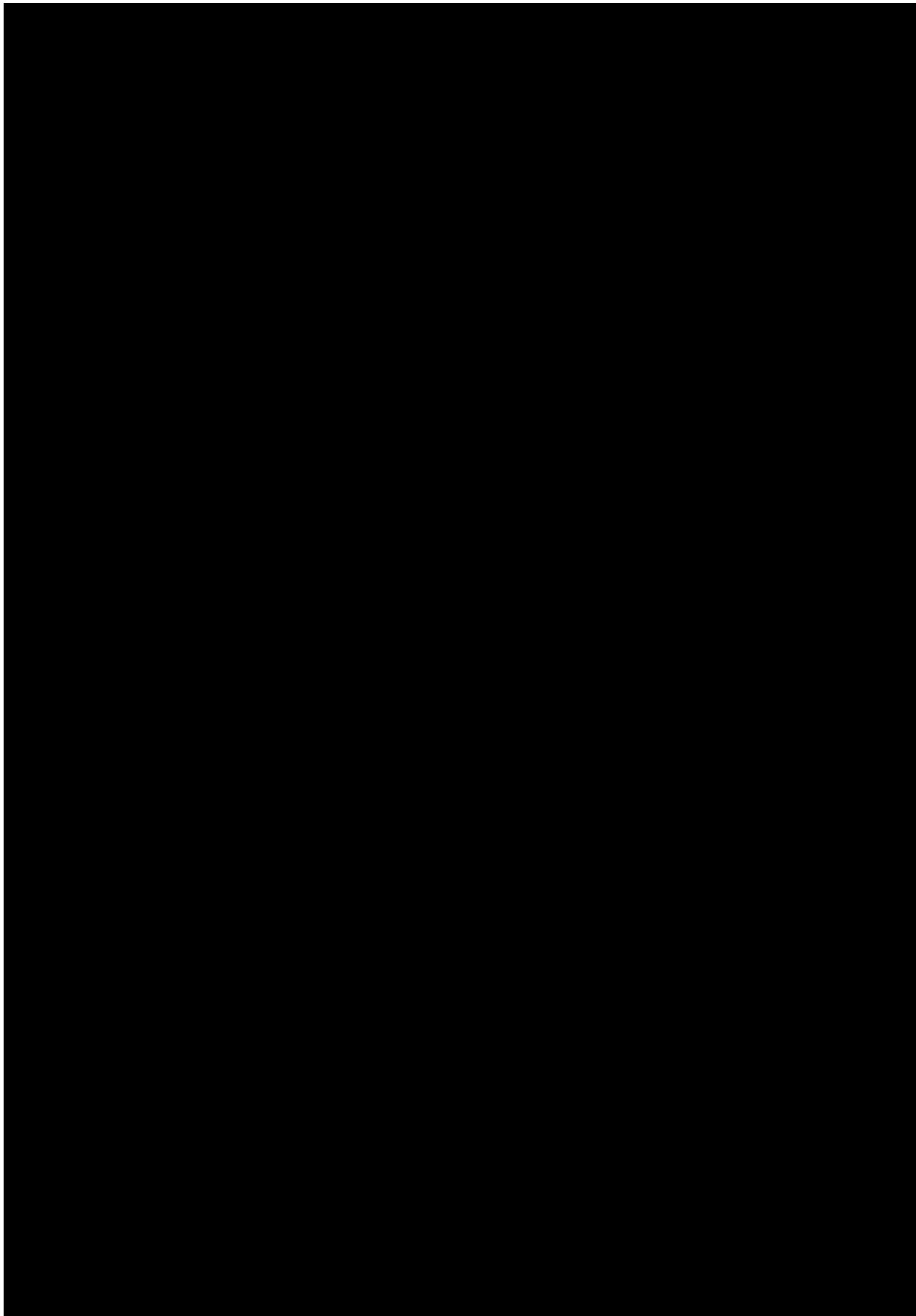
Another example of the `\addfontfeature` command is shown in example 6, in which vulgar fractions may be easily created on the fly.<sup>6</sup> This example also uses a macro from the `xunicode` package, by Ross Moore, which provides macros (backwards compatible with existing L<sup>A</sup>T<sub>E</sub>X conventions) for a large number of Unicode character slots; in this case, `\textfractionsolidus` refers to U+2044 FRACTION SLASH.

### 5.1 Features for all fonts

The first class of features is independent of the choice of font; that is, there are no restrictions on their use and they do not require specific font support.

The first is the most interesting, and deals with a specific feature of X<sub>H</sub>T<sub>E</sub>X. The `Mapping` feature allows a last minute Unicode remapping of the input stream, providing such features as non-standard

<sup>6</sup> In practise, as always, things are a bit more complex; the `\vfraction` macro shown is actually AAT-font specific due to differences with how such things are handled in OpenType. A more complex macro could be written to handle both cases transparently, however.



---

**Example 10:** Selecting an uppercase hyphenchar. Rather than doing it manually, an OpenType font feature performs this automatically.

---

```
\fontspec{Adobe Garamond Pro}      \TEXT
\addfontfeature{Letters=Uppercase} \TEXT
```

---

UPPERCASE TRACKING AND HYPHEN-CHAR  
UPPERCASE TRACKING AND HYPHEN-CHAR

---



---

**Example 11:** Ligature examples

---

```
\fontspec[Ligatures=NoCommon]{Hoefler Text}
  strict firefly \quad
\fontspec[Ligatures=Rare]{Hoefler Text}
  strict firefly

\fontspec[Palatino]
  Apple \quad
\fontspec[Ligatures=Logos]{Palatino}
  Apple
```

---

strict firefly    strict firefly  
Apple    

---

## 5.2 Features for fonts that support them

Ligatures are single glyphs that are used to represent many characters for reasons of elegance, decoration, or tradition. They are essential for typesetting many non-Western languages correctly. Some of the ligature features are shown in example 11.

Another example of shape-specific feature selection is shown in example 12, for the case of a stylistic variant in the roman form, and contextual swashes in the italic (a decidedly dubious combination). In this example, the `Contextuals` feature controls the contextual swashes on the lowercase letters, and `Alternate=1` activates swash caps for the uppercase.

Glyph variations can go on almost indefinitely for decorative typefaces; they are enumerated within the `Variant` feature. Apple Chancery is a font distributed with Mac OS X, a very small selection of whose variant glyphs are shown in example 13. For a more extreme example, see ‘Zapfino’, example 15.

Features for which I haven’t shown examples include: optical font sizes, fractions, various ideographic and alphabetic CJK features, including support for vertical typesetting, and OpenType script and language selection.

---

**Example 12:** Different forms for different shapes.

---

```
\fontspec[
  UprightFeatures = {Style = Engraved} ,
  ItalicFeatures = {
    Contextuals = {WordInitial, WordFinal} ,
    Alternate = 1 }
  ]{Hoefler Text}
[ABCD\ldots WXYZ] \\
\textit{Australian vegemite}
```

---



---



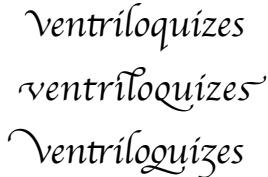
---

**Example 13:** Apple Chancery’s design complexities.

---

```
\fontspec[Variant=1]{Apple Chancery}
ventriloquizes \\
\fontspec[Variant=2]{Apple Chancery}
ventriloquizes \\
\fontspec[Variant=3]{Apple Chancery}
ventriloquizes
```

---



---

## 5.3 Font feature meta-details

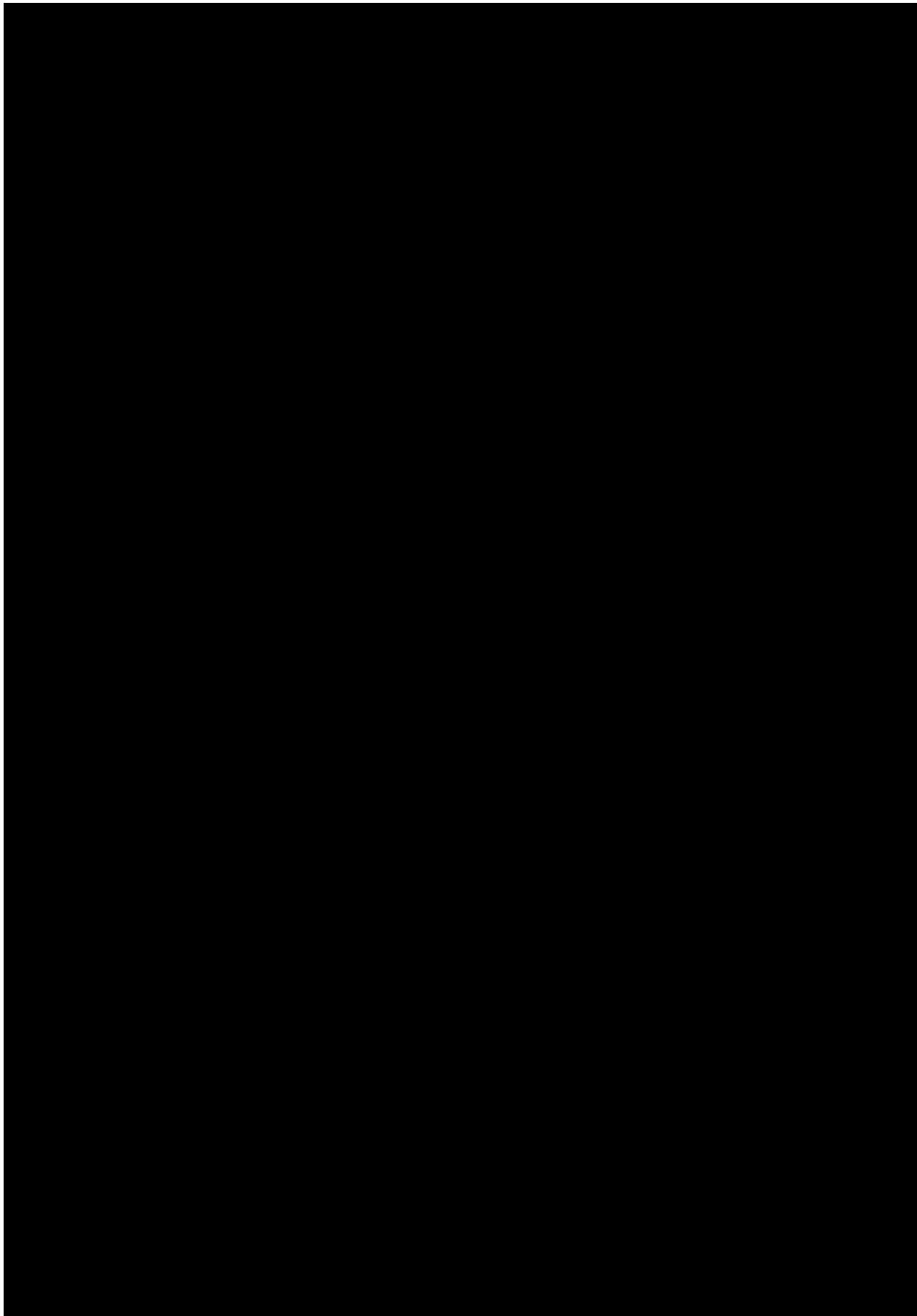
User commands are available to specify font features that `fontspec` doesn’t cater for. These use the plain TeX syntax to define new keys for `\fontspec`. Coincidentally, more than one feature may be activated by a single feature key in this way.

More interestingly, font features and their options may be aliased to alternate name for abbreviation or translation purposes. For example, typing `[VerticalPosition=ScientificInferior]` would be tiresome more than once or twice a document.<sup>8</sup> (Of course, multiple uses of a font instance should be effected with a `\newfontinstance` as previously discussed...). In any case, it may be desirable for whichever reason to rename some features and their options, as shown in example 14.

I haven’t seen this key-aliasing feature offered by other packages<sup>9</sup> although it is simple to implement.

<sup>8</sup> `fontspec` always takes the verbose option for default option names, in the interest in self-documentation.

<sup>9</sup> To be honest, I haven’t really looked.



Indeed, until now all support for feature-rich fonts have required significantly font-specific implementations, which has allowed clever shortcuts like ligatures with  $\text{\^}$  to access alternate glyph forms (also see Toledo [13]). But since the features that fonts offer (especially ‘fancy’ fonts such as Apple Chancery, Poetica and Zapfino) are so varied, it is unrealistic to expect a macro package to be able to fully deal with arbitrarily complex font features in such an abbreviated manner. Provided that the font offers the support, `fontspec` can access any feature the designer wishes to offer without having to mess around with virtual fonts, customised ligatures, and font-specific input-to-glyph mappings, at the expense of the brevity that these schemes allow.

A solution is offered by Toledo: “A better way would be for L<sup>A</sup>T<sub>E</sub>X editors... to allow the user to select glyphs visually and to automatically produce the plain-text markup in the L<sup>A</sup>T<sub>E</sub>X input file.” [13]

## 6 The future of `fontspec`

L<sup>A</sup>T<sub>E</sub>X’s future involves a revamp of the font interface, in order to accommodate access to feature rich fonts through more advanced T<sub>E</sub>X-variants such as X<sub>E</sub>T<sub>E</sub>X. This package simply adds another layer on top of the NFSS, but FRANKly, I am not well-equipped enough (yet...) to implement an NFSS-replacement, either in experience or time. Hopefully, however, `fontspec` grows into something more suitable, or plants the seeds to get there via a different route.

On an internal level, moving the feature processing code from `keyval` to `xkeyval` is planned for the current and next release; this latter package contains many conveniences that will simplify `fontspec`’s feature processing considerably, as well as provide some functionality that would be too convoluted to perform (for me) without it.

### 6.1 Moving away from the NFSS

The NFSS has served the L<sup>A</sup>T<sub>E</sub>X community extraordinarily well, especially when the fact that it is over ten years old is considered. What have been its greatest successes? The first and most obvious is its context-independent commands for shape selection. `\emph` is its crowning achievement in this area. Although many users remain blissfully unaware of this fact, writing logical markup really is next to Godliness.

A greater technical achievement (in my eyes) is the entirely transparent selection of optically-sized fonts. Granted, there haven’t been that many of them. But the fact that `\fontsize{9}{11}` and `\fontsize{12}{14}` selects different shapes of Computer Modern is a testament to the cleverness of the original writers.

---

**Example 16:** Only some of the weights of Helvetica Neue; the only way to access them in the NFSS is with fixed weight codes.

---

```
\newcommand\showfont[1]{\fontspec{\#1}\par}
\showfont{Helvetica Neue UltraLight}
\showfont{Helvetica Neue Light}
\showfont{Helvetica Neue}
\showfont{Helvetica Neue Condensed Bold}
\showfont{Helvetica Neue Condensed Black}
```

---

Helvetica Neue UltraLight  
Helvetica Neue Light  
Helvetica Neue  
**Helvetica Neue Condensed Bold**  
**Helvetica Neue Condensed Black**

---

Where has the NFSS fallen down? While understandable from technical aspects in the day, the small number of font axes makes it quite difficult, or even impossible, to squash every font in some of the larger font families (Lucida specifically comes to mind) into its framework. Something like the `slantsc` package’s support for italic small caps can only go so far (in fact, it hardly goes far enough).

Secondly, the weight/width series of the NFSS cannot be traversed in an easy manner. If a font with book, medium, and bold weights is installed, there is no convenient way to move between them, let alone condensed or expanded versions. In the same way, it is difficult for `fontspec` to provide access to more than the canonical bold/italic/small caps shapes defined by the `\text..` commands. Expect an improvement on this front in a future version of `fontspec` that uses parameterised series identifiers to deal with fonts with many weights, such as Helvetica Neue, which is shown in example 16. In the NFSS, two letter series codes would be required to move between these shapes. A relative boldness command (*cf.* the `relsize` package) would perform the job much more conveniently.

Finally, any *further* promulgation of font commands that look like `\textZZ`, in which ZZ can be any cryptic combination of letters that only made sense when it was conceived, has got to stop. There is only so much meaning that can be obtained from two-letter abbreviations, and trying to get lightface, lining figure, oldstyle figures, superior numbers, and so on into that system implies incomprehensible command names.

## 6.2 Limitations of `fontspec`

The `fontspec` package started out before I even knew how to program in L<sup>A</sup>T<sub>E</sub>X, but on the whole I feel that it has been quite successful. The user interface is still being refined, but it is becoming more stable. Anecdotally, people seem to have found it useful to write (often multilingual) documents in L<sup>A</sup>T<sub>E</sub>X with it. However, it is only the first step along the way to a ‘future font selection scheme’, and there are many improvements that could be made.

The OpenType specifications [2] define not only the corpus of standardised features, but also how interacting font features should behave. At present, features are processed by X<sub>E</sub>T<sub>E</sub>X in the order that the user specifies, but `\addfontfeatures` might override one feature with another—`fontspec` should be able to inform the user what is going on in this case (*i.e.*, which features are being overridden and why).

The second, larger problem is that `fontspec` is X<sub>E</sub>T<sub>E</sub>X-specific. A worthy and involved goal would be to unify font access based on OpenType support between the three main T<sub>E</sub>X variants: pdfT<sub>E</sub>X via the LCDF Typetools [5], which ‘only’ provide the means to generate readable fonts based on OpenType font features, not a higher-level interface or consistent framework for doing so; Ω and its future descendants when they emerge [7, 10]; and X<sub>E</sub>T<sub>E</sub>X as discussed herein. ε<sub>X</sub>T<sub>E</sub>X is another T<sub>E</sub>X-like program that I haven’t even considered yet, but whose organisation and development seems very promising [8]. For now, `fontspec`’s implementation is closely tied to the syntax shown in example 1; abstracting the `\fontspec` command away from X<sub>E</sub>T<sub>E</sub>X is not planned for the short term, however.

The most applicable work in this direction would be for the pdfT<sub>E</sub>X–L<sup>A</sup>T<sub>E</sub>X format. It is a great advantage to be able to refer to fonts by name, and an extra file in the TDS that contains a mapping between logical font names and ‘Karl Berry’-like font filenames could be used to achieve this. Hypothetically, fonts would keep their current obscure abbreviated names for selection in plain pdfT<sub>E</sub>X; for a `fontspec`-like higher-level implementation, this new file (or collection of files) could be `\input` and processed during the T<sub>E</sub>X run to get from, *e.g.*, `[Numbers=Lowercase]{Aldus}` to `pasj`.

## 7 Implementation notes

L<sup>A</sup>T<sub>E</sub>X’s NFSS defines font families with variations in series and shape only. With some extra code, it can be extended to handle small caps independently of italics. Further distinctions in more advanced fonts, such as lining and old-style figures, swash charac-

ters and inferior/superior numbers, can be organised with new sub-families by appending various characters to the original family name. These ideas have already been instantiated by Philipp Lehman, in an experimental package he calls `nfssext` [6].

As an example, the Aldus typeface is given the NFSS family name `pas` in its basic form, and `pasj` is Aldus with lowercase numbers. The `nfssext` package contains macros that switch from `pas` to `pasj`, and other forms such as `pas1` for superior numbers, that work for any collection of font families that adhere to these principles.

The `fontspec` package uses this idea of varying font features based on varying NFSS family names. However, it does not use the idea of manipulating the font family name directly, since combinations of variations end up creating an explosion of family names.

Rather than creating, before-the-fact, a slew of `.fd` files for each desired permutation of font features, then carefully selecting between them when it comes time to actually choose the font, `fontspec` instead defines new families as it goes, based on the features that are actually requested at the time.

When a font is selected, the feature list is processed to create a unique identifier for this particular font instance. This identifier is used to determine if a font has been loaded previously; at the same time, the NFSS family name is created by incrementing a counter appended to the name of the plain font.

For example, the unique identifier created when requesting ‘Hoefler Text’ with lining figures and coloured red could be:

```
Hoefler Text+col:CC0000+21,1
```

where (21, 1) is the AAT feature code for accessing lining figures. This is used in a `\csname... \endcsname` construct to refer to the font family name; *e.g.*, `Hoefler Text (0)`, if this is the first time that Hoefler Text has been selected in any form.

Once the NFSS family name is created, it is used in *another* `\csname... \endcsname` construct to save the information used to create it; *i.e.*, the features requested and the font name, such that they can be retrieved for subsequent use with the `\addfontfeatures` command.

These steps for font selection, and later feature addition, are summarised as follows.

### 1. The commands

```
\defaultfontfeatures{Numbers=OldStyle}
\fontspec[Ligatures=Rare]{Warnock Pro}
```

### 2. are equivalent to

```
\fontspec[Numbers=OldStyle,
          Ligatures=Rare]{Warnock Pro}
```

3. which produces the font instance identifier  

```
Warnock Pro+onum+dlig
```
4. which, in `\csname... \endcsname`, gives  

```
Warnock Pro (0)
```

(the NFSS family name)
5. which, in another `\csname... \endcsname` construct, gives  

```
{Numbers=OldStyle,Ligatures=Rare}
{Warnock Pro}
```
6. This is used by  

```
\addfontfeature{Color=CC0000}
```
7. which is in this case equivalent to  

```
\fontspec[Numbers=OldStyle,
          Ligatures=Rare,
          Color=CC0000]{Warnock Pro}
```
8. and the process starts again. So that's how fonts are selected in `fontspec`!

At present, the same features loaded in a different order constitutes the creation of a separate font family. Overcoming this flaw simply involves sorting the features as they are added to the font instance identifier, which shouldn't be too hard to organise, in theory.

A trickier problem is informing the user which features are available for a newly-selected font. To accomplish this, when a new font is selected, a loop would be used to build up a list of the features and options within the font. This list of (low-level) feature strings must then be converted to `fontspec`-level commands for display in the transcript file on request, which would mean that font feature definition in the source would have to be a two-way mapping from both AAT and OpenType features to `fontspec` features.

## 8 Conclusion

Well, I hope you've enjoyed this brief look into this  $\text{\texttt{X}}\text{\texttt{E}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}-\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  package. I've learned an incredible amount since I started, I've got a lot more to learn, and I'm looking forward to improving the package in the future... but not too soon. Comments, suggestions, and criticisms are all especially welcomed.

## 9 Acknowledgements

The `fontspec` package could not have been possible without the efforts of Jonathan Kew, who is developing  $\text{\texttt{X}}\text{\texttt{E}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  at SIL and whom I thank greatly for his hard work and good advice. My gratitude goes also to those who have kindly answered my (often trivial) questions on various mailing lists and `comp.text.tex`.

## References

- [1] William F. Adams. There is no end: Omega and Zapfino. *TUGboat*, 24(2):183–199, November 2003.
- [2] Microsoft Corporation. *OpenType Layout tag registry*. <http://www.microsoft.com/typography/otspec/featuretags.htm>.
- [3] Alan Hoenig. The Poetica family: fancy fonts with  $\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  and  $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ . *TUGboat*, 16(3):244–252, September 1995.
- [4] Jonathan Kew.  $\text{\texttt{X}}\text{\texttt{E}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ , the Multilingual Lion:  $\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  meets Unicode and smart font technologies. *TUGboat*, 26(2):115–124, 2005.
- [5] Eddie Kohler. *LCDF Typetools*. <http://www.lcdf.org/type/index.html#typetools>.
- [6] Philipp Lehman. *The Font Installation Guide*, 2.0 edition, April 2004. (Documentation file `fontinstallationguide.pdf`).
- [7] Anish Mehta, Gábor Bella, and Yannis Haralambous. Adapting  $\Omega$  to OpenType fonts. *TUGboat*, 24(3):550–556, 2003.
- [8] Gerd Neugebauer et al. *The  $\varepsilon\text{\texttt{X}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  project*. <http://www.extex.org/>.
- [9] Heiko Oberdiek. *The pdfcolmk package*. <http://www.ctan.org/tex-archive/macros/latex/contrib/oberdiek/pdfcolmk.sty> (Documentation within .sty file).
- [10] John Plaice, Yannis Haralambous, Paul Swoboda, and Gábor Bella. Moving  $\Omega$  to an object-oriented platform. In *Lecture Notes in Computer Science*, volume 3130, pages 17–26, 2004.
- [11] Walter Schmidt. *Using common PostScript fonts with  $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$* . <http://www.ctan.org/tex-archive/macros/latex/required/psnfss/psnfss2e.pdf>.
- [12]  $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}3$  Project Team.  *$\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}\ 2\varepsilon$  font selection*. <http://www.latex-project.org/guides/fntguide.pdf>.
- [13] Sivan Toledo. Exploiting rich fonts. *TUGboat*, 21(2):121–129, June 2000.

◇ Will Robertson  
 School of Mechanical Engineering  
 University of Adelaide, SA  
 Australia 5005  
 wspr81 (at) gmail.com

## **\BzLATEX:** A package for typesetting “Byzantine” music

Ioannis A. Vamvakas and Panagiotis Kotpoulis

### Abstract

Using Donald Knuth’s METAFONT, almost all the symbols of “Byzantine” music have been designed and organized in three font series: byzf, byyf and bzal. Musical phrases from the Hellenic ecclesiastical and folk music traditions are typeset using the ad hoc package **byzlatex** as examples.

### 1 Introduction

“Byzantine” music is the official music used by the Greek Orthodox Church for her liturgical needs; all other Christian Orthodox churches make use of the Western or European scales and notation. At the same time, “Byzantine” is the music used in folk music tradition in different areas throughout Cyprus and Greece. Efforts by admirers of the Western tradition at the beginning of the previous century seemed to succeed but eventually encountered the strong opposition of the people. In our time there has been a revival of the pure tradition, from the music and dances to the architecture and food recipes. Hundreds of schools specializing in the teaching of folk music (dancing, singing, chanting or playing instruments) have been founded by the Church, local city authorities, and private conservatories.

“Byzantine” music, along with its scales, notes and the rest of its symbols, is very little known in the West. This is understandable and expected since it belongs only to the tradition of a small nation among the thousands of local traditions. Be that as it may, some efforts have been made to design fonts containing the special symbols used in the musical phrases. Dr. Velissarios G. Gezerlis, Ph.D., has designed a program called *ByzWriter 2.0* [2] with which one is able to typeset “Byzantine” musical phrases in the Windows environment. Yet, no package has appeared in the (L)ATEX community to allow a Byzantine musician/TExnician to typeset his own phrases using his favorite editor and LATEX! Through the work presented in this paper, we hope to begin to remedy the lack of “Byzantine” music fonts and LATEX styles.

The authors of this paper belong to the student-teacher category, with the first being the student and the second the teacher. We both belong to that group of people who are in the middle of this comeback of tradition, Panagiotis being Lampadarios (official title for a cantor in the Orthodox Church) at St. George’s church in Rion, Patras, while Ioannis is

a member of “The Aegean Akritae”, a folk dancing group, and is also studying “Byzantine” music.

At the beginning of my (Ioannis’) studies, I hoped to have all the tones and marks used by “Byzantine” Music, typeset and printed in banner size. I thought that this would be a nice way to look at them and eventually learn them by heart. I set out looking among the CTAN archives for “Byzantine” letters and music tones but they were nowhere to be found. My initial disappointment turned into hope when I bought Donald Knuth’s famous book [5] on how to design fonts. Since that day I have not stopped designing and redesigning the characters of the three fonts presented in this paper. I can testify that Donald Knuth’s warning<sup>1</sup> comes true!

### 2 A short history of “Byzantine” music

In April, 311 AD, the Emperor of the Western part of the Roman Empire, Constantine the Great, issued an edict that put an end to the Great Persecution of the Christians. Two years later, on June 13, he and the Emperor of the Eastern Roman Empire, Licinius, promulgated at Mediolanum (today’s Milan, Italy) an edict that allowed all citizens of the Empire to worship whatever Deity they wanted. That day marked the birth of Christianity as an accepted religion. Ten years later Christianity became the official religion of the Roman Empire. On May 11, AD 330, Constantine, as the only ruler of the state, selected Byzantium, a small town that had been founded in 657 BC by Byzas, a Greek from Megara, as the new capital of his Empire. Finally, on Christmas Day, AD 800, when Charles, son of Pepin the Frank, was crowned by Pope Leo III as Emperor of Rome, it marked the political separation of the Western from the Eastern Roman Empire, which, by then, had become Greek in language and habits. With the Great Schism of 1054, the spiritual separation took place and the permanent estrangement of the two peoples was completed.

This brief historical overview of the Eastern Roman Empire that became known as “Byzantine”,<sup>2</sup> is important for us to realize that citizens in the East

<sup>1</sup> DEK: Type design can be hazardous to your interests. Once you get hooked, you will develop intense feelings about letterforms; the medium will intrude on the messages that you read. And you will perpetually be thinking of improvements to the fonts that you see everywhere, especially those of your own design.

<sup>2</sup> According to the Wikipedia Internet Encyclopedia (<http://en.wikipedia.org>):

The name “Byzantine Empire” is a modern term and would have appeared alien to its contemporaries. The term was invented in 1557, about a century after the fall of Constantinople (May 29, 1453) by German historian Hieronymus Wolf (ca. 1516–ca. 1580),

had a perception of life that was completely foreign to those in the West. As far as music is concerned, according to Christodoulos Halaris [4], a composer and researcher in ancient and medieval music:

... The ancient Greeks legated to the Byzantine both their musical philosophy and analytical music writing system they had invented. Thanks to the writings of Alupius, but also to Meibom who published in the 17th century the compiled works of ancient harmony writers, this system is to a great extent, readable in our times.

In the 4th century AD the ancient Greek music scripture system, whose symbols were derived from the ancient Greek and Ionian alphabets, is abandoned. A new system is born, which is also named “Parasemantics” but which, in contrast to the previous one, is of a clearly algorithmic nature. Its symbols do not denote tonal style, duration etc. but describe the behavior of the performer and, through it, the musical event to follow. As regards graphics, this system is inspired from the “pneumata” that define the pronunciation of words in the written form of the Greek language in Hellenistic times ...

This new system born in the 4th century AD may have had its symbols changed compared to the ancient ones, yet, the most important thing to remember is that there is an unaltered continuation of Greek music, both as a concept and more practically as scales and intervals, from ancient times,

---

who introduced a system of Byzantine historiography in his work *Corpus Historiæ Byzantinæ* in order to distinguish ancient Roman from medieval Greek history without drawing attention to their ancient predecessors. Standardization of the term did not occur until the 17th century when French authors such as Montesquieu began to popularize it. Hieronymus himself was influenced by the rift caused by the 9th century dispute between Romans (Byzantines as we render them today) and Franks, who, under Charlemagne's newly formed empire, and in concert with the Pope, attempted to legitimize their conquests by claiming inheritance of Roman rights in Italy, thereby renouncing their eastern neighbors as true Romans. The Donation of Constantine, one of the most famous forged documents in history, played a crucial rôle in this. Henceforth, it was fixed policy in the West to refer to the emperor in Constantinople not by the usual *Imperator Romanorum* (Emperor of the Romans) which was now reserved for the Frankish monarch, but as *Imperator Graecorum* (Emperor of the Greeks) and the land as *Imperium Graecorum, Graecia, Terra Graecorum* or even *Imperium Constantinopolitanus*.

For this reason the authors feel the need to restore truth by enclosing the term within “” throughout this article.

the Hellenistic period, on to the early Roman era, to “Byzantine” times, during the years of the Ottoman Empire, to this day.

Until the beginning of the 19th century, many laymen and church people tried to collect all the information on music notation and classify it in such a manner that would make it acceptable and easy to learn by all. Here we will not discuss details of the evolution of the “Byzantine” notation throughout the centuries; the interested reader will be able to find a detailed description of it on the web page of St. Anthony’s Greek Orthodox Monastery [6], Florence, AZ.

We do want to mention the great reformation that took place in 1814. The basic reformer was Metropolitan of Prussa, Chrysanthos of Madytos (ca. 1770–ca. 1840), who, along with the protopsaltes (chief cantor) Gregorios and the archivist Chourmouzios, made up the so-called “Three Teachers”. Basing their method on the Western sol-fa system, they invented the seven monosyllabic sounds according to the first seven letters of the Greek alphabet, facilitated the complex medieval neumatic notation and simplified the teaching of this art. Overall, what they accomplished was to shorten the teaching-learning process from ten years to ten months!

Having the blessing of the Ecumenical Patriarchate of Constantinople (today's Istanbul, Turkey), they established their own school of music in 1815, thus making certified teachers to propagate this new method. Eventually, Chrysanthos wrote a book titled *Introduction to the theory and practice of ecclesiastical music written for the use of those studying according to the new method*, where he described the new system of teaching; the book was published in Paris in 1821. In 1832 he published the book *Great Theory of Music*, where a more detailed presentation and explanation of the new method is given.

### 3 A short introduction to the theory of “Byzantine” music

“Byzantine” music (BM) is the official ecclesiastical music of the Greek Orthodox Church, as well as being part of the folk traditions in Greece, Cyprus and anywhere else Greeks live. Other Eastern Orthodox Christians use Western notation and system for their liturgical needs. Thus, these melodies are more familiar to Western ears; the Hellenic (Greek<sup>3</sup>) musical notation system is a *Great Unknown*. Over the past century a host of people have tried to make BM more accessible to the West by transcribing “By-

---

<sup>3</sup> The term “hellen” refers to Greeks who may not be citizens of the Greek state.

zantine” melodies into Western staff notation, yet, the result to an Eastern Orthodox has been acoustically very poor. This is because there are differences between the two musical systems that render them quite foreign to each other.

Next, we briefly discuss the most important features that differentiate the two systems. We hope that this brief discussion will provide a clue to the logic behind the structural method we followed for the design of the font series presented later.

First, as a general observation, we should make a reference to the opposing ways these two civilizations (Western and Hellenic) conceive the world. There is a deep gap between them, dating from the 2nd century BC, when the newly born Roman and the ancient Greek worlds met. Within the boundaries of the Roman Empire the clash over *Knowledge of Truth* broke out. The Greek understands Truth in *relation* with his fellow-men, he “... refuses to exhaust knowledge of the truth in its formulation...” [7], which is a fundamental concept in the Roman world. A Western mind does not like the idea of staying in the *description* of knowledge thus allowing for a free interpretation of it. It feels secure when it has an *absolute determination* of a concept which leaves no room for any other understanding. According to Prof. Christos Yannaras, there is a:

... refusal to exhaust knowledge of the truth in its formulation. The formulation is necessary and required, because it *defines* the truth, it separates and distinguishes it from every distortion and falsification of it... At the same time though, this formulation neither replaces nor exhausts the knowledge of the truth, which remains experiential and practical, a way of life and not a theoretical construction...

On the other hand, he claims that for the Western conception:

... The conventional logic of everyday understanding can very easily give man a false sense of a sure knowledge which, being won by the intellect, is already exhausted by it, completely possessed by it...

Just as the laws of justice fix the boundaries of the objective and effective assurance of social harmony, so also the definite, inescapably schematic—but commonly received—defining of truth assures the effective objectivity of knowledge and constitutes a kind of law of truth.

And so, for the first time in history, truth is identified with its formulation and knowledge

or the possession of truth with the individual understanding of this formulation. The truth is separated from the dynamic of life, it is identified with the concept, with right reasoning...

This differentiation in the perception of truth runs through all aspects of human life, from the legal system, governing of state, to the approaching of God and, of course, music.

After this general introduction, we can now see the most important differences between the two systems:

1. BM is strictly *monophonic*. *Polyphony* and *harmony*, which are the basis of Western music (WM), have no place in the Hellenic tradition.
2. Western notation is *absolute* and *determinative*, whereas “Byzantine” notation is *relative* and *descriptive*. To quote Prof. Demetrios Giannelos [3]:

A descriptive notation, such as that of Byzantine music, describes the essentials of the piece, leaving to oral tradition the task of completing with precision whatever is not described. On the contrary, a determinative form of writing, such as Western notation with staves, determines with great precision the manner of execution, to the point that the interpretation of the person executing it is delineated by factors that depend directly on the definitive indications of the music symbols. These indications can be absolutely restricting in that they preclude all room for interpretation.

In practice, the ...

... Western notation describes the melody in terms of absolute pitches whereas “Byzantine” notation describes the melody as relative pitches within a particular predefined scale... [6]

3. In a “Byzantine” melody the music has only one goal, to serve and emphasize the Word. This is not a characteristic only of the “Byzantine” tradition but it has spread over 5000 years of Greek history. The Word comes first and the music follows to stress the former, never the opposite; this rule is meticulously obeyed even in modern Greek popular music. To return to the “Byzantine” notation, every musical phrase is made up of two parallel lines: an upper line containing the quantity and the quality symbols and a lower line containing the syllables of the

C	D	E	F	G	A	B	C
200	167	133	200	133	233	133	
200	200	100	200	200	200	100	

**Figure 1:** Comparison between a typical Western (lower) and “Byzantine” (upper) scale

particular hymn, each syllable corresponding to one or more symbols. Thus, a BM font designer will have to take into consideration the distance between neighboring lines when deciding the dimensions of the bounding box that enclose an individual character, most especially its depth and height.

4. To a Western ear a Greek melodic line seems foreign. This is because the intervals between the notes (the frequency that a note is higher or lower than its neighbor) in the two scales are different. To understand this let us introduce the two scales. The typical seven pitches of the Western scale are:

do, re, mi, fa, sol, la, ti

The seven pitches of a “Byzantine” scale have names that originate from the first seven letters of the Greek alphabet ( $\text{Α}, \text{Β}, \text{Γ}, \text{Δ}, \text{Ε}, \text{Ζ}, \text{Η}$ ) with a consonant prepended or a vowel appended to produce a sounded syllable; in other words:

$\pi\text{Α} \quad \text{Βou} \quad \Gamma\alpha \quad \Delta\iota \quad \chi\text{E} \quad \text{Ζω} \quad \nu\text{H}$   
(paa voo gaa thee ke zo nee)

The two sets coincide when:

D	re	$\leftrightarrow$	$\pi\text{Α}$
E	mi	$\leftrightarrow$	Bou
F	fa	$\leftrightarrow$	$\Gamma\alpha$
G	sol	$\leftrightarrow$	$\Delta\iota$
A	la	$\leftrightarrow$	$\chi\text{E}$
B	ti	$\leftrightarrow$	$\text{Ζω}$
C	Do	$\leftrightarrow$	$\nu\text{H}$

Figure 1 compares the typical Western C major diatonic with the “Byzantine” soft chromatic scale. The distance between the two ends of the scale (from do to do’ or from C to the next C) is equal to 1200 cents. By inspection, one can see that the interval between D and E is 33 cents shorter and between E and F is 33 cents longer. The same interval difference exists between G and A, A and B and B and C. Any Western scale consists of whole and half intervals (200 cents and 100 cents, respectively); no other intervals are allowed. One can convert a

half interval into a whole and vice versa by making the end notes either sharp or flat. On the other hand, the typical “Byzantine” scale, besides having intervals other than 200 cents and 100 cents, as we see in fig. 1, one may sharpen or flatten a note to a non-Western pitch. Western notation is not adequate to cover all these individual cases.

5. Both systems possess many qualitative marks that describe how a note or a group of notes should be chanted. The difference enters not through the symbols used by each system but through their semantics. Each system has its own definition of “quality” and this is mirrored in the symbols used. Since BM is strictly vocal, as we discussed before, the main goal of a potential composer is to stress the word. Traditionally, in the ancient Orthodox Church, the performance of a melodic phrase is accompanied by heavy gesturing (“neume” in Greek) by the cantor or the chorist (something that, unfortunately, has been abandoned by modern-day cantors). The quality marks, the neumes, would have to resemble the hand motion.

To give an example, **petastē** is a neume that bears both quantity and quality properties; it tells the cantor that upon ascending the scale between two neighboring pitches he will have to abruptly raise his voice. The shape of petastē resembles the trace of the fingertip as it moves from the lower up to the higher note with the abrupt voice raising taking place when closer to the latter.

Another example is the **psifistōn**. Its symmetrical shape shows the cantor that he will have to stress the syllable upon which the psifiston acts, as soon as he is on it and not before or after.

As mentioned previously, a scale is a series of seven pitches, either ascending or descending, with the two edges having distance equal to 1200 cents or the highest note having twice the frequency of the lowest. To make a scale all we need is to start with

Character	Name	Interval	Character	Name	Interval
—	Ison	0	↗	Ypsēlē	+4
—	Oligon	+1	↘	Apóstrophos	-1
—	Petastē	+1	↙	Elaphrón	-2
—	Kentēmata	+1	↓	Yporroē	-1 - 1
—	Kéntēma	+2	↖	Hamelē	-4

Table 1: The 10 quantity symbols of the “Byzantine” notation

a note, say Re (D), and ascend or descend till we reach the next higher or previous lower, Re' or re, respectively.

Two kinds of scales are standard in Western music, majors and minors. The former have seven consecutive intervals of the form WWHWWWW (where “W” stands for a whole interval or 200 cents and H stands for a half interval or 100 cents), while the seven intervals of the latter have the form WH-WHWW.

In “Byzantine” music we can also make scales according to the Western analogy, but here we must introduce the idea of *mode*. The backbone of a mode is still a scale but the intervals do not remain the same ascending and descending. The entire melodic line seems to revolve around a note, and this note, called the “drone”, creates the foundation upon which the melody is built. Ancient Greek music had 15 such modes; ecclesiastical music kept only eight of those, deemed less secular and more suitable to the Church piety. The eight are split into four pairs, the regular and its “plagal” mode, the latter having a scale that starts four intervals higher or lower than the former. In the next section we further discuss the rôle of the modes in the “Byzantine” melody.

#### 4 Constituents of “Byzantine” notation

The basic notation used in “Byzantine” music uses ten marks, called *quantity* neumes. These characters with their names can be seen in table 1.

Also seen in the table are the intervals that the pitch should ascend (+) or descend (-) relative to the previous quantity character. The ‘Ison’ (0) character tells the cantor that he should move on to the next character without ascending or descending his pitch. To exemplify, Oligon asks for a 1-interval ascending, which means that if the previous character was on C the next to be pronounced should be D.

On the other hand, Yporroē asks for two consecutive descendings, e.g., if the previous character was on E, we should clearly pronounce D and C. Combinations of the 10 basic quantity symbols allow for ascending or descending a number of intervals other than the ones shown in table (1). For example, placing a Kéntēma over Oligon, as in:



represents three intervals ascending, whereas placing Apóstrophos under Hamelē, as in;



allows for six intervals descending.

Four pieces of information are needed before starting the chanting of a piece:

- [a] Based on the relativism of “Byzantine” music, each character does not exist on its own but in connection with its previous neighbor (unlike Western notation, where a note is perfectly known from its position on the staff). The name, that is, C, D, etc., of the very first character is known with the help of a clef symbol known as *Martyrēa*. The word “martyrō” in Greek means to bear witness. In other words, *Martyrēa* tells the cantor what is the very first character of the melodic piece.
- [b] The cantor needs to know how he should chant the particular character he is on: whether he should stress or lower his voice, do a short trembling, etc. The symbols bearing this kind of information are called *quality* marks and there are six of them, shown in table 2.
- [c] The cantor needs to know how long he should spend on each syllable. The time characters, shown in table 3, provide this kind of information, given a unit of time or tempo. Table 4 shows the six possible tempos along with their

Character	Definition	Name	Quality Action
˘	\ank	Antikénoma	Short jerking of the voice
ˉ	\ank	Vareā	Chanting in a base manner
˙	\ank	Endófonon	Humming
˘	\ank	Eteron	Connecting same quantity characters
˘	\oml	Omalón	Waving the voice in a trembling manner
˘	\oml	Psēphistón	Stressing the specific vowel

Table 2: The 6 quality characters of the “Byzantine” notation

Character	Definition	Name	Action
˘	\kla	Klásma	2× the unit of time
.		Aplē	2× the unit of time
..		Diplē	3× the unit of time
...		Triple	4× the unit of time
˘	\grg	Gorgón	1/2× the unit of time
˘	\dgrg	Dēgorgón	1/3× the unit of time
˘	\tgrg	Trēgorgón	1/4× the unit of time
˘	\arg	Argón	3/2× the unit of time
˘	\darg	Dēargon	5/2× the unit of time
˘	\targ	Trēargon	7/2× the unit of time
+	\stav	Stavrós	Pause to breathe
˘	\paf	Páfsis	Pause for the unit of time
˘	\kor	Koronēda	Pronouncing for unlimited time

Table 3: The 12 time characters of the “Byzantine” notation

Western counterparts, the calligraphic  $\chi$  being the first letter of the word “time” in Greek ( $\chiρόνος$ ). If there are no time marks then the normal tempo is assumed.

- [d] Finally, the cantor needs to know the *mode* according to which he chants the specific hymn. As mentioned earlier, there are eight modes appropriate for the liturgical needs of the Orthodox Church. Table 5 shows symbols for all eight of them, along with their names and their *martyrees*.

Also as mentioned previously, “Byzantine” music is strictly vocal and monophonic. The music notation always goes hand-in-hand with the words of the hymn. Nowhere will one find music notation on its own. The exercises made especially for “Byzantine” music students are written according to the sol-fa

system, whereas, special pieces, called “kratēmata”, written for practicing reasons, use meaningless syllables like **te**, **ta**, **rē**, **rem**, that help the cantor demonstrate his vocal skills.<sup>4</sup>

Every musical phrase consists of two lines, one above the other. The first line contains the musical symbols whereas the second line contains the syllables of the hymn to be chanted; each syllable corresponding to one or more symbols should be placed exactly under it, so there is no room for doubt as to the correspondence between symbols and syllables. As far as typesetting is concerned, this is a typical alignment problem which can be solved in L<sup>A</sup>T<sub>E</sub>X

---

<sup>4</sup> Given what we have seen so far, it will be understandable that this kind of showing off is not part of the liturgical music. Chanters do “kratēmata” only outside the church and in secular “Byzantine” concerts.

Tempo Symbol	Beats/minute	Action	Western symbol
	130	Normal	Andante
	168–208	Fast	Allegro
	> 208	Very fast	Presto or Prestissimo
	100–168	Moderate	Moderato
	80–100	Slow	Lento or Adagio
	56–80	Very slow	Grave or Largo

Table 4: The 6 tempo characters of the “Byzantine” notation

Martyrēa	Mode Name	Martyrēa	Mode Name
	First	$\pi$	First Plagal
	Second	$\lambda$	Second Plagal
	Third		Grave
	Fourth	$\lambda$	Fourth Plagal

Table 5: The 8 modes of the “Byzantine” music

with the *tabular* environment. We turn now to such typesetting issues.

## 5 Typesetting

The user starts by opening a L<sup>A</sup>T<sub>E</sub>X document along the lines of the following sample:

```
\documentclass[12pt]{article}

\usepackage[greek,english]{babel}
\usepackage{longtable}
\usepackage{fullpage}
\usepackage{byzfonts}

\begin{document}
```

The standard *babel* package for multi-langular support, with the *greek* and the *english* options, is necessary since the characters were designed to be used in a purely Greek environment; the \wg macro can be used when Greek environment is mingled with English. Next, a 16-column tabular environment with no vertical or horizontal rules is required for the basic layout. David Carlisle’s *longtable* pack-

age is used to typeset tables that cover more than one page, which is exactly what we would like to do here. We have found that, due to the larger size of the music characters compared to the letters under them and for readability purposes, 16 characters per line in the *fullsize* page environment is the optimum selection.

For the body of the document, the user types the 16 musical characters and, on the following line, he types the 16 syllables. Sometimes he types a *pause* mark, a *diastole* (a vertical line that separates different time rhythms), an empty space or something other than a syllable.

The appendix shows the L<sup>A</sup>T<sub>E</sub>X code for an example psalm using the ByZL<sup>A</sup>T<sub>E</sub>X package, along with the corresponding output. We describe some of the package details next.

While struggling with the onomatopoeia for the macros of the individual glyphs, we decided to keep the traditional names for the ten basic quantity characters as well as for a few other simple neumes, for instance, \iso and \apo. As for the rest, we fol-

lowed a simple mnemonic rule: every character possesses a particular place in the font tables, given by its numerical position (from 0 to 255 in the decimal system, 000 to 377 in octal, or 00 to FF in hexadecimal). Every glyph is given a 3-letter macro name corresponding to its position in the font table as an octal number, according to this simple mapping:

$$\begin{array}{llll} 0 \leftrightarrow o & 1 \leftrightarrow a & 2 \leftrightarrow b & 3 \leftrightarrow c \\ 4 \leftrightarrow d & 5 \leftrightarrow e & 6 \leftrightarrow f & 7 \leftrightarrow g \end{array}$$

Thus, the glyph found at position 056 octal is represented by `\oef`.

Since the number of character combinations is greater than 256, we needed two font sets, which we named *bzf* and *byf*. To access glyphs from either font, we prepend each character's macro name with `\z` for characters taken from *bzf* or `\y` for characters belonging to *byf*, to select the given font.

We also designed a third font, called *blal*, that contains the following. [a] A series of capital calligraphic letters with height equal to the distance between neighboring baselines, so that they can be used in the place of the first letter of the first word of a melodic phrase. [b] A series of small calligraphic letters, similar to those found in original “Byzantine” codes at the Athonite Monasteries. [c] A series of ligatures.

To access these characters without conflicting with the macro names of the two previous font series, we followed a different approach. Each character is given a 3-letter macro name, as before, but now the previous mapping is replaced by:

$$\begin{array}{lll} 0 \leftrightarrow (z)ero & 1 \leftrightarrow (o)ne & 2 \leftrightarrow (t)wo \\ 3 \leftrightarrow th(r)ee & 4 \leftrightarrow (f)our & 5 \leftrightarrow f(i)ve \\ 6 \leftrightarrow si(x) & 7 \leftrightarrow se(v)en \end{array}$$

Thus, the character at octal 105 will be accessed by `\ozi`. Prepending the `\f` macro to indicate the *blal* font, glyph oct105 is represented by `\f\ozi`. Of course the calligraphy letters are accessed by just putting `\f` in front of the letter, as in `\f B`.

Finally, to include a “Byzantine” font character in a sentence, of course the font switch must be enclosed in a group, as in `{\f\ozi}`.

## 6 Conclusion

Using Donald Knuth's METAFONT we have designed *almost* all the glyphs (quantity and quality neumes, tempo characters, sharp and flat marks, as well as a series of letters and ligatures) used in the “Byzantine” music (the music of the Greek Orthodox Church and of Greek folk songs and dances). We also constructed the **BZLATEX** package employing the LATEX's *tabular* environment for typesetting nice music phrases.

One shortcoming of the package is the lack of a method for users to make character combinations from existing simple ones. This is the reason we emphasized the word “almost” above. We meant the combinations that came to our knowledge by the time of this article. For complex characters not found in the tables the user will either have to design them from scratch (using METAFONT) or contact the authors.

## 7 Acknowledgments

The authors would like to thank Dr. Apostolos Syropoulos, President of the Greek TEX Friends user group [1] and Dr. Dimitrios Filippou, editor of *Eutopon* (the Greek publication for TEX/LATEX), from Demokritos University, Xanthi, Greece, as well as Dr. Ioannis Dimakos from Patras University, Patras, Greece, for their technical support. Without their expertise this paper would have not been seen the light of publication. We would also like to thank Maria Malliaris for her useful suggestions and language corrections.

## References

- [1] Greek TEX Friends.  
<http://obelix.ee.duth.gr/eft>.
- [2] Velissarios Gezerlis. Pattern recognition: Two new applications for the typesetting and the optical recognition of the Byzantine music notation. *Eutopon*, (7), October 2001.
- [3] Demetrios Giannelos. Theory and praxis of the chanting art (in Greek). In *1st Panhellenic Conference on the Chanting Art*, page 173.
- [4] Christodoulos Halaris. Byzantine secular classical music.
- [5] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, 1986.
- [6] St. Anthony's Monastery.  
<http://stanthonysmonastery.org>.
- [7] Christos Yannaras. *Elements of Faith*. T & T Clark, 1991.

◊ Ioannis A. Vamvakas  
Naxou 10, Rion  
GR-265 00, Patras  
Greece  
[giavvns \(at\) pythagoras.physics.upatras.gr](mailto:giavvns(at)pythagoras.physics.upatras.gr)  
<http://ctan.org/tex-archive/fonts/byzfonts>

◊ Panagiotis Kotopoulos  
Patras, Greece

## Appendix

L<sup>A</sup>T<sub>E</sub>X code and output for Psalm 102: Excerpts in the Plagal Fourth mode. Adapted from the traditional Athonite melody as written by Hieromonk Hierotheos of Philotheou Monastery and translated from Greek by the monks of St. Anthony's Monastery, Florence, AZ.

```
\setlongtables
\begin{longtable}{*{16}{c}}%{\extracolsep{\fill}c}
&&& \multicolumn{10}{c}{Plagal 4th mode \wgl{\pDHXOC}} && \TAX{200}\\
&&& \multicolumn{10}{c}{\f\OBO}\\
&&& \multicolumn{16}{c}{\f\OBO}\\
&&& \multicolumn{16}{c}{& \y\zxo & \z\iso & \z\cag & \z\apo & \z\oli & \z\zzx & \y\zvi & \\
&&& \z\iso & \z\apo & \z\oli & \z\bgf & \z\apo & \z\cde & \f\OBO & \z\agb \\ 
&&& \{f B\} & less & the & Lord & O & my & soul & bless & \\
&&& - & ed & art & Thou & O & Lord & & Bless \\ 
&&& \z\iso & \z\apo & \z\knt & \z\bgf & \z\apo & \z\iso & \y\zxi & \\
&&& \z\apo & \z\apo & \z\dst & \z\oli & \z\oli & \z\af & \z\aoa & \z\adb \\ 
&&& the & Lord & - & O & my & soul & and & all & \\
&&& - & that & & is & & with & in & me & & bless \\ 
&&& \z\apo & \z\apo & \z\apo & \z\cdg & \z\paf &&&&&&&&& \f\OBO \\ 
&&& His & Ho & ly & Name &&&&&&&&&& \\
&&& & \z\agb & \z\iso & \z\apo & \z\knt & \z\bgf & \z\apo & \z\cde & \\
&&& \f\OBA & \z\apo & \z\iso & \z\oli & \z\knt & \y\zxi & \z\bgf & \z\apo \\ 
&&& \{f B\} & less & the & Lord & - & O & my & soul & & \\
&&& & and & for & get & - & not & all & that & \\
&&& \z\apo & \z\apo & \z\af & \z\ela & \z\dst & \z\acc & \z\paf & & \\
&&& & & & & & & & \f\OBO \\ 
&&& He & hath & done & for & & thee &&&&&&&&& \\
&&& \multicolumn{16}{c}{\f\OBO}\\
\end{longtable}
```

## Software & Tools

### i-Installer: The evolution of a $\text{\TeX}$ install on Mac OS X

Gerben Wierda

#### Abstract

This article reviews the past, present, and future of the i-Installer program on Mac OS X developed by the author, and its related  $\text{\TeX}$  (re)distribution.

#### 1 Apple, NeXT and Mac OS X

Let's start with some history of Apple, as a foundation for the discussion of the i-Installer program and its related  $\text{\TeX}$  redistribution on Apple systems.

Nowadays, perhaps a few percent of PC users use Apple Macintosh systems. This may not sound like much, but subtract the enormous number of tightly regulated office desktops in large corporations and other institutions and we are talking about a sizeable chunk of the desktop population. These days, Apple's market share is again rising drastically. Apple, in fact, has seen a turnaround not often witnessed in the computer world. From 'beleaguered', the company has become a successful forefront of innovation.

Most people these days know Apple for its iPod music player and the revolution it has ignited in the music world. But Apple is still above all a computer maker, which a few years ago was in desperate need of a new operating system, having failed at two attempts at producing a modern operating system itself.

To solve this problem, Apple acquired NeXT, the company that Apple founder Steve Jobs started<sup>1</sup> when he left the company after a row about future developments. When Jobs started NeXT, the idea was to build the best desktop possible. As Jobs put it, NeXT would be either the last big success or the first big failure on the desktop. Technologically it was a big success, but it failed in the market. In fact, Jobs was too late. A de facto standard for word processing had arrived (Microsoft Office) which was strengthened by the advent of increased communication between computer systems, ironically one of NeXTSTEP's key strengths. Microsoft, which saw NeXTSTEP as a possible competitor for OS/2 and later Windows NT, was not interested in strength-

ening the platform with a version of its office suite.<sup>2</sup>

When Jobs started NeXT, he wanted to rectify some mistakes he had made with the Macintosh while at Apple. As Jobs puts it, when Apple visited Xerox PARC, they had witnessed three revolutions, but they were so mesmerized by one (the graphical user interface) that they completely overlooked the other two (networking and object-orientation). For the robust foundation of the new system he chose the BSD flavour of Unix, but based on the latest of kernel technology: Mach from Carnegie-Mellon University.<sup>3</sup>

When Apple acquired NeXT, it acquired not only a very modern operating system,<sup>4</sup> it also acquired the human talent it desperately needed to turn the company around. Apart from Steve Jobs himself, that included Avie Tevanian for software and (though NeXT had stopped making hardware a few years earlier) Jon Rubinstein for hardware. Within a relatively short time, key positions at Apple had been filled by former NeXT employees, thus prompting the often heard comment that NeXT had taken over Apple from the inside. The years since have seen the dramatic shift in Apple's fortunes, and all that during the IT-bust that followed the dot com boom. As Jobs announced when he became CEO: he wanted to innovate Apple out of the downturn and he actually succeeded in doing so. Of course, the fact that Microsoft has not abandoned Apple has helped considerably.

In many ways, therefore, Mac OS X is a much changed and enhanced version of NeXTSTEP, which in its time was a very smooth  $\text{\TeX}$  environment. The foundation remains a Mach kernel and a BSD layer (both open sourced by Apple), on top of which lives a graphics layer based on PDF.<sup>5</sup> This combination of Unix on the one hand and PDF on the other makes it a very welcome environment for  $\text{\TeX}$ , especially since the emergence of pdf $\text{\TeX}$ . Having pdf $\text{\TeX}$ , everything for creating, displaying and printing  $\text{\TeX}$  is available. A Unix-level editor or the standard GUI editor (TextEdit) can be used to write the  $\text{\TeX}$  source. The Unix command line can be used to create the PDF output which can be viewed by the

<sup>2</sup> There was a version of Wordperfect, but though it was the best version of the program available, Wordperfect Corp. had to abandon it because it came under pressure from the emergence of the Microsoft Office monopoly. A very innovative spreadsheet program, Lotus Improv, met the same fate.

<sup>3</sup> Its main designer, Avie Tevanian, chose NeXT above Microsoft for his career and is these days Apple's Vice President in charge of software development.

<sup>4</sup> In principle and in its core, many interfaces, like the BSD Unix interface, had not been kept up to date.

<sup>5</sup> NeXTSTEP was based on Display PostScript.

<sup>1</sup> Besides starting NeXT, Jobs acquired Pixar from George Lucas; Pixar became a huge success in digital movie production.

default PDF previewer (Preview). It is not comfortable, but it is enough.

The step from there to an integrated front end is simple on Mac OS X. With the object oriented frameworks and a nice programming IDE at the disposal of a programmer, a GUI application with an edit window (based on the available text object) and a display window (based on the available view object, which handles PDF) and calling Unix  $\text{\TeX}$  to do the work behind the scenes, a front end is much easier to write than on any other platform. After all, the whole rendering of the result is already taken care of by the operating system. On such a basis the first  $\text{\TeX}$ -IDE (TeXShop) was developed and the developers could spend time on improving the user interface without having to worry about basic technical issues.

### 1.1 Installing software on Mac OS X

Basically, there are two ways of installing software on Mac OS X:

- Drag and drop
- Installer program

Just dragging something to a folder is of course the simplest metaphor for installing software and is preferred in most situations. Even situations that require configuration and sub-installation in protected locations can use this method if the application itself detects that it has not been installed yet when it is run for the first time. Microsoft Office can be installed this way on Mac OS X.

Using an installer is better for more complicated installs, especially if the location of the install is fixed. After all, installing something at the Unix level often requires something to be installed in a special location in the Unix domain. For these kinds of installs, Mac OS X comes with a program named Installer, that installs “packages” (with the .pkg extension). An Installer package contains an archive of software to install, some meta data about location, etc., and may contain scripts which are run just before and just after the archive has been unpacked. The Installer in Mac OS X is a derivative of the Installer.app that was part of NeXTSTEP.

There are also some other commercial installers, such as VISE, but they will not be taken into account here. Mostly they are used by software distributors who used to ship software for the classic Mac operating system.

## 2 Installing $\text{\TeX}$

A front end for  $\text{\TeX}$  on Mac OS X can be a simple application residing anywhere on the system. These

are therefore generally installed by means of a simple drag and drop.

The back end is more complex. Though it can in principle be installed anywhere, it does require quite a bit of configuration afterwards, in terms of the paper size, hyphenation patterns, formats, etc. And as Mac OS X is a multi-user environment, having one  $\text{\TeX}$  install to be used by all requires these configurations to be executed such that all can use the results. E.g. the formats need to be generated at install time with administrator privileges because otherwise all users need to be able to create formats, which does not combine well with the strict authorization setup on a Unix environment.

As software installations come, any full featured  $\text{\TeX}$  back end installation is huge.  $\text{\TeX}Live$  itself spans multiple CD’s or a DVD. And even a te $\text{\TeX}$  download weighs in at over 100 MB these days.

### 2.1 te $\text{\TeX}$

It is important to stress that there would not have been an i-Installer.app, a  $\text{\TeX}$  i-Package or any  $\text{\TeX}$  redistribution by myself had it not been for Thomas Esser’s te $\text{\TeX}$ . Without his efforts as a foundation, none of this would have happened. Even today, while I make a lot of use of  $\text{\TeX}$  Live, the main `texmf` tree is still the tree from te $\text{\TeX}$ , which Thomas maintains. Although I have produced the configuration scripts for the i-Packages (see below), these make heavy use of Thomas Esser’s maintenance tools like `fmtutil`, `updmap` and `texconfig`.

### 2.2 First attempt: from te $\text{\TeX}$ sources

Initially, my wish to install  $\text{\TeX}$  came from my personal wish to use  $\text{\TeX}$  on the first developer release of Mac OS X on Apple hardware (Rhapsody).<sup>6</sup> So, my first attempt was based on downloading te $\text{\TeX}$  and compiling and installing that. The first attempt required some real porting, that is, some changes needed to be made to the te $\text{\TeX}$  code before it would compile and run on Mac OS X. My activities are generally based on the rule that I need to make my installs such that they will not take much time if I have to do them again. Every computer geek knows the frustration of a job that needs to be done once every few months or years, e.g. because one needs to remain up to date or because of a fresh system install. That is not frequent enough to remember what it was that needed to be done. My solution for this has generally been to document and publish, or to write a script. E.g. I will create some sort of

---

<sup>6</sup> Ironically, I have spent far more time on maintaining and redistributing  $\text{\TeX}$  than on using it.

INSTALL document, a set of patches, and anything else needed to complete the install. Putting that on the net is a guarantee for myself that if I have to reinstall from scratch, I can relatively easily reapply my system changes. For this strategy to work reliably, one needs to have access to all the code in the same version that was used for the actual install.

This first attempt also uncovered a problem with line endings in the  $\text{\TeX}$  code. Mac OS X is a system of mixed heritage: the classic Mac OS uses carriage return as end of line and Unix uses linefeed.  $\text{\TeX}$  is supposed to be ignorant of whitespace, but porting to Mac OS X unearthed a line-ending problem. Other problems generally had to do with compilation only. The result was documentation and patches for the installation of  $\text{te\TeX}$ .

### 2.3 Second attempt: a binary installer

But compiling  $\text{\TeX}$  from source is not for most users. In the Unix world, there are many more tech-savvy users than not, but the Mac world—the original Mac OS being as closed as it was—is the opposite in the extreme; even the concept of a command line is alien. Instructing such users to compile  $\text{\TeX}$  is more or less impossible, the language barrier is too high. Hence, as soon as the instruction-based distribution became a bit popular, the demand for a binary distribution grew. Given the fact that a  $\text{\TeX}$  install cannot be simply drag-and-drop and that the instructions led to a lot of frustrated Mac users trying to understand the technical gibberish (and asking the author for help), creating an easy installer was mandatory, to keep the support load down.

The primary choice for such an installer, Apple's Installer.app, inherited a major flaw from its archiver `pax` which had the nasty habit of not honoring symbolic links and if an archive contained a directory ‘foo’ where the system already contained a symbolic link ‘foo’, the link was happily replaced by the directory inside the archive and the link was lost.<sup>7</sup> Archivers like (GNU) `tar` honour the link and follow it to install the contents of the directory in the archive.<sup>8</sup> This change from NeXTSTEP (which uses `tar` for its installer) has been unfortunate and was the reason for creating a separate installer for  $\text{\TeX}$ .

The initial design for this installer was very simple. It would have one small window with only one button, titled ‘Install’. It would probably have been a world record in terms of simplicity for a

<sup>7</sup> Apple has since made this behavior controllable.

<sup>8</sup> At one time, this was the source for a major goof-up by Apple, where one update actually managed to damage the system it was installed on.

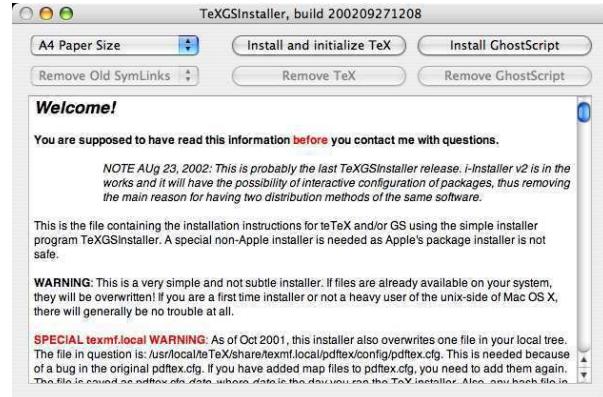


Figure 1: View of TeXGSInstaller application

GUI app, but soon it became apparent that people wanted both install and uninstall, as well as both  $\text{\TeX}$  and Ghostscript. Finally, given that  $\text{te\TeX}$  uses A4 paper for its default setting, but many users require letter-size paper, the selection of paper size became a requirement for any install that does not require many users to use the command line. And finally, the users wanted some sort of idea of progress, the absence of which makes it hard to distinguish a working from a nonworking program. The result was a simple front end to a script that unarchived the archive, and set the paper size, the output of which was captured and displayed. The application was called TeXGSInstaller.app. A screen shot is displayed in figure 1.

### 2.4 Requests and solution

Though liked for its utter simplicity and—together with the TeXShop front end—part of the combo that won the Apple Design Award for “Best Use of Open Source”, it soon turned out that more than just setting the paper size was required by many users. Another problem was that the combination of a complete  $\text{\TeX}$  and Ghostscript was large, and had to be downloaded in full every time a small part was updated. In those days, the updates were rather frequent because pdftEX was still in its initial rapid development. Besides, there were frequent requests for other additions to the redistribution. From  $\text{\TeX}$ -related tools like  $\text{\TeX}4ht$  or  $\text{XML}\text{\TeX}$  to all sorts of converters.

So, I decided to create a more generic installer: i-Installer. This installer was modelled (like Apple's) after the installer from NeXTSTEP, but with some differences:

- It is as transparent as possible. Unknown to most, Apple's Installer.app executes scripts be-

fore or after installation and these scripts may run with administrator privileges. These scripts are a potential source of security problems and I wanted users to be able to inspect them easily.

- It combines the function of downloading with installing, such that it minimizes the download to what is needed for the action selected by the user (hence the ‘i’ of i-Installer). Downloading is automatic, possibly even without user intervention, as downloading in itself is not a security problem.<sup>9</sup>
- Security is a priority.

Security is an aspect of software that is very difficult to get right. Automation implies moving actions from the conscious to the nonconscious realm. And given that a secure operation is best described as ‘conscious risk taking’ there is a true conflict of goals.<sup>10</sup> In terms of functionality, security and complexity, the following type of relation might hold:

$$\text{Security} = \frac{\text{Complexity}}{\text{Functionality}}$$

Although systems may combine functionality and security, the result will be complexity; that is, the combined complexity of what the system can do and what the user needs to do. There is no escape from this problem.<sup>11</sup>

## 2.5 i-Installer v2

The first attempt was expanded with basic functionality for interaction with the user and released as i-Installer v2 (version 2.2.0) in December 2002. A basic description follows.

Installation with i-Installer has at most four phases:

**Selection** This optional stage<sup>12</sup> allows the user to select which part of an i-Package needs to be installed. E.g. the TeX i-Package comes with various versions of the TeX programs (currently based on one of TL 2003, TL 2004 or TL 2005)

---

<sup>9</sup> Any action with a possible security risk will not be taken without user interventions, some of which are handled by the operating system, such as authentication for administrator access.

<sup>10</sup> This is the main reason why it has been so difficult for Microsoft to make their systems secure. For years they built an empire around automating indiscriminately. But while having an attachment of a mail message open automatically might save a mouse click, it also removes the consciousness from the action.

<sup>11</sup> One of the escapes is establishing trust as a simplification of the user’s part in this. In a way, this shifts the burden from the user to the system. i-Installer has been designed to accomodate both so that users have a *choice* of trusting or not. This increases the complexity for the i-Package designer (there is no free lunch).

<sup>12</sup> Available since January 2004.

and various versions of the texmf trees (based on teTeX 2 or teTeX 3). Also, one can choose to install documentation or not, etc. Selecting which parts to install of course influences what needs to be downloaded, hence, for people on slow Internet connections it pays to do a basic install instead of a full install.

**Preparation** Sometimes it is a good idea to do some preparation before unarchiving software in a certain location. E.g. if a texmf tree is rearranged it is wise to remove the old one before a new one is unarchived or one would get multiple copies of parts of the tree and it would be unpredictable which one would be used. This is why the TeX i-Package removes old stuff before unarchiving the new stuff. Of course, the removal of parts should react to the selections made in the previous phase.

**Unarchiving** This is nothing more than unarchiving the compressed tar archives in the correct locations.

**Configuration** Sometimes software must be configured. For TeX it means setting paper sizes, choosing hyphenation patterns (languages), formats, etc.

The configuration phase, if required, can be run separately.

Most of the phases (everything but unarchiving) may be interactive. Using an Apple technology called ‘Distributed Objects’, certain predefined settings (e.g. which parts of a package are to be installed) can be communicated between the main i-Installer GUI program and the subprocesses (generally Perl or shell scripts) that make up the phases. Environment variables can also be passed from phase to phase this way. Using this mechanism the install can be reasonably flexible. E.g. a choice for a simple install during selection will result in a simple configuration after unarchiving.

i-Packages have many more settings and options, like ‘required’ and ‘recommended’ dependencies, and many more possibly complex behaviours (like semi-automatic updates), which are beyond the scope of this article.

i-Installer has support for (linked) i-Directories (directories of i-Packages), dependencies, automatic background checking for updates to i-Packages and much more. Setting up a repository requires only a working web server. In fact, I run a personal repository in my personal `~/Sites` directory for testing purposes.

### 3 The TeX i-Package

#### 3.1 Layout

The TeX i-Package has a default install location `/usr/local/teTeX`, reflecting the history of starting as a pure teTeX redistribution. This location can be changed in the i-Package's properties. As it follows the teTeX layout, the texmf trees are in a subdirectory called `share`. In that subdirectory four texmf trees are found:

**texmf** This tree contains whatever a build of TeX from sources produces in a texmf tree.

**texmf.tetex** This tree contains the basic texmf tree from teTeX, which is well balanced and a reasonable size. Generally, this tree is equivalent to the latest teTeX release.<sup>13</sup>

**texmf.gwTeX** This tree contains additions to the teTeX base, such as additional fonts or macros. This tree differs depending on the install of either teTeX 2 or teTeX 3 in `texmf.tetex`.

**texmf.local** This is the standard 'local' tree of a TeX install.

The TeX i-Package was the first distribution to have the split between the program-related texmf tree (e.g. with the `.pool` files) and the basic foundation texmf tree (e.g. `texmf-dist` in TeX Live). The reason for this was that in the beginning of i-Installer there were two i-Packages for TeX: one with the programs and one with the foundation. As pdftTeX was rapidly developed, and improvements in pdftTeX were so instrumental for Mac OS X users, it was important to be able to update both separately and independently. These days, i-Packages may have many compressed tar archives and have logical sets of those which are presented to the user, so having different i-Packages for different parts is not necessary anymore.

#### 3.2 Construction

Construction of an i-Package on my own systems is managed with `make`. The TeX i-Package contains 35 different compressed tar archives.<sup>14</sup> Sets of these archives comprise the 'sets' of the TeX i-Package.<sup>15</sup> This makes it possible to have logical entities, like 'gwTeX' which are built out of different parts and

<sup>13</sup> Only as a last resort will I change anything here. The only current exception is updating the Latin Modern fonts in the teTeX 3 tree or adding those fonts to the teTeX 2 tree.

<sup>14</sup> Having the archive fragmented in many archives was added to i-Installer in September 2003. This meant that one could still have partial updates and less bandwidth requirements while having large combined i-Packages.

<sup>15</sup> Sets and the selector phase were added in January 2004.

which do not need to be downloaded entirely if only something in one part has changed.

The 35 different compressed tar archives are created by other `make` commands, which create the basis from which the archives are built. E.g. the teTeX compressed archive is unpacked in a directory, the Latin Modern fonts are removed and the Latin Modern fonts from the TeX Live repository replace them. The result is placed in a directory `/usr/local/Build/teTeX3`. From this, the `doc`, `fonts`, `tex`, etc. subdirectories end up in compressed tar archives in the i-Package and the `fonts` and `tex` compressed archives end up in the `tetex3` set while the `doc` compressed archive ends up in the `tetex3-doc` set.

#### 3.3 Contents

The contents of the TeX i-Package have not changed appreciably over the last years. It currently contains programs (binaries & scripts) based on a patched TeX Live 2003, a patched TeX Live 2004, and TeX Live 2005. TeX Live 2003 programs are combined with a texmf tree based on teTeX 2.0.2, the 2004 and 2005 versions with a texmf tree from teTeX 3.0. Both teTeX trees are updated/extended with a recent version of the Latin Modern fonts. Added to this is a set of macros and fonts in a separate gwTeX texmf tree, based on user requests. These are taken almost exclusively from the TeX Live master tree. Some of these are already available in teTeX 3, which means they are not installed when the 2004 or 2005 setup is chosen. As a result, choosing the 2003 setup with teTeX 2 may actually give you more recent TeX Live versions of certain additional macros.

A recent addition has been support for the immediate use of certain Apple fonts in pdftTeX. This support was created by Adam Lindsay and Thomas Schmitz and has been added to texmf.gwTeX. Full use requires unpacking of the Apple fonts into a TeX texmf tree by a program like `fondu`; this is done automatically if `fondu` is available. (An i-Package for `fondu` is available.)

#### 3.4 Configuration

Configuration handles choosing the paper size (for pdftTeX, dvips and dvipdfm), language patterns for LATEX, precompiled formats, font maps and adding TeX to the system-wide PATH settings for the major user command line shells. Since the OpenType versions of Latin Modern give problems in Mac OS X, the Type 1 versions are converted to TrueType by FontForge, if a recent enough FontForge is available. (An i-Package for FontForge is available.)

## 4 Other TeX-related i-Packages

There are additional i-Packages for the CM-Super fonts, the CB-Greek fonts and MusixTeX. There is an i-Package for Ghostscript, for those who need to run TeX+dvips (e.g. users of pstricks).

Since ConTeXt is very actively maintained and developed, there is a special ConTeXt updater i-Package. This i-Package installs a ConTeXt (a choice of stable versus beta is offered if the beta is available) in texmf.local. This makes it possible to uninstall the ConTeXt update and revert to whatever is in texmf.tetex. The Pragma ADE website is checked nightly for any changes to ConTeXt; if any changes are found, the ConTeXt Updater i-Package is rebuilt and uploaded automatically. Since i-Installer can inform users via mail whenever an i-Package which they have installed has been updated, this means that ConTeXt users need spend little energy to remain up to date.

## 5 Looking back

### 5.1 i-Installer

Looking back, my TeX on Mac OS X project has seen four restarts (source, binary, i-Installer v1 and i-Installer v2). Since the fall of 2002, i-Installer has seen several fundamental additions, including fragmented archives, logical sets of archives, and a substantially improved user interface based on very critical user feedback I received.<sup>16</sup>

The code has withstood the additions so far in that the application is stable enough, but it is now at the point that important additions to the functionality become difficult to add. Luckily, not many more are needed, it seems. i-Installer is heavily based on functionality that comes with Mac OS X: the availability of a Unix layer and a slew of developer frameworks which require time to master but which also make it easy to program functionality.<sup>17</sup>

---

<sup>16</sup> The best criticism I received was in e-mail that had a flame-like quality, as in “this is the worst user interface I have ever encountered, you produce a worthless program” which after some discussion resulted in such a fundamental critique that it prompted a complete overhaul of the user interface. The program and the i-Packages have been completely user driven.

<sup>17</sup> Though Apple provides rich frameworks, some make life for the developer of an installer pretty difficult. The interface to the authentication mechanism is worse than archaic and inconsistent, for instance, and i-Installer’s code in this area is complex as a result. Certain problems can be seen as directly following from Apple-provided functionality (e.g. the loss of the last bytes of the error output of subprocesses). Also, Apple has developed several improved interfaces to certain functionalities (like http traffic) over the years but backwards compatibility with older versions of Mac OS X prevents them from being used in i-Installer.

## 5.2 The TeX i-Package

The support for TeX on Mac OS X has completely run out of control. There are now 3 releases of the programs combined with 2 releases of the teTeX texmf tree available in the basic TeX i-Package.<sup>18</sup> Configuration is more than 10,000 lines of script code (mostly Perl and FontForge) and though much of the work is automated, keeping an eye on progress of building and uploading the i-Packages takes a considerable amount of time (even with a reasonably fast computer and a reasonable ADSL link). Requests for additions have dwindled to almost nothing, which gives me hope that the current teTeX + gwTeX set is close to ideal.

## 6 Looking forward

### 6.1 i-Installer

As of the current version of i-Installer, only one item is on my wish list: replacing the use of outdated Mac OS X frameworks by more recent ones, enabling, for instance, the use of authenticating proxies. Having learnt all that I have in these years provides me with a perspective with which I could design and build an even better installer, one which would support a fine-grained MiKTeX-like fragmentation of what is maintained on local disk (although this would possibly come in conflict with the very simple functionality now required of repository sites) and would be usable from the command line or via web services (authentication would be troublesome).

I hope that Apple will make i-Installer obsolete by producing a world class installer that I can use instead. In the meantime, i-Installer will remain pretty stable.

### 6.2 The TeX i-Package

For the TeX i-Package, keeping up to date with TeX and providing easy access to users to the different releases and changed functionality is difficult enough. No big changes to the current setup are planned.<sup>19</sup>

## 7 Availability and acknowledgements

The i-Installer and i-Packages are available from my TeX home page below. I would like to thank the TeX user groups and the TeX Development Fund.

◊ Gerben Wierda  
<http://www.rna.nl/tex.html>

---

<sup>18</sup> The 2004 release will probably be removed when TL 2006 development starts.

<sup>19</sup> The TeX i-Package contains both PowerPC and Intel binaries so TeX runs natively on both Apple hardware architectures. The other support i-Packages, such as Ghostscript and ImageMagick, have been made Intel-ready.

## The MacTeX distribution

Herbert Schulz

### Abstract

This article describes the contents of the MacTeX distribution included in the TeX Collection 2005. It gives an overview of the distribution, rather than an exhaustive discussion.

### 1 Introduction

For the TeX Collection 2005, the TUG Working Group for Mac OS X produced a modern TeX distribution and related applications for Mac OS X, named MacTeX. For more information than this brief overview can provide, as well as downloads of the software, please see <http://tug.org/mactex>.

The principal feature of MacTeX is an Easy Install Package, created by Jonathan Kew, containing Gerben Wierda's gwTeX re-distribution, plus XeTeX, TeXShop and related packages for those who wish to install a full, mature TeX working environment on the Macintosh platform at the click of a mouse. Many additional individual applications are included (but not installed by default) for those with more specialized or advanced needs.

### 2 A bit of history

The idea for, and initial discussion about, a TeX distribution for Mac OS X came at a "Birds of a Feather" meeting at the PracTeX 2005 conference held at Chapel Hill, North Carolina, USA. Members present in Chapel Hill were: Kaveh Bazargan, Karl Berry, Hans Hagen, Jonathan Kew, Richard Koch, Wendy McKay, Volker Schaa, and other meeting attendees. Several others could not attend directly but participated via text and audio chat<sup>1</sup> and collaborative editing:<sup>2</sup> Bob Kerstetter, Jérôme Laurens, Adam Lindsay, Ross Moore, Will Robertson, Herb Schulz, Joseph Slater and Gerben Wierda.

Discussion continued at the conference banquet that evening and the next morning Jonathan Kew presented the group with the Easy Install Package.<sup>3</sup>

This initial impetus gave rise to a TeX4Mac project with a CVS repository at [sarovar.org](http://sarovar.org) in August, 2005. The name TeX4Mac was later changed to MacTeX for aesthetic reasons. The basic directory structure was constructed and an initial selection of software to be included in this first release was gathered. We also discussed licensing restrictions:

<sup>1</sup> Using Apple's iChat AV, <http://www.apple.com/macosx/features/ichat/>.

<sup>2</sup> Using SubEthaEdit, <http://www.codingmonkeys.de/subethaedit/>.

<sup>3</sup> Word has it that Jonathan remained fully awake all day.

not all the applications that come as part of the MacTeX distribution may be included with commercial re-distributions, and source is not available for all included applications, but all may be noncommercially copied. In parallel with all this, Richard Koch started testing the Easy Install Package for compatibility with different versions of Mac OS X.

### 3 The MacTeX Easy Install Package

The MacTeX Easy Install Package contains a complete and up to date TeX system as well as Graphical User Interface (GUI) applications. The supplied GUI applications allow you to:

1. Edit, typeset and preview your documents using TeXShop as a front end to the TeX typesetting system;
2. Spell Check your documents, skipping embedded commands, using Excalibur; and
3. Build and maintain BIBTeX databases using BibDesk.

In detail:

- Installed into /usr/local, as if by i-Installer (using /Library/i-Installer/Receipts):
  - TeX, installed with options Full, TL 2005 and x86/ppc. This is Gerben Wierda's gwTeX re-distribution based on teTeX (for the foundation) and TeX Live 2005 (for the programs and scripts) with some additions. MacTeX assumes defaults for some of the configuration steps based on the user's environment; these can be changed by re-running the Configuration stage of the TeX i-package using i-Installer (see below). This distribution is ready for the announced change by Apple from PowerPC (ppc) to Intel (x86) processors during 2006.
  - XeTeX. A typesetting system based on a merger of the TeX system with Unicode and Mac OS X font technologies. More information about XeTeX can be found in Jonathan Kew's article [1].
  - CM-Super, CB Greek and MusixTeX fonts and any associated packages;
  - ConTeXt updater;
  - Ghostscript 8;
  - ImageMagick, with Freetype2 and libwmf;
  - libconv (only installed on Mac OS X 10.2; later versions of OS X already have it);
- Installed into /Applications/TeX:
  - TeXShop (version 1.40 for Panther and 2.04 for Tiger; the Tiger version is both ppc and x86 compatible);

- BibDesk (also both ppc and x86 compatible);
- Excalibur (in a sub-directory, with documentation and dictionary).
- Installed in `/Applications/Utilities`:
  - i-Installer. Gerben Wierda’s combined install, update and configure application for gw<sub>T</sub><sub>E</sub>X and other packages.

#### 4 Other folders

Separate from the Mac<sub>T</sub><sub>E</sub>X Easy Install Package are several other folders. These contain:

- documents on getting started with the installed system;
- some introductory documentation on L<sub>A</sub>T<sub>E</sub>X;
- duplicates of some of the GUI applications for those who already have a <sub>T</sub><sub>E</sub>X distribution or other special needs; and,
- some additional or alternative software.

We briefly discuss each of these.

##### 4.1 Demos

This folder has some showcases of what can be accomplished with the most popular “dialects” of <sub>T</sub><sub>E</sub>X on Mac OS X; Plain <sub>T</sub><sub>E</sub>X, L<sub>A</sub>T<sub>E</sub>X, Con<sub>T</sub><sub>E</sub>Xt, X<sub>E</sub>T<sub>E</sub>X and X<sub>L</sub>A<sub>T</sub><sub>E</sub>X.

##### 4.2 Documentation

Some URL files for further information about <sub>T</sub><sub>E</sub>X on Mac OS X on the Internet as well as copies of “A (Not So) Short Introduction to L<sub>A</sub>T<sub>E</sub>X” in several languages [2].

##### 4.3 Extras

The Extras folder contains several subdirectories:

- duplicates of several GUI programs in the Easy Install Package for those who already have a <sub>T</sub><sub>E</sub>X distribution;
- alternatives to the GUI applications in the Easy Install Package; and
- additional software that a <sub>T</sub><sub>E</sub>Xer might find useful.

The subdirectories are:

**Bibliography:** Bibliography programs for building and maintaining BIB<sub>T</sub><sub>E</sub>X databases.

**Browsers:** Programs to browse, look at documentation for and download extensions to L<sub>A</sub>T<sub>E</sub>X.

**Editors & Front Ends:** Alternate Editors, Type-setters and Previewers for <sub>T</sub><sub>E</sub>X. These range from “WYSIWYM (What You See Is What You

Mean)” to a Programmer’s Editor which interacts with the <sub>T</sub><sub>E</sub>X system via script extensions.

**Equation Editors:** These allow the user to create beautiful equations, etc., that may be exported for use in other applications; e.g., illustration and presentation software.

**Previewers:** A separate DVI and PDF previewer for use as an external viewer with other editors.

**Scripts:** Files to integrate some external programmer’s editors with the <sub>T</sub><sub>E</sub>X system.

**Spell Checkers:** <sub>T</sub><sub>E</sub>X, L<sub>A</sub>T<sub>E</sub>X and Con<sub>T</sub><sub>E</sub>Xt aware spell checkers.

#### 5 Production

The TUG working group on Mac OS X consisted of many individuals contributing time and effort assembling this first release of Mac<sub>T</sub><sub>E</sub>X, including: Karl Berry (ISO production, licensing), Peter Dyballa, Jonathan Kew (creator of the Easy Install Package), Bob Kerstetter, Richard Koch (Easy Install maintenance), Adam Lindsay, Adam Maxwell, Wendy McKay (chief instigator), Herb Schulz (editor and repository maintenance), Maarten Sneep (testing), Bruno Voisin, Gerben Wierda. Our apologies for any inadvertent omissions!

#### 6 In closing

If you find the distribution useful, please consider supporting the effort by joining TUG, <http://tug.org/join.html>, or the <sub>T</sub><sub>E</sub>X user group best for you, <http://tug.org/usergroups.html>. If you have problems, want to report bugs, etc., please join the mailing list for <sub>T</sub><sub>E</sub>X Users on Mac OS X, <http://www.esm.psu.edu/mac-tex/main.shtml#Mailinglist>. Additional contributors (development, testing, documentation) are most welcome.

#### References

- [1] Jonathan Kew, “X<sub>E</sub>T<sub>E</sub>X, the Multilingual Lion: <sub>T</sub><sub>E</sub>X meets Unicode and smart font technologies,” *TUGboat*, 26(2):115–124, 2005.
- [2] Tobias Oetiker, Hubert Partl, Irene Hyna and Elisabeth Schlegl, “The (Not So) Short Introduction to L<sub>A</sub>T<sub>E</sub>X 2<sub>E</sub>,” Version 4.14, 04 April, 2004. <http://www.ctan.org/tex-archive/info/lshort>, and `/usr/local/texTeX/share/texmf/tetex/doc/latex/general/lshort.pdf` in the gw<sub>T</sub><sub>E</sub>X distribution.

◊ Herbert Schulz  
 herbs (at) wideopenwest.com

## TeX Live for Debian

Norbert Preining

### Abstract

TeX Live is a widely used TeX distribution incorporating most of the free (in the Debian sense) packages from CTAN, and binaries for many different architecture-operating system combinations.

Debian GNU/Linux is a popular operating system distribution based on the Linux kernel, containing only free [3] programs. Like most distributions of the Linux flavor, Debian has a strong package managing facility. Since TeX Live is not packaged for any distribution (SuSE, Red Hat, ...), users and system administrators have the choice of either using the TeX system coming with the distribution, or installing TeX Live outside of the normal package management system.

It has thus been a longstanding wish to package TeX Live for Debian, so that system administrators can install TeX Live like any other Debian package. This article describes a project in this direction.

### 1 History

In times past, Sebastian Rahtz had some scripts ready, but unfortunately they were lost in a hard disk crash. In January 2005 the present author prepared the first proposal on packaging TeX Live for Debian. Within two months we had some scripts ready for building Debian packages directly from the TeX Live source repository. Sebastian Rahtz developed the scripts further, so that by March 2005 we had the first installable Debian packages of TeX Live.

In mid-May 2005 I came into contact with Frank Küster, who is responsible for packaging teTeX for Debian, and together with the Debian teTeX maintainers we developed a common base for the two TeX systems on Debian. Out of this grew improved versions of the packaging scripts, and by December we arrived at a state where we could initiate the first upload to the experimental Debian branch.

### 2 Users' point of view

From the users' point of view there is little difference between the original TeX Live distribution and the TeX Live Debian packages. The documentation of TeX Live present in the Debian packages describes typical use. Most differences arise only in the system administrators' perspective, discussed next.

### 3 System administrators' point of view

Standard TeX Live is organized in small units called *TeX packages*, each of which is described by a tpm

file. These files list the contents of the package together with dependencies and any special actions to be taken at installation time. These packages are grouped into around 30 so-called *collections*, which the administrator can select via the normal TeX Live installation procedure.

In the course of packaging TeX Live for Debian, we discussed the possibility of making a single Debian package from every tpm. This option was, however, quickly rejected, as it would have generated more than 1000 new packages. Thus we came to the conclusion that we would follow the TeX Live installation program and generate for each collection a Debian package.

Please see [1] for more details and installation instructions beyond the brief descriptions below.

#### 3.1 Basic installation

The packages have been included into the experimental branch of Debian, so system administrators of Debian GNU/Linux systems can use any Debian mirror. Or, for those who do not want Debian's experimental branch, you can use the Debian TeX Live repository on tug.org. In this case, to get the files you should add

```
deb http://www.tug.org/texlive/Debian/ pool/
deb-src http://www.tug.org/texlive/Debian/ pool/
```

to your `sources.list` file.

After this you can install the package `texlive` to get a close approximation of a normal TeX Live system (for differences see below). If you prefer a smaller installation, the Debian package `texlive-latex-recommended` is a good start. We recommend adding at least `texlive-fonts-recommended` and the needed `texlive-lang-*` packages for your desired hyphenation patterns.

#### 3.2 Further installation

At this time, several other Debian packages have dependencies on only the Debian teTeX packages and do not provide alternative dependencies onto TeX Live packages. Other packages do provide this alternative dependency. We hope this issue will be completely resolved soon. In the meantime, we provide updates to the affected packages in another repository on the TUG server. To access them, add

```
deb http://www.tug.org/texlive/Debian/ updpkg/
deb-src http://www.tug.org/texlive/Debian/ updpkg/
```

to your `sources.list` file. The list of such updated packages currently includes `lyx`, `pdfjam`, `muttpprint` and around 20 others.

## 4 Differences from TeX Live

The packaging of TeX Live for Debian has brought about some changes due to Debian constraints and paradigms. The interested reader may consult this thread [4].

A general guideline to Debian specifics can be found in the Debian TeX Policy [2].

### 4.1 Naming scheme

The names of the deb packages roughly correspond to the names of the collections in TeX Live, but have been slightly modified to conform to Debian standards (additional hyphens, country codes for the documentation packages).

### 4.2 Location of files

As in TeX Live there are several support trees used. The most important are:

```
TEXMFDIST = /usr/share/texmf-texlive
TEXMFLOCAL = /usr/share/texmf
TEXMFSYSCONFIG = /etc/texmf
TEXMFSYSVAR = /var/lib/texmf
```

### 4.3 Configuration

Debian policy states that programs must be configured via files in /etc, and that changes to the configuration must be preserved over upgrades. This led to a slightly different handling of configuration files in the Debian packages of TeX Live and teTeX.

Since TEXMFSYSCONFIG is the /etc/texmf directory, system administrators can configure the system by making a copy of any file in the main or dist tree and placing it either in TEXMFLOCAL or TEXMFSYSCONFIG.

Some files needed special attention and are handled through a Debian-specific mechanism that allows TeX Live or teTeX, add-on packages and local administrators to combine their changes. These files are the central configuration file `texmf.cnf`, the configuration file of formats `fmtutil.cnf`, the font configuration file `updmap.cfg`, and the language configuration file `language.dat`.

All of these should *not* be modified directly, but only through files in special subdirectories of /etc: `texmf.cnf` in /etc/texmf/texmf.d, `fmtutil.cnf` in /etc/texmf/fmt.d, and so on. After modifying and/or adding files in these directories, the commands `update-texmf`, `update-fmtutil`, `update-updmap`, and `update-language`, respectively, should be run, followed by `updmap-sys` and/or `fmtutil-sys` as usual.

### 4.4 Replaced packages

Many tpm packages are not included in the TeX Live Debian packages since there are equivalent and/or improved packages available in Debian (e.g. `preview`, `texinfo`, `cmsuper`). As a result, some TeX Live collections are *not* matched by a Debian package (`langarab` → `arabtex`, `langcjk` → `latex-cjk-all`, `htmlxml` → several components). Other collections are not included since they should be available as independent Debian packages (e.g. `graphicstools`, `ttfutils`).

## 5 Call for Developers

In the last month developing these packages has been a one-man show. Interested developers are encouraged to join the TeX Live for Debian developers mailing list of the `pkg-texlive-maint` project at [alioth.debian.org](http://alioth.debian.org) [5]. Help is needed!

## 6 Closing

I want to thank Sebastian Rahtz, Karl Berry, and all other TeX Live developers for the incredible amount of work they have put into TeX Live. In addition, Karl Berry and TUG deserve a big thanks for providing the web space for the Debian packages. From the Debian side I want to thank Frank Küster (the teTeX maintainer for Debian) for his permanent advice and support; without his cooperation, nothing would have been achieved.

## References

- [1] TeX Live for Debian web site.  
<http://www.tug.org/texlive/debian.html>.
- [2] Debian TeX policy. <http://people.debian.org/~frank/Debian-Tex-Policy/>.
- [3] Debian Free Software Guidelines contained in the Debian Social Contract.  
[http://www.debian.org/social\\_contract](http://www.debian.org/social_contract).
- [4] Discussion thread on the `debian-devel` mailing list. <http://lists.debian.org/debian-devel/2005/11/msg01649.html>.
- [5] TeX Live for Debian maintainers mailing list.  
<http://lists.alioth.debian.org/mailman/listinfo/pkg-texlive-maint>.

◊ Norbert Preining  
Università di Siena  
Pian dei Mantellini, 44  
53100 Siena, Italy  
preining (at) logic.at

---

## Hyphenation patterns in ConTEXt

Hans Hagen

### Abstract

A brief discussion of hyphenation patterns, exceptions, and TeX languages, especially in ConTEXt.

### 1 Pattern files

TeX has two mysterious commands that the average user will never or seldom meet:

```
\hyphenation{as-so-ciates}
\patterns  {.ach4}
```

Both commands can take multiple strings, so in fact both commands should be plural. The first command can be given any time and can be used to tell TeX that a word should be hyphenated in a certain way. The second command can only be issued when TeX is in virgin mode, i.e. starting with a clean slate. Normally this only happens when a format is generated.

The second command is more mysterious than the first one and its entries are a compact way to tell TeX between what character sequences it may hyphenate words. The numbers represent weights and the (often long) lists of such entries are generated with a special program called `patgen`. Since making patterns is work for specialists, we will not go into the nasty details here.

In the early stage of ConTEXt development it came with its own pattern files. Their names started with `lang-` and their suffixes were `pat` and `hyp`.

However, when ConTEXt went public, I was convinced to drop those files and use the files already available in distributions. This was achieved by using the ConTEXt filename remapping mechanism. Although those files are supposed to be generic, this is not always the case, and it remains a gamble if they work with ConTEXt. Even worse, their names are not consistent, and the names of some files as well as locations in the tree keep changing. The price ConTEXt users pay for this is lack of hyphenation until such changes are noticed and taken care of. Because constructing the files is an uncoordinated effort, all pattern files have their own characteristics, most noticeably their encoding.

After the need to adapt the name mapping once again, I decided to get back to providing ConTEXt specific pattern files. Pattern cooking is a special

craft and TeX users may count themselves lucky that it's taken care of. So, let's start with thanking all those TeX experts who dedicate their time and effort to get their languages hyphenated. It's their work we will build (and keep building) upon.

In the process of specific ConTEXt support, we will take care of:

- consistent naming, i.e. using language codes when possible as a prelude to a more sophisticated naming scheme, taking versions into account
- consistent splitting of patterns and hyphenation exceptions in files that can be recognized by their suffix
- making the files encoding independent using named glyphs
- providing a way to use those patterns in plain TeX as well

Instead of using a control sequence for the named glyphs, we use a different notation:

```
[sssharp] [zcaron] [idiaeresis]
```

The advantage of this notation is that we don't have to mess with spacing and parsing, and cleanup with scripts becomes more robust. The names conform to the ConTEXt way of naming glyphs and the names and reverse mappings are taken from the encoding files in the ConTEXt distribution, so you need to have ConTEXt installed.

The ConTEXt pattern files are generated by a Ruby script. Although the conversion is rather straightforward, some languages need special treatment, but a script is easily adapted. If you want a whole bunch of pattern files, just say:

```
ctxtools --patterns all
```

Or, if you want one language:

```
ctxtools --patterns nl
```

If for some reason this program does not start, try:

```
texmfstart ctxtools --patterns nl
```

When things run well, this will give you four files:

<code>lang-nl.pat</code>	the patterns in an encoding independent format
<code>lang-nl.hyp</code>	the hyphenation exceptions

<code>lang-nl.log</code>	the conversion log (can be deleted afterwards)
<code>lang-nl.rme</code>	the preambles of the files used (copyright notices and such)

If you redistribute the files, it makes sense to bundle the `rme` files as well, unless the originals are already in the distribution. It makes no sense to keep the log files on your system. When the file `lang-all.xml` is present, the info from that file will be used and added to the pattern and hyphenation files. In that case no `rme` and `log` file will be generated, unless `--log` is specified.

In the Dutch pattern file you will notice entries like the following:

```
e[ediaeresis]n3
```

So, instead of those funny (encoding specific) `~~fc` or (format specific) `\"e` we use names. Although this looks ConTeXt dependent it is rather easy to map those names back to characters, especially when one takes into account that most languages only have a few of those special characters and we only have to deal with lowercase instances.

The ConTeXt support module `supp-pat.tex` is quite generic and contains only a few lines of code. Actually, most of the code consists of a dedicated XML handler. Loading a pattern meant for EC encoded fonts in a system other than ConTeXt is done as follows:

```
\bgroup
  \input supp-pat

  \lccode"FC="FC
    \definepatterntoken ediaeresis ^~fc
  \lccode"FF="FF
    \definepatterntoken ssharp      ^~ff
  ...

  \enablepatterntokens
  \enablepatternxml

  \input lang-de.pat
  \input lang-de.hyp
\egroup
```

In addition to this one may want to set additional lower and uppercase codes. In  $\varepsilon$ -TeX these are stored with the language.

Just for completeness we provide the magic command to generate the XML variants:

```
ctxtools --patterns --xml all
```

This will give you files like:

```
<?xml version='1.0' standalone='yes'?>
<!-- some comment -->
<patterns>
  ... e&ediaeresis;n3 ...
</patterns>
```

This is also accepted as input but for our purpose it's probably best to stick to the normal method. The pattern language is a TeX specific one anyway.

## 2 Installing languages

Installing a language in ConTeXt should not take too much effort given that the language is supported. Language specific labels are grouped in `lang-*` files, like `lang-ger.tex` for the germanic languages.

Patterns will be loaded from the files in the general TeX distribution unless `lang-nl.pat` is found, in which case ConTeXt assumes that you prefer the ConTeXt patterns. In that case, run

```
ctxtools --patterns all
```

You need to move the files to the ConTeXt base path that you can locate with:

```
textools --find context.tex
```

You can also use `kpsewhich`, but the above method does an extensive search. Of course you can also generate the files on a temporary location. Now it's time to generate the formats:

```
texexec --make --all
```

Since XeTeX needs patterns in UTF-8 encoding, we provide a switch for achieving that:

```
texexec --make --all --utf8
```

Beware: you need to load patterns for each language and encoding combination you are going to use. You can configure your local `cont-usr` file to take care of this. When an encoding does not have the characters that are needed, you will get an error. When using the non-ConTeXt versions this may go unnoticed because the encoding is hard coded in the

file. Of course it will eventually get noticed when the hyphenations come out wrong.

The ConTeXt distribution has a file that holds the copyright and other notes about patterns, named `lang-all.xml`. An example description:

```
<description language='nl'>
  <sourcefile>nehyph96.tex</sourcefile>
  <title>TeX hyphenation patterns for the
    Dutch language</title>
  <copyright>
    <year>1996</year>
    <owner>Piet Tutelaers
      (P.T.H.Tutelaers at tue.nl)</owner>
    <comment>8-bit hyphenation patterns
      for TeX based upon the new Dutch
      spelling, officially since
      1 August 1996. These patterns follow
      the new hyphenation rules in the
      Woordenlijst Nederlandse Taal [...]
    </comment>
  </copyright>
</description>
```

*This file is ‘work in progress’: more details will be added and comments will be enriched.*

### 3 Commands

You can at any moment add additional hyphenation exceptions to the language specific dictionaries. For instance:

```
\language[nl] \hyphenation{pa-tiÃn-ten}
```

Switching to another language is done with the `\language` command. The document language is set with `\mainlanguage`.

If you want to let TeX know that a word should be hyphenated in a special way, you can use the `\-` command, for instance:

Con\-\TeXt

Compound words are not recognized by the hyphenation engine, so there you need to add directives, like:

```
the ConTeXt|-|system
```

If you are using XML as the input format, you need to load the hyphenation filter module. Here we assume that UTF encoding is used:

```
\useXMLfilter[utf,hyp]
```

In your XML file you can now add:

```
<hyphenations language='nl' regime='utf'>
  <hyphenation>pa-tiÃn-ten
  </hyphenation>
  <hyphenation>pa-tiÃn-ten-or-ga-ni-sa-tie
  </hyphenation>
  <hyphenation>pa-tiÃn-ten-plat-form
  </hyphenation>
</hyphenations>
```

This filter also defines some auxiliary elements. Explicit hyphenation points can be inserted as follows:

Zullen we hier  
af<hyphenate/>bre<hyphenate/>ken of niet?

The compound token can be anything, but keep in mind that some tokens are treated specially (see other manuals).

Wat is eigenlijk een  
patiÃnnten<compound token="--"/>platform?

A language is chosen with:

```
nederlands
<language code="en">english</language>
nederlands
```

If you set attribute `scope` to `global`, labels (as used for figure captions and such) adapt to the language switch. This option actually invokes `\mainlanguage`.

### 4 Languages

When users in a specific language area use more than one font encoding, patterns need to be loaded multiple times. In theory this means that one can end up with more instances than TeX can host. However, the number of sensible font encodings is limited, as is the number of languages that need hyphenation. Now that memory is cheap and machines are fast, preloading a lot of pattern files is no problem.

◊ Hans Hagen  
Pragma ADE  
pragma (at) wxs.nl

# Graphics

## Diagxy, a Lego-like diagram package

Michael Barr

### 1 Overview

This is an introduction to my `diagxy` package that sets commutative (usually) diagrams using an interface that patches pieces together, Lego style. Since it comes with full documentation/tutorial, I will just hit the high points here and refer to the package for more details.

There are a number of good packages for making commutative diagrams. Mostly, they are based on `\halign`. This has advantages and disadvantages. The advantages are that the syntax is mostly familiar and the underlying `TEX` engine does all the work in its native mode, at a considerable advantage in speed, not to mention relative ease in programming. The main disadvantage from my point of view is lack of flexibility. `TEX` makes all the spacing decisions and the user has few options. I first became aware of this when I used `Xy-pic` to make a “W” shaped diagram and discovered that, owing to a mismatch between the various nodes of the diagram, the “W” was leaning to one side. Another problem was that a fixed distance was used for arrows and if the label on an arrow was larger than that distance, it was hard to force it to be larger and the increments available (by adding nodes) were large.

In the 1980s I had designed a diagram package built around `LATEX`’s picture mode that was based on defining shapes that fit together like Lego blocks. The sizes of the shapes were under user control. This violates the `LATEX` principle of logical markup, but I do not feel that diagrams can be done using logical markup, at least not at present. The original package had defined shapes such as squares (really rectangles, but we call them squares anyway) and triangles at various orientations that would fit together. I was limited by the directions available in the picture mode (an arrow could go in only one of 48 different directions, at a resolution that varies between 3 and 14 degrees). The package was not entirely satisfactory for other reasons.

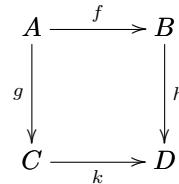
One real advantage of `Xy-pic` is that it has a full 256-character font of line segments in different directions and two fonts of arrowheads that

allow arrows in any of 512 directions, which is as many as one could wish for. It was also possible to make macros to replace the plain `TEX` arrowheads with those of `Xy-pic`, giving a more uniform look to papers, especially ones using many arrows. Therefore I had the idea a few years ago of reimplementing my original package making use of the `Xy-pic` arrows. When I sat down to do this I discovered that, underlying the `Xy-pic` matrix package was a drawing engine of great generality. In fact, it turned out to be much easier to use this engine as the back end to my reimplementation than to just use the arrows. As a result, it is possible to mix my diagram package with native `Xy-pic` code with excellent results. Also, the present `diagxy` works with both `LATEX` and plain `TEX`. Except for one incompatibility described below, `diagxy` appears to be compatible with `AMS-LATEX` as well, although I have not tested that extensively.

There are two kinds of people: those who can read and write syntax diagrams and those who learn by example. I am far out on the latter limb of what is really a continuum. Accordingly, most of the documentation of `diagxy` is in the form of a tutorial, `diaxyd.doc.tex`, that gives many examples. Even though I designed and implemented the package, I myself still refer to the tutorial all the time.

### 2 Simple diagrams

The most elementary is



which is produced using the code

```
 $$\bf bfig\bf square[A' B' C' D'; f' g' h' k']\bf efig$$
```

Here I used the ‘ symbol to separate the nodes. It seemed to be the symbol that was least likely to actually be used. If it is, it must be enclosed in braces to avoid being interpreted by `TEX` as a delimiter. Similarly, the ; was chosen as a character little used in mathematics (although unfortunately used a lot in computer programs) that separates the nodes from the arrow labels. Any of the entries can be left empty. Although the default is to place all four arrows, optional arguments allow omission of arrows too. Incidentally, square brackets are used internally as delimiters, so any square brackets should also

be enclosed in braces. Sometimes they are unnecessary, but they can never hurt and I have not actually worked out what situations require their use.

Suppose the square is not large enough. If you want the diagram

$$\begin{array}{ccc} \text{Hom}(A, B) & \xrightarrow{\text{Hom}(f, B)} & \text{Hom}(A', B) \\ \text{Hom}(A, g) \downarrow & & \downarrow \text{Hom}(A', g) \\ \text{Hom}(A, B') & \xrightarrow{\text{Hom}(f, B')} & \text{Hom}(A', B') \end{array}$$

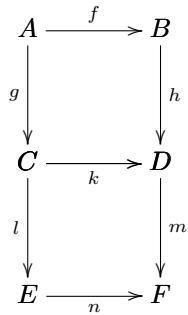
you use size parameters set off by `<` and `>` as in

```
$$\bfig\square<1000,500>[\text{Hom}(A,B)`\text{Hom}(A',B)`'\text{Hom}(A,B')`'\text{Hom}(A',B');`\text{Hom}(f,B)`\text{Hom}(A,g)`\text{Hom}(A',g)`'\text{Hom}(f,B')]\efig$$
```

Here the numbers 1000 and 500 refer to the width and height of the square in units of .01 em. This is doubtless too fine; I hardly ever use a distance not a multiple of 10 (or, occasionally, 5), but I feel that now I am stuck with that choice. The default is `<500,500>` and will be supplied if not specified.

### 3 Lego blocks

Although simple diagrams are the most common, the real power of `diagxy` is the ease of putting blocks together to make more complicated diagrams. In order to do this, there are parameters, enclosed in `()`, that tell where to place a block in the given coordinate system. For example:



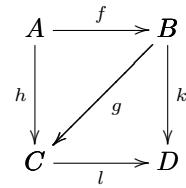
This is produced by

```
$$\bfig\square(0,500)[A`B`C`D;f`g`h`k]`\\square[C`D`E`F;`l`m`n]\efig$$
```

This is an example of the Lego block approach. Note that `C` and `D` are repeated. This is necessary in order that the arrows be the correct lengths. On the other hand, there is no need to repeat `k` and, if you did, it would by default appear both above and below the middle arrow.

There are other optional parameters to tell where the arrow labels should go and to change the arrow style. The latter can get very complicated, as complicated as `Xy-pic` allows. Arbitrary `Xy-pic` arrow styles, including arrows that follow arbitrary Bezier curves, are allowed.

There are a number of basic triangle shapes, which are named according to the letter in the alphabet that most neatly fits in the triangle. For example, a `\ptriangle` is a right triangle whose hypotenuse goes from east to south and an `\atriangle` is an isosceles triangle whose base is horizontal. This example



is produced by

```
$$\bfig\ptriangle[A`B`C;f`h`g]`\\dtriangle[B`C`D;`k`l]\efig$$
```

An alternate way of making the same diagram is given by

```
$$\bfig\square[A`B`C`D;f`h`k`l]`\\morphism(500,500)<-500,-500>[B`C;g]\efig$$
```

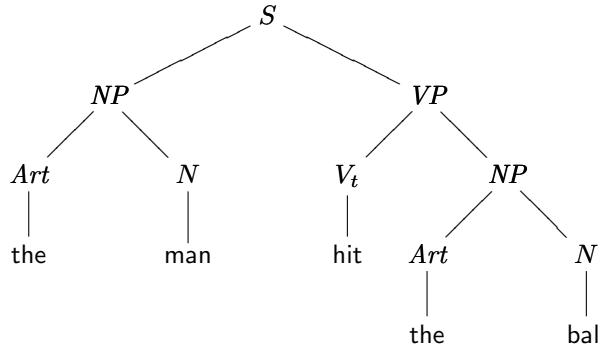
where `\morphism` creates the arrow from `B` to `C`.

Incidentally, the effect of `\bfig... \efig` in a displayed diagram is nearly the same as `\xy ... \endxy` except that it has the effect of enclosing the whole in a `\vcenter` box so that equation numbers, if any, are vertically centred. In case you are curious about the names `\bfig` and `\efig`, I used an NROFF-type system briefly in the very early 1980s (in the paleolithic era before `TeX`) and figures were set off using `.BFIG` and `.EFIG`.

### 4 In-line arrows

One of my minor gripes about `LaTeX` is that the arrows used in text look quite different from those used in diagrams. So one of the things I have done is define a macro `\to` that works somewhat like `\rightarrow` but uses the `Xy-pic` arrowheads. It has several other features, including various options, but the most important is that it grows to accommodate long labels. An example is  $A \xrightarrow{\text{a long label}} B$ , for which the source is

```
$A\to^{\{\rm mbox{\rm \{}}\rm a long label\}}B$.
```



**Figure 1:** A tree diagram from mathematical linguistics

There are other macros, such as `\two` and `\three`, which make double and triple arrows. The directions of the arrows (right or left) can be independently set, using standard XY-pic controls.

## 5 Compatibility

There is nothing in this code (or in XY-pic) that uses anything but plain TeX. I have used it extensively under LATEX and never unearthed any incompatibility. I have not used it much with AMS-LATEX but there is one known incompatibility. One of the AMS symbol fonts makes a little box which is named `\square`. A simple fix is to load the `amsfonts` first. If you actually require that character, then before loading `diagxy`, simply say `\let\Box\square`, to call it `\Box`. Incidentally, the @ sign is used in the XY-pic syntax, along with just about every other non-alphanumeric character. Thus you cannot change the catcodes of any of them, which means there are no private control sequences. Thus one must be careful not to redefine any of the internal sequences used.

## 6 An alternate syntax

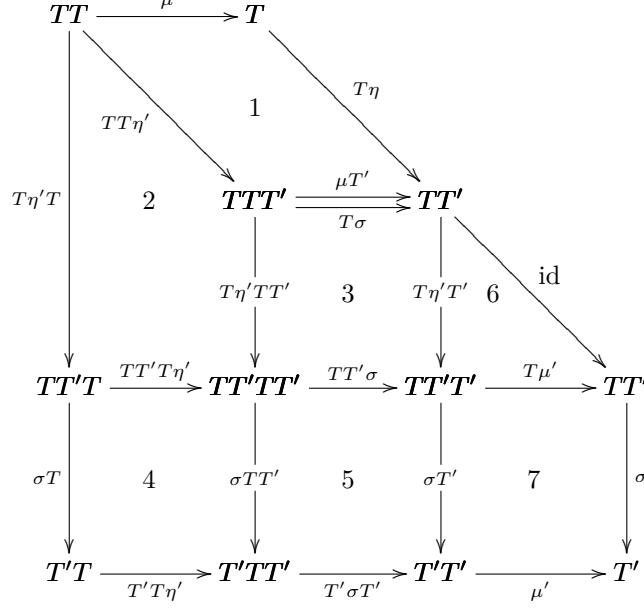
A couple years after this package was released, I received a note from a graduate student named Gerd Zeibig who suggested an alternate syntax in which you specify the placement of nodes in the coordinate system and then draw arrows between them. He had also implemented this in a way that used two counters for each node. Given the shortage of counters in standard LATEX, I decided to reimplement it using macro definitions in place of these counters. So it is now possible to describe diagrams as illustrated below. The diagram in Figure 1 appears in a set of notes on mathematical linguistics. While there is certainly no problem producing it using the ear-

lier code, it is much more systematic to describe trees in this way:

```

$$\bfig
\newcommand{\NP}{\hbox{\mit NP}}
\newcommand{\VP}{\hbox{\mit VP}}
\newcommand{\Art}{\hbox{\mit Art}}
\node 1a(0,0)[S]
\node 2a(-600,-300)[\NP]
\node 2b(600,-300)[\VP]
\arrow/-/[1a`2a;]
\arrow/-/[1a`2b;]
\node 3a(-900,-600)[\Art]
\node 3b(-300,-600)[N]
\node 3c(300,-600)[V_t]
\node 3d(900,-600)[\NP]
\arrow/-/[2a`3a;]
\arrow/-/[2a`3b;]
\arrow/-/[2b`3c;]
\arrow/-/[2b`3d;]
\node 4a(-900,-900)[{\sf the}]
\node 4b(-300,-900)[{\sf man}]
\node 4c(300,-900)[{\sf hit}]
\node 4d(600,-900)[\Art]
\node 4e(1200,-900)[N]
\arrow/-/[3a`4a;]
\arrow/-/[3b`4b;]
\arrow/-/[3c`4c;]
\arrow/-/[3d`4d;]
\arrow/-/[3d`4e;]
\node 5a(600,-1200)[{\sf the}]
\node 5b(1200,-1200)[{\sf ball}]
\arrow/-/[4d`5a;]
\arrow/-/[4e`5b;]
\efig$$
  
```

First you define the nodes and then draw arrows between them. In this case, we wanted only lines, whence the `/-` specification on the arrows. The labels on the nodes are almost completely arbitrary (limited only by what `\csname ... \endcsname` allows).

**Figure 2:** A larger diagram

## 7 A large diagram

Figure 2 shows a large diagram that is taken directly from the book *Toposes, Triples and Theories* by Michael Barr and Charles Wells, published by Springer Verlag in 1984. It may well have been the very first book produced using L<sup>A</sup>T<sub>E</sub>X, which was not released until 1985. Using diagxy, it can be produced with the following code.

```
$$\bfig
\scalefactor{1.4}
\qtriangle(0,1000)/>'>/[TT'T' TTT;
;\mu' TT\eta',]
\btriangle(500,1000)/>'@<-14\ul>/%
[T' TTT,'TT';T\eta'T\sigma]
\morphism(0,1500)|1|/>|<0,-1000>%
[TT'TT;T\eta'T]
\square(500,500)|ammx|/@<14\ul>'>'/[TT,'TT' TT' TT;
\mu T'T\eta' TT' T\eta' T];
\morphism(1000,1000)|r|/>|<500,
-500>[TT'TT; \hbox{\rm id}]
\square/>'>'>/[TT'T' TT' T'T'T'
TT';TT'\eta',\sigma T'T'T\eta']
\square(500,0)|ammb|[TT'TT' TT'T'
T'TT'T'; TT'\sigma\sigma\sigma TT'
\sigma T'T'\sigma T']
```

```
\square(1000,0)/>'>'>/[TT'T' TT'
T'T' T';T\mu' '\sigma'\mu']
\place(500,1250)[1]
\place(215,1000)[2]
\place(750,750)[3]
\place(215,250)[4]
\place(750,250)[5]
\place(1140,750)[6]
\place(1250,250)[7]
\efig$$
```

## 8 Availability

diagxy can be found on CTAN in the directory *macros/generic/diagrams/barr/*, and at my own ftp site at <ftp.math.mcgill.ca/pub/barr/> *diagxy.zip*.

◇ Michael Barr  
Dept. of Math. and Stats.  
McGill University  
805 Sherbrooke St. W  
Montreal, QC H3A 2K6  
Canada  
**mbarr (at) barrs.org**

---

## Embedding fonts in MetaPost output

Troy Henderson

### Abstract

MetaPost [1, 2, 3] is a powerful graphics language (by John Hobby) based on Donald Knuth's METAFONT [4] with high quality PostScript output. An outstanding feature of MetaPost is that typeset fonts in the output graphics are consistent with those in TeX-based documents. However, MetaPost does not embed these fonts inside the output PostScript files. This article addresses a specific technique for performing such an embedding.

### 1 Introduction

MetaPost is one of the most elegant means for generating high quality vector graphics. The language itself is very mathematical in nature and consists of statements that draw and fill paths and label typeset text. The very name itself indicates that MetaPost is a language that generates another language, namely PostScript. With PostScript as output, the graphics are perfectly scalable to any arbitrary resolution. John Hobby, its author, writes:

“[MetaPost] is really a programming language for generating graphics, especially figures for TeX [5] and troff documents.”

This quote by Hobby indicates that MetaPost figures are not only intended to be embedded inside of TeX-based documents but also require TeX to be complete. This is apparent when examining the PostScript containing typeset text which is output by MetaPost. MetaPost does not embed the fonts used inside of the output files. The philosophy for this is that there is no real need for such embedding if the figures are going to appear inside a TeX document because TeX itself will embed the necessary fonts. However, as with any programming language, the source is often compiled (and viewed) several times before the user completes each figure. So, at least for debugging purposes, self-contained output is often desired. That is, we often want PostScript output which has all necessary fonts embedded.

### 2 Embedding the fonts

According to the statement above, a naïve approach to performing this embedding is to simply include the figure inside a TeX-based document, run TeX, and use Dvips to generate the stand-alone PostScript graphic. These steps highlight the embedding process; however, several details must be addressed in order to create a fully functional approach. In particular, in order to embed the figure into a TeX

document, the size of the figure must first be determined so that the paper size of the resulting DVI document is correct. If the paper size is too small, then the TeX→Dvips process will clip the figure. Determination of the appropriate paper size can be done either during or after the MetaPost process.

To make this concrete, suppose we have a MetaPost file `foo.mp` with the contents:

```
beginfig(1)
    draw commands
endfig;
beginfig(2)
    draw commands
endfig;
:
end
```

The Perl script `mpstoeps`, available from

[http://ctan.org/tex-archive/graphics/  
metapost/contrib/tools/mpstoeps/](http://ctan.org/tex-archive/graphics/metapost/contrib/tools/mpstoeps/),

can automate the above process. `mpstoeps` assumes that the filename of each figure is of the form `foo_1.mps`, `foo_2.mps`, ... as opposed to the canonical `foo.1`, `foo.2`, ... naming scheme. `mpstoeps` transforms a MetaPost figure into a stand-alone EPS in a method explained in greater detail in Section 3. Furthermore, `mpstoeps` tightens the bounding box of the resulting EPS so that it matches that of the original MetaPost output.

### 3 Nuts and bolts

As an alternative to the *magical* `mpstoeps`, we may also use MetaPost itself to determine the width and height of the paper size needed to include the figure in a TeX document. As a first step in accomplishing this task, we place

```
numeric w,h;
w := xpart urcorner bbox currentpicture
      - xpart llcorner bbox currentpicture;
h := ypart urcorner bbox currentpicture
      - ypart llcorner bbox currentpicture;
```

immediately before the `endfig` statement. Once these lengths `w` and `h` are determined, we continue by identifying a good basename for the working files.

```
string base;
base:=jobname&"_&decimal(charcode);
```

We now begin writing an external LATEX file which will use the `geometry` and `graphicx` packages.

```
write "\documentclass{article}" to base&".tex";
write "\usepackage{geometry}" to base&".tex";
write "\usepackage{graphicx}" to base&".tex";
```

The `geometry` package is used to guarantee that the output will have the precise geometry (i.e. paper size, margins, etc.) needed.

```
write "\geometry{papersize={"
  & decimal(ceiling(w)) & "bp,"
  & decimal(ceiling(h)) & "bp}}"
  to base&".tex";
write "\geometry{margin={0bp,0bp}}"
  to base&".tex";
write "\geometry{noheadfoot,nomarginpar}"
  to base&".tex";
```

Once these preliminaries for the L<sup>A</sup>T<sub>E</sub>X document are established, we then insert the MetaPost output file and complete the document.

```
write "\begin{document}" to base&".tex";
write "\thispagestyle{empty}" to base&".tex";
write "\noindent\includegraphics{"
  & jobname & "." & decimal(charcode) & "}"
  to base&".tex";
write "\end{document}" to base&".tex";
write EOF to base&".tex";
```

Now that the document is complete, we output a few messages so that the user knows the precise commands to execute in order to create the stand-alone EPS. This must be done because MetaPost (for security reasons) will not call external commands.

```
message "=====";
message "Execute these commands to generate "
  & base&".eps";
message "latex " & base& ".tex";
message "dvips -E -T " & decimal(ceiling(w))
  & "bp," & decimal(ceiling(h))
  & "bp -q -o "&base&".eps "&base&".dvi";
message "=====";
message "";
```

It is worth noting that even though this process uses L<sup>A</sup>T<sub>E</sub>X, the MetaPost process itself does not necessarily use L<sup>A</sup>T<sub>E</sub>X to process the text. On most MetaPost installations, the default processor for text labels is plain T<sub>E</sub>X. Furthermore, the above process for embedding fonts usually increases the bounding box of the figure by at least 1 bp on each side, unlike `mpstoeps`(which simply preserves the original bounding box).

#### 4 Typesetting fonts using L<sup>A</sup>T<sub>E</sub>X

As previously mentioned, for most MetaPost distributions, the default processor for text labels is plain T<sub>E</sub>X. However, some users find it convenient to use L<sup>A</sup>T<sub>E</sub>X to process the text. For example, L<sup>A</sup>T<sub>E</sub>X users are accustomed to using `\frac{a}{b}` to create

fractions; this command (as well as many others) is not available in T<sub>E</sub>X. A typical MetaPost source file `bar.mp` which uses L<sup>A</sup>T<sub>E</sub>X to process the text may be organized in the following manner.

```
verbatimtex
\documentclass{amsart}
\begin{document}
etex
beginfig(1)
  draw commands
endfig;
beginfig(2)
  draw commands
endfig;
:
end
```

However, this format for `bar.mp` alone is not sufficient to force L<sup>A</sup>T<sub>E</sub>X to process the text. The `mpost` command must also be instructed to use L<sup>A</sup>T<sub>E</sub>X. This can be done via

```
mpost -tex=latex bar.mp.
```

To make this preference the default under teT<sub>E</sub>X, we set the environment variable `TEX=latex`.

#### 5 Working example

We now illustrate the processes mentioned above by applying them to a simple MetaPost figure. We will use two copies of the figure—one with `mpstoeps` and the other with the process described in Section 3. Both figures will be defined in a MetaPost source file `tri.mp`.<sup>1</sup> In order to use L<sup>A</sup>T<sub>E</sub>X to process the text labels, we preface the code with:

```
verbatimtex
\documentclass{article}
\begin{document}
etex
```

We then draw the figure with the following commands:

```
picture pict;
beginfig(1)
  u:=36;
  w:=fontsize defaultfont;
  x1=u;x2=u*cosd(120);
  y1=0;y2=u*sind(120);
  draw (x1,y1)--(x2,y2)--(x2,-y2)--cycle;
  label(btex $a$ etex,(x1-w,y1));
  label.lft(btex $A$ etex,(x2,y1));
  label(btex $b$ etex,(x2-w*cosd(120),
```

---

<sup>1</sup> The electronic version of this article contains `tri.mp` embedded as an attachment to the PDF.

```

y2-w*sind(120));
label.lrt(btex $B$ etex,((x1+x2)/2,-y2/2));
label(btex $c$ etex,((u-w)*cosd(120),
(-u+w)*sind(120)));
label.urt(btex $C$ etex,((x1+x2)/2,y2/2));

```

We store the picture into `pict` since we want to reuse it in the next figure. We then “reload” it into the next figure by

```

beginfig(2)
currentpicture:=pict;
write commands from Section 3
message commands from Section 3
endfig;

```

Finally, we append the canonical closing statements for MetaPost.

```
message "";
```

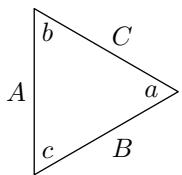
Once `tri.mp` is compiled, we rename `tri.1` to `tri_1.mps` and apply `mpstoeps`. This provides the stand-alone EPS `tri_1.eps` with all fonts embedded:

```
mv tri.1 tri_1.mps
mpstoeps tri_1.mps
```

Furthermore, we are also instructed to execute

```
latex tri_2.tex
dvips -E -T 69bp,67bp -q -o tri_2.eps tri_2.dvi
```

After following these steps, we obtain `tri_2.eps`, which is virtually identical to `tri_1.eps`, and both of these EPS files have their fonts embedded. This mutual figure is shown below:



**Figure 1:** Output of `tri.mp`.

## 6 Conclusion

It is worth mentioning that although both stand-alone EPS graphics in Section 5 look virtually identical to the original MetaPost output, they are significantly larger in file size. This drastic difference is clearly due to size of the embedded fonts.

Also, renaming the MetaPost output files using the `.mps` naming scheme is a convenient method for using either L<sup>A</sup>T<sub>E</sub>X or pdflAT<sub>E</sub>X to compile the document. The latter does not allow arbitrary PostScript graphics, but does support MetaPost output—as long as the file is renamed with extension `.mps`.

As a final note, arbitrary EPS files must first be converted to PDF before they can be included with pdflAT<sub>E</sub>X. Thanks to Hans Hagen, many distributions of T<sub>E</sub>X now include a utility called `mptopdf` which provides a method of easily converting such graphics to PDF.

## References

- [1] J. D. Hobby. Drawing graphs with MetaPost. Technical Report 164, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Also available at <http://www.tug.org/docs/metapost/mpgraph.pdf>.
- [2] J. D. Hobby. Introduction to MetaPost. EuroT<sub>E</sub>X '92 Proceedings, pages 21–36, 1992. Also available at <http://www.tug.org/docs/metapost/mpintro.pdf>.
- [3] J. D. Hobby. A user’s manual for MetaPost. Technical Report 162, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Also available at <http://www.tug.org/docs/metapost/mpman.pdf>.
- [4] D. E. Knuth. *METAFONT: The Program*, volume D of *Computers and Typesetting*. Addison Wesley, Boston, 1986.
- [5] D. E. Knuth. *The T<sub>E</sub>Xbook*, volume A of *Computers and Typesetting*. Addison Wesley, Boston, 1986.

◊ Troy Henderson  
Dept. of Mathematical Sciences  
United States Military Academy  
West Point, NY 10996  
USA  
troy@tlhiv.org  
<http://www.tlhiv.org>

## Hints & Tricks

### Glisternings

Peter Wilson

Seagulls scream upon the shorelines' wrack  
And seals abound  
Amid the setting sun's glistering track  
Across the Sound.

*Puget Sound*

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. Speaking of which, David Elliott was the first<sup>1</sup> to point out that in the last column [6] I mistakenly attributed Matthew Arnold's poem *Dover Beach* to Tennyson. I have no idea why I should have done that.

The three topics which are the subject of this month's column have all been suggested by readers.

They cannot scare me with their empty spaces  
Between stars — on stars where no human race is.

*Desert Places*, ROBERT FROST

### 1 Empty arguments

In an earlier column [5] I talked about how to check if two strings were the same, that is, that they consisted of the same characters in the same order. A recent query on the *texhax* mailing list [2] asked about how to check if an argument was empty, which on the face of it is just a check comparing a string with an empty string. However the earlier approach does not work in this case.

In L<sup>A</sup>T<sub>E</sub>X there is often a need to check if an optional argument is present or not. The typical form for this is:

```
\newcommand{\amacro}[1][\@empty]{...  
  \ifx\@empty#1 ... % no optional  
  \else ...          % optional not \@empty
```

The question at hand, though, is what is the proper replacement for the pseudo code in the second line below?

```
\newcommand{\amacro}[1]{...  
  \if(#1 is empty) ... % no argument
```

<sup>1</sup> My wife was a close second.

```
\else ...          % argument not empty
```

where 'empty' means zero or more spaces. Thus {} and { } both qualify as 'empty'. If you are a L<sup>A</sup>T<sub>E</sub>X user then the *ifmtarg* package on CTAN provides a solution. For T<sub>E</sub>X users here is the equivalent code, noting that all the macro definitions before the \begingroup are a regular part of L<sup>A</sup>T<sub>E</sub>X.

```
\def\makeatletter{\catcode`\@11\relax}  
\def\makeatother{\catcode`\@12\relax}  
\makeatletter  
\long\def@gobble #1{}  
\long\def@firstofone#1#{#1}  
\long\def@firstoftwo#1#2#{#1}  
\long\def@secondoftwo#1#2#{#2}  
\begingroup  
\catcode`\Q=3  
\long\gdef@ifmtarg#1{  
  \Q\xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}  
\long\gdef@ifmtarg#1#2Q#3#4#5\@nil{#4}  
\long\gdef@ifnotmtarg#1{  
  \Q\xifmtarg#1QQ\@firstofone\@gobble\@nil}  
\endgroup  
\makeatother
```

The useful parts of this are

```
\@ifmtarg{(arg)}{\{empty code\}}{\{not empty code\}}  
\@ifnotmtarg{(arg)}{\{not empty code\}}
```

For example these could be used like

```
\def\isempty#1{\@ifmtarg{#1}{EMPTY}{FULL}}  
\def\isnotempty#1{\@ifnotmtarg{#1}{FULL}}%  
\def\mt{}  
\isempty{}      -> EMPTY  
\isempty{}      -> EMPTY  
\isempty{\mt}   -> FULL  
\isempty{ E }   -> FULL  
\isnotempty{}   ->  
\isnotempty{}   ->  
\isnotempty{\mt} -> FULL
```

The *ifmtarg* package originally had a much simpler approach until Donald Arseneau pointed out the error of my ways. The perils of empty were discussed in the late Michael Downes' *Around the Bend* series; the one in question is available from CTAN in *info/aro-bend/answer.002*

Faultily faultless, icily regular, splendidly null,  
Dead perfection, no more.

---

*Maud*, ALFRED, LORD TENNYSON

### 2 The usefulness of nothing

Another respondent on *texhax* [1] wanted an even-page version of L<sup>A</sup>T<sub>E</sub>X's *\cleardoublepage*. It might appear that a *\cleardoublepage*, which will get you to the next odd-numbered page, followed by a

\clearpage or \newpage will then get to an even-numbered page, but this is not so as you will find that you can't move on from a page with nothing on it (excepting headers and footers). What is required is something that appears to be a nothing or a null but which is not, so far as T<sub>E</sub>X is concerned. For the purposes at hand an empty box will do. T<sub>E</sub>X has a \null command, which is shorthand for an empty horizontal box, but we can use something with wider applicability which I will name \nowt.<sup>2</sup>

```
\newcommand*{\nowt}{\leavevmode\hbox{}}
% \nowt <==> \mbox{}

\newcommand{\cleartoevenpage}[1][\empty]{
  \clearpage
  \ifodd\c@page
    \nowt\ifx\empty#1\else #1\fi
    \newpage
  \fi}
```

This clears the current page and if the next is not an odd one then the task is finished. Otherwise we put (the invisible) \nowt on the odd page we have reached and move on to the next one, which will be even. The optional argument can be used to put some text or illustration on the skipped over odd page. For instance:

```
\cleartoevenpage[%
  \vfill\centering THIS PAGE LEFT BLANK\vfill
  \thispagestyle{empty}]
```

where the phrase 'THIS PAGE LEFT BLANK' will be centered on the odd page, and there will be neither a header nor a footer.

If you have ever tried something like this:

```
\begin{description}
\item[Nothing] \\
  Also known as \ldots
...
```

then you probably got an error message saying:  
There's no line to end here.

This can be resolved by putting \nowt just before the \\ newline command.

We may be in some degree whatever character we choose.

---

*London Journal, JAMES BOSWELL*

### 3 Picking characters

A texhax reader [4] wanted a macro that would ensure that the first letter of a string would be in uppercase. Various answers were supplied and I'm providing a couple of my own. All the solutions depend

---

<sup>2</sup> 'Nowt' is a Northern English dialect word meaning naught or nothing as in "Y' can't get owt fer nowt" — You can't get something for nothing.

on the fact that a T<sub>E</sub>X macro takes as a single argument either braced text or a single token, where a token is either a command name (the name of a macro) or a single character. Further, when defining a T<sub>E</sub>X macro the argument list is ended by a token, which is usually the initial opening brace of the definition.

Here's my first, long winded solution.

```
\def\gettwo#1#2\nowt{%
  \gdef\istchar{#1}\gdef\restchars{#2}}
\def\splitoff#1{\gettwo#1\nowt}
\def\Upfirst#1{\splitoff{#1}%
  \MakeUppercase{\istchar}\restchars}
```

\gettwo expects two arguments with the end of the second denoted by '\nowt' (I have chosen this on the assumption that it will not be part of either argument; any other command name that would not be in the arguments would serve as well). The macro \splitoff takes a single (string) argument and passes it on to \gettwo, which then takes the first character in the string as its first expected argument, and the rest of the string as its second argument. It globally defines \istchar and \restchars as the two arguments. \Upfirst takes a string argument, calls \splitoff, and hence \gettwo, and then ensures that \istchar is typeset in uppercase, followed by the rest of the characters.

This does not work if the argument to \Upfirst is a macro that is defined as a string (for example \def\arg{string}). This can be resolved by using T<sub>E</sub>X's \expandafter command to make sure that \Upfirst's argument is expanded<sup>3</sup> before being used by \splitoff:

```
\def\Upfirst#1{%
  \expandafter\splitoff\expandafter{#1}%
  \MakeUppercase{\istchar}\restchar}
```

The second version, below, is not as versatile as the first as the string is consumed internally instead of being made available in the form of the \istchar and \restchars macros.

```
\def\upperfirst#1#2\nowt{%
  \MakeUppercase{#1}\MakeLowercase{#2}}
\def\Upfirst#1{\expandafter\upperfirst#1\nowt}
```

The basic idea is the same as the first proposal. It has the added function of ensuring that only the first character in the string is uppercase (it lowers the remainder). Neither solution can handle the case where the first character is a ligature (e.g., \oe) or accented (e.g., \^a), or other commands.

Uwe Lück [3] provided a more complete but more complex solution.

```
\DeclareRobustCommand{\Upfirst}[1]{%
```

---

<sup>3</sup> To one level only.

```
\protected@edef\upfirst@rg{#1}%
\expandafter\upit\upfirst@rg\nowt}
```

Using `\DeclareRobustCommand` instead of `\def` or `\newcommand` ensures that `\Upfirst` can be used in a moving argument without having to be protected. The `\protected@edef` is used to expand the argument while maintaining any `\protect`s. In order to handle an accented initial character the string has to be split into three parts: the first element (which may be a character or an accent command), the second (which may be the argument to an accent command), and the third is the remainder of the string. The string, by the way, must have at least two characters.

```
\def\upit#1#2#3\nowt{%
\let\@uptokone#1%
\let\@xuptoktwo\@empty
\def\@yuptoktwo{#2}%
\expandafter\test@ccent\@ccentlist\@sentinel
\MakeUppercase{#1\@xuptoktwo}%
\MakeLowercase{\@yuptoktwo#3}}
```

The `\upit` macro takes three arguments, which are then the three portions of the initial string, and stores the first two in `\@uptokone` and `\@yuptoktwo` respectively. The macro `\test@accent` determines if the first token is an accent, changing `\@xuptoktwo` and `\@yuptoktwo` if it is.

```
\def\@ccentlist{'>"\`\'\b\c}" plus the rest
\def\test@ccent#1{%
\ifx#1\@sentinel\else
\ifx\@uptokone#1
\let\@xuptoktwo\@yuptoktwo
\let\@yuptoktwo\@empty
\fi
\expandafter\test@ccent
\fi}
```

`\@ccentlist` is a list of the accent commands; if a string is likely to start with an analphabetic character, such as an opening quote (`), then these characters should also be included in the list.

The `\test@ccent` macro iterates through the list of accent commands and characters supplied as its argument and if there is a match with `\@uptokone` then it swaps the `\@xuptoktwo` and `\@yuptoktwo` values. The end result is that if the initial string starts with an accent then `\@xuptoktwo` has the accented character and `\@yuptoktwo` is empty, otherwise `\@xuptoktwo` is empty and `\@yuptoktwo` has the second character in the string.

Following are some examples using the last definition of `\Upfirst`.

```
low UP \& \Upfirst{low UP} ->
low UP & Low up
\def\stuff{rAnDoM 26 sTuFf}
\stuff{} \& \Upfirst{\stuff} ->
rAnDoM 26 sTuFf & Random 26 stuff
\oe{}rstead \& \Upfirst{\oe{}rstead} ->
œrstead & œrstead
\c{c}edilla \& \Upfirst{\c{c}edilla} ->
cedilla & Çedilla
\emph{strong} \& \emph{\Upfirst{strong}} ->
strong & Strong
‘quote’ \& \Upfirst{‘quote’} ->
‘quote’ & ‘Quote’
>que? \& \Upfirst{>que?} ->
¿que? & ¿Que?
```

As always, if you are doing things with macros that include `@` in their name, either put the code into a package (.sty) file or enclose the code in a `\makeatletter ... \makeatother` pair.

Perhaps next time I'll take a look at traversing a string character by character and other kinds of looping macros but on the other hand, perhaps not.

## References

- [1] Susan Dittmar. Variant of `\cleardoublepage` starting on even page numbers. Post to `texhax` mailing list, 18 August 2005.
- [2] Adam Fenn. Empty arguments. Post to `texhax` mailing list, 17 August 2005.
- [3] Uwe Lück. Re: [texhax] read and process single characters. Post to `texhax` mailing list, 24 June 2005.
- [4] Torsten Wagner. Read and process single characters. Post to `texhax` mailing list, 24 June 2005.
- [5] Peter Wilson. Glisterings. *TUGboat*, 22(4):339–341, December 2001.
- [6] Peter Wilson. Glisterings. *TUGboat*, 25(2):201–202, 2004.

◊ Peter Wilson  
18912 8th Ave. SW  
Normandy Park, WA 98166  
USA  
herries.press (at) earthlink.net

## Pearls of TeX programming

The title of the BachoTeX 2005 conference was “The Art of TeX Programming,” TAOTP for short, therefore the idea of a “Pearls of TeX Programming” session arose. Bogusław Jackowski came up with the session motto: “Behold”—Bhaskara (see, e.g., <http://www.aurora.edu/mathematics/bhaskara.htm>).

The idea was to invite TeXies known to be TeXperts, TeX Masters or perhaps even TeX Grandmasters<sup>1</sup> to contribute.

The call stated what was wanted:

- a short TeX, METAFONT, or METAPOST macro or macros (preferably a few lines)
- results should be virtually useful yet not obvious
- easy to explain: 10 minutes at most

**Editor’s note:** This article is reprinted with permission from the proceedings of BachoTeX 2005: *Bulletyn Polskiej Grupy Użytkowników Systemu TeX*, Zeszyt 22, May 2005, T. Przelewski, B. Lichoński, S. Wawrykiewicz, P. Bolek, eds. Reprinted with permission.

<sup>1</sup> Of course the blame for a failure to contact somebody fitting this description should be put at the doorstep of the conference organizers.

Prospective contributors were asked to kindly provide the source of a macro or macros and a display or short description of the result, the size of it to be altogether not more than one A4 page, preferably—half of an A4.

We also stated that this is not a contest and that contributions were requested even from authors who are unable to attend the conference. In such a case the author was free either to elect one of the participants to present his work or “leave the proof to the gentle reader” aka “Behold”. The latter can be done anyway...

As can be seen from the examples, we did not strictly adhere to the stated program/macro limitations, with the notable exception being Frank Mittelbach’s contribution. The result is here for the gentle reader to digest and profit from.

We intend to continue the TAOTP initiative at future BachoTeX conferences: TeX has so much more up its sleeves ... A web display, similar in spirit to the “TeX Showcase” maintained by Gerben Wierda (at <http://tug.org/texshowcase>), is also being considered for the future.

## barbara beeton

### New symbols from old

Sometimes one needs a symbol that can’t be found in any font, but that is either a rotation or a reflection of a symbol that *is* available. The `graphicx` package to the rescue!

```
\newcommand{\reflectit}[1]{\reflectbox{\ensuremath{#1}}}
\newcommand{\turnover}[1]{\rotatebox{origin=c}{180}{\ensuremath{#1}}}
\newcommand{\turnne}[1]{\rotatebox{origin=c}{45}{\ensuremath{#1}}}
\newcommand{\turnnw}[1]{\rotatebox{origin=c}{135}{\ensuremath{#1}}}
\newcommand{\turnsw}[1]{\rotatebox{origin=c}{225}{\ensuremath{#1}}}
\newcommand{\turnse}[1]{\rotatebox{origin=c}{315}{\ensuremath{#1}}}
```

$\leq \geq : \leq \geq;$        $\Rightarrow : \nearrow \nwarrow \swarrow \nearrow;$        $\sim : \sim$

When you define new names for such symbols, it’s a good idea to specify the class (`\mathord`, `\mathbin`, etc.) in the definition so you get the correct spacing when they’re used.

## Martin Schröder

Colour separation in pdfTEX

```
\newcommand*{\AC@addColor}[5]{%
  \immediate\pdfobj stream
    attr {
      /FunctionType 4
      /Domain [0.0 1.0]
      /Range [0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0]
    }
  { { dup ?2 mul exch
      dup ?3 mul exch
      dup ?4 mul exch
      ?5 mul } }%
\edef\AC@ColorFunctionObj{\the\pdflastobj}%
\immediate\pdfobj { [ /Separation /?1
      /DeviceCMYK
      \AC@ColorFunctionObj\space 0 R ]}%
\begingroup
\toks@\expandafter{\AC@colorhook}%
\edef\x{%
  \endgroup
  \gdef\noexpand\AC@colorhook{%
    \the\toks@
    /?1\space\the\pdflastobj\space 0 R %
  }%
}%
\x
}
% later
\edef\AC@expand{\global\pdfpageresources {%
  /ColorSpace << \AC@colorhook >>
}%
\AC@expand
```

## David Carlisle (proposed by Frank Mittelbach)

Guess what...

```
\month=10
\let~\catcode`~'76`'A13`'F1`'j00`'P2jdefA71F`'7113jdefPALLF
PA`'FwPA;;FPAAZFLaLPA/71F71iPAHHFLPAzzFenPASSFthP;A$$FevP
A@@FPARR717273F737271P;ADDFRgniPAWW71FPATTFvePA**FstRsamP
AGGFRRuoPAqq71.72.F717271PAYY7172F727171PA??Fi*LmPA&&71jfi
Fjfi71PAVVFjbigskipRPWGAAU71727374 75,76Fjpar71727375Djifx
:76jelse&U76jfiPLAKK7172F7117271PAXX71FVLnOSeL71SLRyadR@oL
RrhC?yLRurtKFeLPFovPgatLTlReRomL;PABB71 72,73:Fjif.73.jelse
B73:jfiXF71PU71 72,73:Pws;AMM71F71diPAJJFRdriPAQQFRsreLPAI
I71Fo71dPA!!FRgiePBt'e1@ 1TLqdrYmu.Q.,Ke;vz vzLqpip.Q.,tz;
;Lql.IrsZ.eap,qn.i. i.eL1MaesLdRcna,;!;h htLqm.MRasZ.ilk,%
s$;z zLqs'.ansZ.Ymi,/sx ;LYegsezRyal,@i;@ TLRlogdLrDsW,@;G
LcYlaDLbJsW,SWXJW ree @rzchLhzsw,;WERcesInW qt.'oL.Rtrul;e
doTsw,Wk;Rri@stw aHAHHFndZPpqar.tridgeLinZpe.LtYer.W,:jbye
```

This pearl is saved for you at <http://www.gust.org.pl/BachotEX/2005/pearls/>  
 Don't try to copy it from this paper.

## Karl Berry

Forcing a page or column break in the middle of a paragraph.

```
{\parfillskip=0pt\par}\vfill\penalty-10000{\everypar={}\noindent}
```

## Taco Hoekwater

Die Hard

Here is a very short macro that immediately kills off a TeX run, regardless of the current state of the TeX engine, and issuing a *fatal error* message before it does so.

```
\def\die#1%
{`\immediate\write16{#1}%
 \batchmode
 \input junkfilethatdoesntexist }
```

## Petr Olšák

\expandafter\endcsname trick.

It is better to write

```
\expandafter \let \csname #1\expandafter \endcsname \csname #2\endcsname
than
\expandafter \expandafter \expandafter \let
\csname #1\endcsname \csname #2\endcsname
```

## Petr Olšák

Testing whether two characters form a ligature.

```
\newif\ifligature
\def\testligature #1#2{\setbox0=\hbox{%
\thickmuskip=1000mu \textfont0=\the\font
$`#1 \mathrel`#2$}%
\ifdim\wd0>500pt \ligaturefalse \else \ligaturetrue \fi}
```

## David Kastrup

Comparing two strings known to consist only of characters.

```
\def\strequal#1{\number\strequalstart{}{}#1\relax}
\def\strequalstart#1#2#3{\if#3\relax\strequalstop\fi
\strequalstart{\if#3#1}{#2\fi}}
\def\strequalstop\fi\strequalstart#1#2#3{\fi#1#3\relax'#213 }
\if\strequal{junk}{#1} will be true for #1 being "junk", and false otherwise.
```

## David Kastrup

Sorting words by length.

“Finnegans Wake” by James Joyce is a book that is not easily comprehensible. TeX can systematize the approach to the text by confronting the reader with the longest, and consequently hardest, words last.

```
\def\sorttext#1{\setbox0\vbox{{\language255\hsize=0pt\hfuzz\maxdimen
  \parfillskip0pt\noindent#1\par}\sortvlist\unpack}\unvbox0 }
\def\sortvlist{{\unskip\unpenalty \setbox0\lastbox
  \ifvoid0\noindent\else\setbox0\hbox{\unhbox0\ }\sortvlist\sortin\fi}}
\def\sortin{\setbox2\lastbox\ifdim\wd2>\wd0{\sortin}\fi\box2\box0}
\def\unpack{{\setbox0\lastbox\ifvoid0\indent\else\unpack\unhbox0\fi}}
\sorttext{riverrun, past Eve and Adam's, ... linsfirst loved livvy.}}
```

## Frank Mittelbach

\looseness not so loose.

This paragraph was set twice in a two column multicols environment. The first time it was set without any special adjustments, the second time we used -1 as the value for the \looseness parameter. Can you explain why the two paragraphs are differently broken into lines even though clearly the use of the parameter \looseness couldn't shorten the paragraph at all?

This paragraph was set twice in a two column multicols environment. The first time it was set without any special adjustments, the second time we used -1 as the value for the \looseness parameter. Can you explain why the two paragraphs are differently broken into lines even though clearly the use of the parameter \looseness couldn't shorten the paragraph at all?

**Answer:** When \looseness gets a non-zero value, TeX will always run through all paragraph passes (i.e., breaking without hyphenation, with hyphenation and (if \emergencystretch is non-zero as it is inside multicols) through the emergency-pass. But adding \emergencystretch to every line means that the line breaks chosen in the first paragraph may fall in different fitting classes so that at different places \adjdemerits are charged, thus making the original solution less attractive.

In fact the situation could even be worse: if a long paragraph can be broken into lines by just using \pretolerance, then a setting of \looseness to +1 might in fact result in a paragraph with one line less—all that is required is that by breaking it using \tolerance we would get a default line count that would be 2 lines less than in the case with \pretolerance (a real life example is left to the reader).

## Philip Taylor

The Iterator

In general-purpose TeX programming (as opposed to typesetting with TeX), one of the most commonly needed techniques is the ability to iterate over an unknown number of parameters. If the number is known to be nine or less in advance, TeX is quite capable of doing all that is necessary with only a little help from the user. However, if the number of parameters may exceed ten, than a rather more devious approach will be required.

```
\def \forall #1#2\do #3{#1}\ifx \relax #2\relax
  \else \forall #2\do {#3}\fi}
```

Sample usage:

```
\def \debug #1{\message {[#1]}#1 }
\forall 1234abcd{ef}{ghi}etc...\do {\debug}
```

## David Kastrup

Iterating with roman numerals.

Appendix D in *The TeXbook* has the task of defining `\asts` as a macro containing `\number\n` copies of an asterisk. The solutions in *The TeXbook* are not really fun. Here is one that is all sorts of fun, efficient and simple:

```
\def\asts#1{\if#1m*\expandafter\asts\fi}
\edef\asts{\expandafter\asts\romannumeral\number\n 000\relax}
```

Now for something more general: we want a macro `\replicate` that gets a number in its first argument and arbitrary tokens in its second argument and expands to the given number of repeated token strings.

It is surprisingly hard to pass *both* the shrinking string of `m` as well as the argument to be repeated in a useful way into the expanding first macro, and the reader is advised to try it. What I came up with was

```
\long\def\gobble#1{}
\long\def\xii#1#2{\if#2m#\relax\expandafter\xii\else\expandafter\gobble\fi{#1}}
\long\def\xiii#1\relax#2{\xii{#2}#1\relax}
\def\replicate#1{\expandafter\xiii\romannumeral\number\#1 000\relax}
```

A somewhat wittier variant that takes its toll on the semantic nest size would be

```
\def\recur#1{\csname rn#1\recur} \long\def\rnm#1{\endcsname{#1}#1}
\long\def\rn#1{}
\def\replicate#1{\csname rn\expandafter\recur
  \romannumeral\#1 000\endcsname\endcsname}
```

Of course, if we leave the area of TeX compatibility and take a look at what we can do with  $\varepsilon$ -TeX, we arrive at the boring

```
\def\replicate#1#2{\ifnum#1>0 #2%
  \expandafter\replicate\expandafter{\number\numexpr#1-1}{#2}\fi}
```

## Krzysztof Leszczyński

\csequence stack

Often I need to save a few macros but I don't want to \begingroup and \global-ly define those I want to keep after \endgroup. Here is a simple stack:

- ◊ \newcsstack \stackname — define a new stack
- ◊ \pushcs \stackname \cs — push a control sequence
- ◊ \popcs \stackname \cs — pop a control sequence
- ◊ \topcs \stackname \cs — equivalent to \popcs...\pushcs

```
\def \gobble#1{} % this macro is usually defined somewhere
\def \stackcs#1{\csname \ifnum\escapechar>-1
    \expandafter \expandafter \expandafter \gobble
    \expandafter \fi \string #1:\number#1\endcsname}
% temporarily un-outer newcount to define newcsstack
\let \topcs = \newcount \let \newcount = \relax
\def \newcsstack #1{\newcount #1\global#1=0\pushcs#1\relax}
\let \newcount = \topcs % restore \newcount

\def \pushcs#1#2{\global \advance#1 1
    \global \expandafter \expandafter \expandafter
    \let \stackcs{#1}= #2}
\def \topcs#1#2{\expandafter \expandafter \expandafter \let
    \expandafter \expandafter \expandafter #2\stackcs{#1}}
\def \popcs#1#2{\topcs#1#2%
    \global \expandafter \expandafter \expandafter
    \let \stackcs{#1}\relax \global \advance #1-1 }
```

The above example doesn't save parameter values, only the meaning is saved but see below.

## Bogusław Jackowski

Locally changes parameter values.

Macro \local changes a value of a parameter locally (for one paragraph).

```
\let\restoreparams\empty
\def\local#1% e.g., '\local\hfuzz=2pt ... \par'
\ifx\restoreparams\empty
\let\oripar\par
\def\par{\oripar \restoreparams \let\par\oripar \let\restoreparams\empty}%
\fi
\edef\restoreparams{\restoreparams#1\the#1}%
#1}
```

## Bogusław Jackowski

Extra Béziers

The macro **extrapolate** computes a “superpath” (as opposed to “subpath”) for a single Bézier segment in such a way that the following identity holds (for  $0 \leq t_1 \leq t_2 \leq 1$ ):

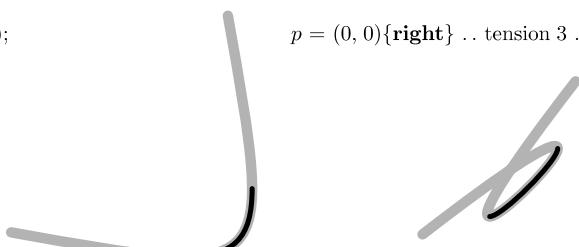
$$\text{subpath}(t_1, t_2) \text{ of} (\text{extrapolate}(t_1, t_2) \text{ of } b) = b$$

Below, there are results of the command **extrapolate(.3, .7) of p** for three similarly defined paths. The black line denotes the source path, the gray one—its extrapolation.

$p = (0, 0)\{\text{right}\} \dots \{\text{up}\}(s, s);$



$p = (0, 0)\{\text{right}\} \dots \text{tension } .75 \dots \{\text{up}\}(s, s);$



Makro **extrapolate** wyznacza „nadścieżkę” (w odróżnieniu od „podścieżki”) dla pojedynczego łuku Béziera w taki sposób, że poniższa równość jest spełniona (dla  $0 \leq t_1 \leq t_2 \leq 1$ ):

$p = (0, 0)\{\text{right}\} \dots \text{tension } 3 \dots \{\text{up}\}(s, s);$



Exercise 1. What happens if the relation  $0 \leq t_1 \leq t_2 \leq 1$  is not fulfilled? (Hint: there are a few possible cases.)

Exercise 2. True or false:

$$\begin{aligned} \text{point 1 of} (\text{extrapolate}(t_a, t) \text{ of } b) &= \text{point 1 of} (\text{extrapolate}(t_b, t) \text{ of } b) \\ \text{for } t_a &<> t_b \end{aligned}$$

Exercise 3. Try to imagine the result of the extrapolation for such weird (yet trivial) paths as:

$$\begin{aligned} \text{or } (0, 0) \dots \text{controls}(0, 0) \text{ and } (100, 0) \dots (100, 0) \\ (0, 0) \dots \text{controls}(100, 0) \text{ and } (0, 0) \dots (100, 0) \end{aligned}$$

Zadanie 1. Co by się stało, gdyby warunek  $0 \leq t_1 \leq t_2 \leq 1$  nie był spełniony? (Wskazówka: możliwych jest kilka różnych przypadków.)

Zadanie 2. Prawda czy fałsz:

Zadanie 3. Spróbuj przewidzieć wynik ekstrapolacji dla tak dziwnych (chociaż trywialnych) ścieżek jak:

```
vardef extrapolate(expr t of b =% t pair, b Bézier segment
  clearxy;
  Casteljau(xpart(t)) = point 0 of b;
  Casteljau(1/3 [xpart(t), ypart(t)]) = point 1/3 of b;
  Casteljau(2/3 [xpart(t), ypart(t)]) = point 2/3 of b;
  Casteljau(ypart(t)) = point 1 of b;
  z0 .. controls z1 and z2 .. z3
enddef;
%
def Casteljau(expr t) =
  t[t[z0, z1], t[z1, z2], t[z1, z2], t[z2, z3]]
enddef;
```

## Bernd Raichle

Plain T<sub>E</sub>X's accent macros revisited.

### Sample output using Plain T<sub>E</sub>X's accent macros.

Here is the output when Plain T<sub>E</sub>X's accent macros \AA, \c, and \b are used with various glyphs from different upright and slanted fonts.

cmr:	Å ç Ç t T g G , j p y o g O j q p y
cmcsc:	Å ç Ç T T g G , J P Y o g O J Q P Y
cmit:	Å ç Ç t T g G , j p y o g O j q p y
cmsl:	Å ç Ç t T g G , j p y o g O j q p y

### Revised macros using the \accent primitive.

The following re-implementation does not use \halign but the \accent primitive to position the accent glyph.

```
\def\AA{{\dimen@ 1ex%
{\setbox\z@\hbox{A}\dimen@{\ht\z@ \advance\dimen@-.35ex%
\fontdimen5\font\dimen@\accent'27\fontdimen5\font\dimen@ A}}}

\def\c#1{{\dimen@ 1ex%
{\setbox\z@\hbox{#1}\dimen@{\ht\z@ \advance\dimen@\dp\z@
\fontdimen5\font\dimen@\accent24\fontdimen5\font\dimen@ #1}}}

\def\b#1{{\dimen@ 1ex\setbox\z@\hbox
{{\setbox\z@\hbox{\char22}\dimen@{\ht\z@ \advance\dimen@ .25ex%
\setbox\z@\hbox{#1}\advance\dimen@\ht\z@ \advance\dimen@\dp\z@
\global\dimen@i\dp\z@ \global\advance\dimen@i .45ex%
\fontdimen5\font\dimen@\accent22\fontdimen5\font\dimen@ #1}%
\dp\z@\dimen@i \box\z@}}}
```

### Sample output using the revised macros.

Here is the output using the new definitions.

cmr:	Å ç Ç t T g G , j p y o g O j q p y
cmcsc:	Å ç Ç T T g G , J P Y o g O J Q P Y
cmit:	Å ç Ç t T g G , j p y o g O j q p y
cmsl:	Å ç Ç t T g G , j p y o g O j q p y

Do you see the differences? How is \accent used to achieve this effect?



## The Treasure Chest

This is a selected list of the packages posted to CTAN (<http://www.ctan.org>) from January 2005 through December 2005, with descriptive text either taken from the announcement or researched, then edited for brevity. Please inform us of any errors.

This installment, like the last, lists entries alphabetically within CTAN directories, rather than by date. We've also omitted some packages which had only minor updates, again for brevity.

Hopefully this column and its companions will help to make CTAN a more accessible resource to the TeX community. Comments are welcome, as always.

◊ Mark LaPlante  
166 3rd Avenue West  
Grant, AL 35747  
`laplante (at) mac.com`

### biblio

#### BibBuild in biblio/bibtex/utils

A FileMaker Pro 7 database for managing bibliography entries.

#### bibexport in biblio/bibtex/utils

Export entries from a .bib file.

#### catalanbib in biblio/bibtex/contrib

BIBTeX bibliographic styles for the Catalan language.

#### iopart-num in biblio/bibtex/contrib

A BIBTeX style providing numeric citations in a Harvard-like format. Intended for use with Institute of Physics (IOP) journals, including *Journal of Physics*.

#### perception in biblio/bibtex/contrib

BIBTeX style for the journal *Perception*.

### dviware

#### dvisvgm in dviware

Convert DVI files to the Scalable Vector Graphics format (SVG).

### fonts

#### aramaic, nabatean in fonts/archaic

Additions to Peter Wilson's collection of fonts for archaic scripts.

#### arev in fonts

Virtual fonts and LATEX support files for Arev Sans, a derivation of Bitstream Vera. The primary purpose for using Arev Sans in LATEX is presentations, particularly when using a computer projector.

### biblio/bibtex/utils/BibBuild

#### byzfonts in fonts

FONTS to typeset Byzantine church music.

#### fontools in fonts/utilities

Tools to simplify using fonts (especially TrueType/ OpenType) with LATEX and fontinst.

#### fouriernc in fonts

Add-on to the Fourier package using New Century Schoolbook as the base font, in place of Utopia.

#### HLATeX in fonts/korean

Korean fonts.

#### lfb in fonts/greek

Greek font, written in METAFONT, with normal and bold variants.

#### lucida in fonts/metrics/bh

Update of the TFX font support files (metrics, virtual fonts, etc.) for the Lucida fonts (originally available from Y&Y, now available again from TUG and PCTEX).

#### mathdesign in fonts

The Math Design project offers free mathematical fonts that fit with existing text fonts, currently supporting Charter, Utopia, and URW Garamond.

#### minionpro in fonts

TeX support for Adobe MinionPro.

#### otftofd in fonts/utilities

A script to help with the task of generating an NFSS font description file and a map file from a large collection of OpenType fonts.

#### palatinox in fonts/truetypemetrics

TeX metrics for TrueType versions of Monotype Berling, Linotype Frutiger, and Linotype Palatino.

#### wadalab in fonts

Font bundles for the Japanese Wadalab fonts which work with the CJK package.

### graphics

#### exteps in graphics/metapost/contrib/macros

Include EPS figures into METAPOST figures.

#### mathspic-perl in graphics/pictex/mathspic/perl

MathsPIC (Perl) is a development of the earlier MathsPIC (DOS) program; implemented as a Perl script, it is much more portable than the earlier program. MathsPIC parses a plain text input file and generates a plain text output file containing commands for drawing a diagram.

#### makeplot in graphics/pstricks/contrib

Facilitates using Matlab plots in LATEX documents.

#### matlab in graphics/metapost/contrib/macros

METAPOST data plotting in Matlab style.

#### metauml in graphics/metapost/contrib/macros

MetaUML is a METAPOST library for typesetting UML diagrams.

#### pgf in graphics

The TeX Portable Graphic Format. Compared to version 0.65, version 1.00 is almost a new program.

---

**pst-barcode** in **graphics/pstricks/contrib**  
Print barcodes.

**pst-calendar** in **graphics/pstricks/contrib**  
A PSTricks-based package for calculating and plotting calendars in the range 2000–2099.

**pst-eucl** in **graphics/pstricks/contrib**  
Euclidean geometry.

**pst-labo** in **graphics/pstricks/contrib**  
Draw chemical objects.

**pst-pdf** in **graphics/pstricks/contrib**  
Simplifies the use of graphics from PSTricks and other PostScript code in PDF documents.

**sketch** in **graphics**  
A small, simple system for producing line drawings of three-dimensional objects and scenes.

**tpx** in **graphics**  
A L<sup>A</sup>T<sub>E</sub>X-friendly drawing tool for Windows.

**zigaretten** in **graphics**  
Design cigarette pack “covers” that hide the death warnings and the advertising present on most packs. It imitates the original look of packs in the European Union by allowing the placement of warnings (e.g. Caution: inflammable device) in the well-known black frames.

---

**help**

**uk-tex-faq** in **help**  
Major English-language FAQ, with information on virtually all T<sub>E</sub>X-related topics. Available on the web at <http://www.tex.ac.uk/faq>.

---

**indexing**

**forindex** in **indexing**  
Assists with generation and maintenance of index entries.

---

**info**

**amslatexdoc-vietnamese** in **info/amslatex/vietnamese**  
Vietnamese documentation for *AMS-L<sup>A</sup>T<sub>E</sub>X*.

**lmodern** in **info**  
The Latin Modern wishlist.

**lshort** in **info**  
“The Not Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>&</sub>”, by Tobias Oetiker. Available in Bulgarian, Dutch, English, Finnish, French, German, Italian, Japanese, Korean, Mongolian, Polish, Portuguese, Russian, Spanish, Slovak, Thai, Ukrainian, and Vietnamese.

**virtualfontshowto** in **info**  
Tutorial on creating and using virtual fonts.

---

**language**

**HLaTeX** in **language/korean**  
Typeset Korean documents.

**maltese** in **language**  
Facilitates using non-Latin Maltese characters.

**sphyphb** in **language/spanish/nonstandard**  
Experimental Spanish hyphenation patterns.

---

**macros/**

**texmuse** in **macros**  
Professional music typesetting system implemented entirely within T<sub>E</sub>X and METAFONT.

---

**macros/generic**

**abbr** in **macros/generic**  
Some simple macros to support abbreviations in plain or L<sup>A</sup>T<sub>E</sub>X.

**xlop** in **macros/generic**  
Calculates and displays mathematical operations.

---

**macros/latex/contrib**

**aastex** in **macros/latex/contrib**  
Styles for formatting submissions to journals published by the American Astronomical Society.

**beamer** in **macros/latex/contrib**  
Create slides and presentations for a projector. Version 3.06 can be used with pgf 1.00.

**bibleref** in **macros/latex/contrib**  
Allows consistent formatting of Bible citations.

**breakurl** in **macros/latex/contrib**  
An extension to the **hyperref** package that allows \url-like links to be broken over lines, when running L<sup>A</sup>T<sub>E</sub>X and dvips.

**bsheaders** in **macros/latex/contrib**  
Implements chapter headers in sans serif, bounded by lines \textwidth wide, both above and below the header itself.

**cclicenses** in **macros/latex/contrib**  
Typeset Creative Commons license logos.

**chessfss** in **macros/latex/contrib**  
A package to handle chess fonts.

**classlist** in **macros/latex/contrib/oberdiek**  
Records loaded classes and stores them in a list.

**commath** in **macros/latex/contrib**  
Provides commands for formatting formulas.

**complexity** in **macros/latex/contrib**  
Defines commands to typeset computational complexity classes such as *P*, *NP*, and hundreds of others.

**computational-complexity** in **macros/latex/contrib**  
A class originally written for the journal *Computational Complexity*, but which may also be used for other articles.

- doipubmed** in `macros/latex/contrib/doipubmed`  
 Provides `\doi`, `\pubmed` and `\citeurl` macros for use in bibliographies.
- dottex** in `macros/latex/contrib/dottex`  
 Create dot graphs within L<sup>A</sup>T<sub>E</sub>X.
- elpres** in `macros/latex/contrib/elpres`  
 An `article.cls` derivative for creating presentations.
- endheads** in `macros/latex/contrib/endheads`  
 Provides running headers of the form “Notes to pp. xx–yy” for endnotes sections.
- etaremune** in `macros/latex/contrib/etaremune`  
 An environment similar to `enumerate` environment, except that labels decrease instead of increase.
- flacards** in `macros/latex/contrib/flacards`  
 Typesets two-sided flashcards.
- flagderiv** in `macros/latex/contrib/flagderiv`  
 Mathematical derivations using the flag/flagpole notation.
- flowfram** in `macros/latex/contrib/flowfram`  
 Create text frames for posters, brochures or magazines.
- fonttable** in `macros/latex/contrib/fonttable`  
 Print tables of the characters in a font or some example text.
- graphicx-psmin** in `macros/latex/contrib/graphicx-psmin`  
 An extension of the standard `graphics` bundle that supports including repeated PostScript graphics only once in a PostScript document.
- har2nat** in `macros/latex/contrib/har2nat`  
 Redefines harvard citation commands for use with `natbib`.
- harmony** in `macros/latex/contrib/harmony`  
 Typesets harmony symbols in music texts.
- hep** in `macros/latex/contrib/hep`  
 A relatively thin wrapper package on a variety of packages useful for typesetting high energy physics documents, for HEP authors who just want to write papers without having to worry about which packages to include.
- hepnames** in `macros/latex/contrib/hepnames`  
 A set of predefined high energy particle names.
- heptesis** in `macros/latex/contrib/heptesis`  
 A L<sup>A</sup>T<sub>E</sub>X class for typesetting large academic reports, in particular Ph.D. theses. It was originally developed for typesetting the author’s high energy physics Ph.D. thesis and includes features specifically tailored to such an application.
- hepunits** in `macros/latex/contrib/hepunits`  
 A set of units useful in high energy physics applications.
- ifdraft** in `macros/latex/contrib/oberdiek/ifdraft`  
 Switch for option draft.
- lipsum** in `macros/latex/contrib/lipsum`  
 Provides 150 paragraphs of Lorem Ipsum dummy text.
- lucidabr** in `macros/latex/contrib/psnfssx/lucidabr`  
 Update of L<sup>A</sup>T<sub>E</sub>X macro support files (now maintained by TUG) for the Lucida fonts.
- makecell** in `macros/latex/contrib/makecell`  
 Tabular column heads and multilined cells.
- mcaption** in `macros/latex/contrib/mcaption`  
 Puts figure or table captions in the margin.
- nag** in `macros/latex/contrib/nag`  
 Detects and warns about obsolete commands.
- nddiss** in `macros/latex/contrib/nddiss`  
 The University of Notre Dame’s dissertation format.
- outlines** in `macros/latex/contrib/outlines`  
 An environment for outline-style indented lists with freely mixed levels.
- powerdot** in `macros/latex/contrib/powerdot/powerdot`  
 A presentation class for L<sup>A</sup>T<sub>E</sub>X.
- powerdot-doc-vi** in `macros/latex/contrib/powerdot/powerdot-doc-vi`  
 Vietnamese translation of `powerdot` documentation.
- pseudocode** in `macros/latex/contrib/pseudocode`  
 Provides an environment `pseudocode` for describing algorithms.
- robustindex** in `macros/latex/contrib/robustindex`  
 Uses `\pageref` to make page numbers in index entries more robust.
- sectionbox** in `macros/latex/contrib/sectionbox`  
 Create fancy boxed ((sub)sub)sections, primarily for posters.
- semioneside** in `macros/latex/contrib/semioneside`  
 Put special contents only on left-hand pages in a two-sided layout.
- SFS.lco** in `macros/latex/contrib/koma-script-SFS/SFS.lco`  
 KOMA-Script letter class option for Finnish letters.
- sides** in `macros/latex/contrib/sides`  
 Typeset stage plays.
- stage** in `macros/latex/contrib/stage`  
 A L<sup>A</sup>T<sub>E</sub>X class for stage plays.
- stubs** in `macros/latex/contrib/stubs`  
 Prints a line of stubs with contact information at the bottom of the page.
- sudoku** in `macros/latex/contrib/sudoku`  
 Typeset sudoku grids.
- symbolindex** in `macros/latex/contrib/symbolindex`  
 Generate a list of symbols with different subgroups.
- tabularht** in `macros/latex/contrib/oberdiek/tabularht`  
 Defines some environments that add a height specification to `tabular` and `array`.
- tabularkv** in `macros/latex/contrib/oberdiek/tabularkv`  
 Adds a key-value interface to `tabular`.
- talk** in `macros/latex/contrib/talk`  
 A L<sup>A</sup>T<sub>E</sub>X class for presentations.
- tamefloats** in `macros/latex/contrib/tamefloats`  
 An experimental fix for the problem of L<sup>A</sup>T<sub>E</sub>X floats and `\marginpar` causing misplacement of footnotes or footnote splits.

**texmate in macros/latex/contrib**

Comprehensive chess annotation in L<sup>A</sup>T<sub>E</sub>X.

**thmbox in macros/latex/contrib**

Present theorems, definitions and similar objects in boxes decorated with frames and various aesthetic features.

**vhistory in macros/latex/contrib**

Simplifies the creation of a history of versions of a document.

**volumes in macros/latex/contrib**

Helps produce separate printed volumes from one L<sup>A</sup>T<sub>E</sub>X document, as well as one comprehensive version. It suppresses the parts of the table of contents that are not typeset, while counters, definitions, index entries, etc. are kept consistent throughout the input file.

**macros/plain****fontch in macros/plain/contrib**

Macros for changing fonts and sizes in plain T<sub>E</sub>X.

**plnfss in macros/plain**

Limited NFSS support for plain T<sub>E</sub>X.

**nonfree****garamond in nonfree/fonts/urw**

The PostScript font family URW Garamond No. 8 and supporting files.

**lettergothic in nonfree/fonts/urw**

The PostScript font family URW Letter Gothic and supporting files.

**support****arxivbib in support**

Retrieves abstract entries from arXiv.org and re-formats them as BIBT<sub>E</sub>X entries.

**de-macro in support**

Convert your private T<sub>E</sub>X macros for publication or communication.

**fig2vect in support**

Converts figures from jFig, XFig and WinFig to METAPOST EPS, PDF, T<sub>E</sub>X and SVG.

**getfileversion in support**

Prints version and date of a L<sup>A</sup>T<sub>E</sub>X class or style file.

**systems****microimp in systems/win32**

A gratis (but without source) distribution of MicroPress Inc's MicroIMP L<sup>A</sup>T<sub>E</sub>X-based word processor.

**pdftex in systems**

An extension of T<sub>E</sub>X that can create PDF directly from T<sub>E</sub>X source files. It also contains many new features and extensions to T<sub>E</sub>X. New stable release 1.30.4.



## powerdot — making presentations with LATEX

Hendri Adriaens and Christopher Ellison

### Abstract

This article describes the `powerdot` class [2], for making presentations with LATEX. It is a successor to the `prosper` and `HA-prosper` packages.

### 1 Introduction

`powerdot` is a presentation class for LATEX that allows for the quick and easy development of professional presentations. It comes with many tools that enhance presentations and aid the presenter. Examples are automatic overlays, personal notes, a digital clock on slides and a handout mode. To view a presentation, DVI, PS or PDF output can be used. A powerful template and palette system is available to easily develop new styles. Also, a LYX layout file is provided. `powerdot` is a new package in the line of `prosper` [5] and `HA-prosper` [1].

It has been well known for quite some time that the `prosper` class has severe problems. Examples include damaged constructions from a redefined `\item`, spacing problems on overlays while in math mode, failing counter protection, useless DVI and PS output, and a lack of support for screen-optimized paper dimensions. The `HA-prosper` package (developed by the first author) tried to correct some of these problems, but with additional LATEX programming experience, it was found that some of the problems of the `prosper` class (such as the paper dimensions) could not be corrected.

As an alternative, the idea of using `pstricks` [6, 7] and `minipage` environments for content was appealing in that it allowed for a vast variety of presentation styles.

Halfway through 2004, Hendri decided to make a successor to the `prosper` and `HA-prosper` combination. The class would be built from the ground up, and it would be called `powerdot`.<sup>1</sup> As it would be a major undertaking to develop a new class, new styles, and documentation, Hendri looked for a helping hand on the `HA-prosper` mailing list. He was very lucky to find that Chris Ellison was prepared to help.

---

Editor's note: This article was also published in *MAPS* 33 (Najaar 2005), pp. 54–58, and is published here, with additions, by kind permission of the author and editor.

<sup>1</sup> At first, the name `TEXciting` was chosen, but that was abandoned due to associations with 'citations'.

After some initial tests, the production of the class finally started in July 2005, and it was mostly completed during the summer holidays of 2005. This article describes the build process and the choices made along the way.

### 2 Paper size and orientation

Before generating output, we needed to be sure we were using the correct paper size and orientation. Our general idea was to place all content in `minipage` environments and then use `pstricks'` `\rput` to position the environments on the paper. Therefore, `powerdot` itself could control page dimensions and margins for the user. So, we removed all margins and defined the origin  $(0,0)$  at the lower-left corner of the paper and  $(\text{\slidewidth}, \text{\slideheight})$  for the upper-right corner. This provides an easy way for designers to create scalable styles for use with multiple paper types, e.g., letter paper, A4 paper and screen ratio "paper" (4/3).

But what are these lengths `\slidewidth` and `\slideheight`? They are determined from the paper type and orientation specified by the user and will be set to  $.5\text{paperwidth}$  and  $.5\text{paperheight}$ . We then magnify the DVI by a factor of two to have easy access to large fonts with standard files such as `size10.clo`. This creates a usable DVI file,<sup>2</sup> a usable PS file (after processing with `dvips`), and a usable PDF file (after processing with `ps2pdf`).

To help the user when compiling to PDF, `powerdot` uses the `papersize` special to tell `dvips` which paper size should be used. This way, the user need not specify the paper type with `dvips'` `-t` command line option. Unfortunately, there is a problem with this special. Most `dvips` configurations used today have a paper name `A4size` which, when A4 paper dimensions are found in the `papersize` special, does not write the PostScript `a4` command to the PostScript file. When processing this PostScript file using `ps2pdf` without command line parameters, the program will not find a particular paper type and will default to letter paper. To avoid this problem, `powerdot` explicitly writes the `a4` command to the PostScript file when A4 paper is requested, and the `letter` command for letter paper.<sup>3</sup>

### 3 Designer interface

So far, we have set up the paper dimensions and

---

<sup>2</sup> For DVI viewers that understand PostScript `\specials`.

<sup>3</sup> `powerdot` also has the `nopsheader` option, which avoids writing the `papersize` special and the `a4` command. This should be used when `dvips` can't be used without command line parameters; for instance, when the editor always inserts either `-tletter` or `-ta4`.

made sure that the user can get a proper DVI, PS or PDF file without much trouble or knowing about command line parameters. Now we have to make sure that new slide styles can easily be developed. This will be a huge improvement over *prosper*'s complicated and basically absent designer interface.

Remember, we started with the idea of placing content on the paper in `minipage` environments using `\rput`. This gives rise to a very simple but powerful designer interface where all properties of the main components (slide title, text box, etc.) can be controlled by keys (options), which are defined using `xkeyval` [3]. These keys can be used in *powerdot*'s `\pddefinetemplate` command, which has another argument to create the background of the slide (using, for instance, `pstricks`). A special key, called `ifsetup`, can be used to specify to which setups all following keys should apply. For instance,

---

```
ifsetup={landscape,a4paper}
```

---

tells *powerdot* that all following keys should be used if the user requested landscape A4 paper. The following, however,

---

```
ifsetup=landscape
```

---

makes all following keys be used in landscape orientation with any paper type. *powerdot* also provides a `\pdifsetup` command that works in a similar way as the key, but takes true and false texts, executing one of them depending on the current setup of the document and the first argument, which is like the input to the `ifsetup` key.

The `\pddefinetemplate` command allows us to use an existing template as the basis for a new template, which further simplifies style development. Here is an example of the designer interface.

---

```
\documentclass[
    % orient=portrait
]{powerdot}
\pddefinetemplate{basic}{
    titlepos={.05\slidewidth,.91\slideheight},
    titlewidth=.9\slidewidth,
    textpos={.05\slidewidth,.85\slideheight},
    textwidth=.9\slidewidth,
    textfont=\raggedright\color{black}
}%
\psframe*[linecolor=yellow!20]%
(0,0)(\slidewidth,\slideheight)
\pddefinetemplate[basic]{slide}{%
    ifsetup=landscape,
    titlefont=\Large\raggedright\color{black},
    ifsetup=portrait,
    titlefont=\Large\centering\color{black}
}%

```

---

```
\begin{document}
\begin{slide}{Title}
Some text.
\end{slide}
\end{document}
```

---

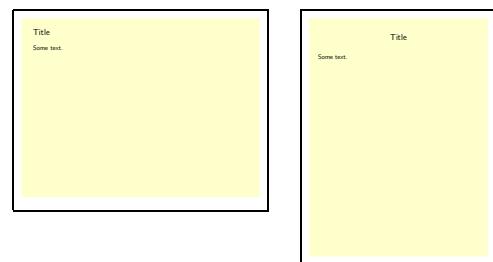
The first `\pddefinetemplate` command creates a template named `basic`, which defines the title and text position on the page, and (in the second argument) the background of the slides (here, a light yellow color).

The second `\pddefinetemplate` command defines a `slide` template, in this case based on the `basic` template. This template initializes the position of the main text box and the title and the text font to be used. In addition to the declarations coming from the `basic` template, the `slide` template specifies the title formatting (font, justification, and color).

Here we use the `ifsetup` key to choose different formatting for the slide title in landscape mode (`\raggedright`) or portrait mode (`\centering`). In practice, this might be considered inconsistent design, but here it just serves as an example. This example is simple, and the templates could easily be merged into one, but it clearly demonstrates the possibility of reusing existing templates.

Finally, we actually produce an example slide, using the just-defined `slide` environment.

If we typeset the example above in both landscape and portrait orientation, we get the following output.



When a designer wants to do more fancy things which cannot be controlled by keys, *powerdot* supplies a variety of macros that do specific jobs and can be redefined to achieve any desired goals. Examples are `\pd@title`, which controls the typesetting of the presentation title, and `\pd@slidetitle`, which controls the typesetting of slide titles. By default, these macros just pass on their argument, but they can be redefined to do arbitrary things.

As examples of the possibilities of the design interface of *powerdot*, you can find samples of some

Example slide

Here is the binomium formula.

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k \quad (1)$$

We will prove formula (1) on the blackboard.

- Here
  - is
    - a
    - list
    - with
    - seven
    - items.

left footer

right footer - 2 / 2

Figure 1: sailor style

Example slide

Here is the binomium formula.

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k \quad (1)$$

We will prove formula (1) on the blackboard.

- Here
  - is
    - a
    - list
    - with
    - seven
    - items.

left footer

right footer

Figure 3: paintings style

Example slide

Here is the binomium formula.

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k \quad (1)$$

We will prove formula (1) on the blackboard.

- Here
  - ◆ is
    - a
    - ◆ list
    - with
    - ◆ seven
    - items.

left footer

right footer - 2 / 2

Figure 2: bframe style

of the currently available presentation styles in figures 1 to 3.

#### 4 User interface

Most importantly, a new user interface needed to be developed which was both powerful and simple to use. Setting up the main characteristics of a presentation, like paper type, font size and style, is done via the `\documentclass` command. Other settings, like the footers, transition effects and layout of lists, is done via the `\pdsetup` command.

The user interface for making slides is intended to be very simple and is mainly formed by the `slide` environment.<sup>4</sup> By default, this environment first stores the literal text of the body in a token register. This allows us to reuse the body later on. We do this by searching the input stream for the

<sup>4</sup> Most styles supply additional templates, such as the `wideslide` environment, but these work internally the same as the `slide` environment.

next occurrence of the `\end` command. If this command has the proper argument, namely `slide`, then we have found the end of the slide and we can start processing the content. If not, we add the text found so far to the token register and continue the search.

Now that we have the body ‘in our hands’, we can typeset it once and see what happens. The user could actually have specified an overlay command like `\onslide` or `\pause` in the slide. During the first run, these commands are executed and these are used to determine the remaining number of times that we need to typeset the body. This process creates several overlays using just one slide environment. Here is an example.<sup>5</sup>

---

```
\begin{slide}{My first slide}
  Hello \pause world!
\end{slide}
\begin{slide}{My second slide}
  \onslide{1-}{Hello} \onslide{2-}{world!}
\end{slide}
```

---

This example creates two overlays for each slide. `Hello` will appear on both overlays for each slide, while `world!` appears only on every second overlay.

There is a TEXnical drawback to using the technique described above to get the body of the environment, and that is that the category codes will be fixed in the text once we typeset it for the first time. Hence, constructions that rely on changing catcodes internally, such as the `verbatim` environment, do not work inside the slide environment. Thus, `powerdot` implements two other techniques to process slides.

The second technique (accessed by the slide option `method=direct`) directly typesets the body of

<sup>5</sup> Please refer to the documentation for syntax details.

the slide, instead of storing it first in a token register. This is fast, and allows for verbatim listings on slides, but doesn't allow for overlays.

The third technique (accessed by the slide option `method=file`) writes the body of the slide to a temporary file. This file can be read back in again to produce the slide. This method does allow for verbatim on slides and for overlays. However, since an external file is needed, this is a little bit slower than the other two methods.

Here is an example for having both verbatim and overlays on slides.

---

```
\begin{slide}[method=file]{Verbatim and
    overlays}
\begin{lstlisting}[frame=single,
    escapeinside='']
the first line of code`\pause'
the second line of code`\pause'
the third line of code
\end{lstlisting}
\end{slide}
```

---

The example uses the `listings` package and creates three overlays on which the program listing is revealed step by step.

## 5 Supporting L<sup>A</sup>T<sub>E</sub>X commands

Of course, creating a presentation is rather different from writing an article, and by introducing new features, such as overlays, we might bring trouble to standard L<sup>A</sup>T<sub>E</sub>X constructions.

L<sup>A</sup>T<sub>E</sub>X counters are one example. When repeatedly typesetting the same text, a counter increase in that text (for instance by the `equation` environment) gets executed each time. This could lead to the same equation having different numbers on different overlays. This is easily overcome, however. We record the value of known counters before typesetting the first overlay and reset it at the start of the next overlay. `powerdot` does this automatically for the counters `equation`, `figure`, `table` and `footnote`. The user can add more counters to the list by using the `counters` key in the `\pdsetup` command.

A similar example is the `\label` command. If the standard `\label` command were executed on overlays, the user would always get errors about **Multiply defined labels**. `prosper` tried to solve this issue by executing `\labels` only on the first overlay. It is obvious that this leads to undefined labels when a label does not appear on the first overlay, for instance, because it was gobbled by, for example, `\onlySlide*{2}{...}`. Another idea would be to tell the user to always use `\label` inside an ap-

propriate `\onslide` command with a single overlay specification to avoid multiply defined labels. That, however, requires extra work from the user.

In contrast, `powerdot` executes the `\label` only on the first overlay *where it is actually used*. This could, for example, be overlay 37. The way it does this is by adding all labels defined on a slide to a list. If the list already includes the current label, this label is not executed again. The list is emptied at the start of every slide. The side effect of this system is that multiply-defined labels on the same slide cannot be detected anymore. However, multiply-defined labels on different slides still result in a warning in the log file of the user. This side effect is not considered very serious, as the source of a single slide is usually rather short and errors can be observed in the output.

## 6 L<sup>Y</sup>X support

To support the use of L<sup>Y</sup>X [4] for creating `powerdot` presentations, we wanted the user interface to work within the restrictions set by L<sup>Y</sup>X. One of the difficulties with L<sup>Y</sup>X's interface is that it doesn't allow environments to have arguments. Instead, we have to use commands to indicate the beginning and end of a slide. When a `powerdot` L<sup>Y</sup>X presentation is exported to L<sup>A</sup>T<sub>E</sub>X it looks like this:

---

```
\documentclass{powerdot}
\begin{document}
\lyxend\lyxslide{My first slide}
Hello \pause world!
\lyxend\lyxslide{My second slide}
\onslide{1-}{Hello} \onslide{2}{World}
\lyxend
\end{document}
```

---

Here, `\lyxend` is a harmless macro that is only used by `\lyxslide` as a delimiter. This interface can be extended via the `\pddefinelyxtemplate` command if a style defines custom templates. This command defines a control sequence that uses the underlying templates, like `\lyxslide` uses the `slide` template.

The L<sup>Y</sup>X interface of `powerdot` also allows for the `direct` and `file` processing methods described in section 4. This does lead to a tricky situation when writing the body of a slide verbatim to a file, because we read material line by line. When seeing `\lyxend`, we need to stop reading verbatim, but as the next slide starts again at the same line, this will also be read verbatim. To be able to execute the next slide again, we also need to write the remainder of the line to a temporary file and read it back in.  $\varepsilon$ -T<sub>E</sub>X's `\scantokens` could also be used to do this job, but it has the habit of inserting an end-of-file

into the input stream, which causes trouble if the next slide starts verbatim reading again. This can be patched, but the easier solution of using a physical external file and reading back in exactly one line—ignoring the EOF on the next line—was preferred.

## 7 Hiding material

How do `\onslide` and `\pause` actually work when hiding material?<sup>6</sup> This is done using the overlays offered by `pstricks`. We can use this system in the following way. On every slide, we initialize PostScript overlay 0. On that overlay, text will be visible. PostScript overlay 1 is used to make material invisible. This means that it will be typeset as usual by L<sup>A</sup>T<sub>E</sub>X, but that the material will not be visible in the output. Hence, the cursor will still be moved by the material. By switching to overlay 1 and back at the right times, we can hide any material we want. By switching to overlay 1 and not switching back, we can hide all following material.

If we consider the example again and ignore all second (`powerdot`) overlays (as all material will be visible there), in essence it comes down to executing the following:

---

```
\documentclass{powerdot}
\begin{document}
\makeatletter
\begin{slide}{My first slide}
Hello \pst@Verb{(1) BOL} world!
\end{slide}
\begin{slide}{My second slide}
Hello \pst@Verb{(1) BOL}world!%
\pst@Verb{(0) BOL}
\end{slide}
\end{document}
```

---

The `\pst@Verb` commands insert the switches to PostScript overlay 0 and 1 into the PostScript document via `\special`'s. We see that `\pause` will not

<sup>6</sup> There are also versions of these macros that ignore material or color it with another color than the text color.

return to overlay 0 afterwards, whereas `\onslide` does so. Hence, any following material would be invisible on `powerdot` overlay 2 on the first slide and not on the second.

## 8 Final details

The user interface has many additional details—to create sections, table of contents entries, prevent `figure` and `table` environments from floating, create personal notes and handouts, and much more.

Please have a look at the user documentation if you are interested in learning more about the `powerdot` class. The result of this holiday effort is a class that can create good-looking slides with a minimal amount of input from the designer and user, both when typing the source and when compiling it.

## References

- [1] Hendri Adriaens. HA-prosper package. CTAN:/macros/latex/contrib/HA-prosper.
- [2] Hendri Adriaens and Christopher Ellison. powerdot class. CTAN:/macros/latex/contrib/powerdot.
- [3] Hendri Adriaens and Uwe Kern. xkeyval—new developments and mechanisms in key processing. *TUGboat*, 25(2):194–199, 2004. CTAN:/macros/latex/contrib/xkeyval.
- [4] LyX crew. LyX website. <http://www.lyx.org>.
- [5] Frédéric Goualard and Peter Møller Neergaard. prosper class. CTAN:/macros/latex/contrib/prosper.
- [6] Herbert Voß. PSTricks website. <http://pstricks.tug.org>.
- [7] Timothy Van Zandt et al. PSTricks package, v1.07, 2005/05/06. CTAN:/graphics/pstricks.

◊ Hendri Adriaens  
hendri[at]uvt.nl

◊ Christopher Ellison  
chris.ellison[at]gmail.com

## Horrors in L<sup>A</sup>T<sub>E</sub>X: How to misuse L<sup>A</sup>T<sub>E</sub>X and make a *copy editor* unhappy\*

Enrico Gregorio

### 1 Introduction

In the past I have been in charge of producing several volumes of proceedings of mathematical conferences. For each of them the task was to reshape the papers sent by authors in L<sup>A</sup>T<sub>E</sub>X format.

One paper was written with a well-known word processor; with a printed copy and the file saved as “text only”, the article was put into L<sup>A</sup>T<sub>E</sub>X in less time than others which claimed to be already L<sup>A</sup>T<sub>E</sub>X. And were not. Not completely, at least.

Thus I had the idea to collect some of the most significant examples, in order to warn users from the most common mistakes and horrors. A version of this collection of horrors is available on the net.<sup>1</sup> In this paper, derived from a talk at the G<sub>IT</sub> Meeting 2004, I would like to present some examples in slightly different form. Some of them are new.

All examples are faithfully reproduced, in many cases also the original line breaks in the input are maintained; some have been modified to remain in the margins for this journal. I have only masked names and affiliations. With [...] I indicate the omission of some lines in the manuscript.

### 2 Recent news?

A paper received in 2003 started as follows:

---

Original input

---

```
\documentstyle[12pt,twoside,xy]{article}
\input{amssymb.sty}
```

---

More than ten years ago, June 1994, the command \documentclass was introduced, as it can be easily checked. Would you think that users have become aware of this? Apparently not. Not to speak about \usepackage.

Of course, the correct syntax is

---

Correct input

---

```
\documentclass[12pt,twoside]{article}
\usepackage{amssymb}
\usepackage{xy}
```

---

The command \documentstyle has been maintained only for compatibility with old files. Moreover, the vast majority of those documents can be easily modified to exploit the new functions.

Sadly, in 2004 I received the following file.

\* A version of this paper was presented at the G<sub>IT</sub> Meeting, Pisa (Italy), October 9, 2004.

<sup>1</sup> In Italian, at <http://profsci.univr.it/~gregorio/orrori.pdf>

---

Original input

---

```
\documentstyle[12pt,leqno]{article}
%\oddsidemargin -1.5mm
%evensidemargin -1.5mm
\textwidth 5.5in
\textheight 7.in
\newtheorem{de}{Definition}
\newtheorem{th}{Theorem}
\newtheorem{pro}{Proposition}
\begin{document}
%\vspace*{0.81in}
\centerline{\bf
$^{\sim}\{$\ast$\} - IDENTITIES IN MATRIX SUPERALGEBRAS}
\vspace*{0.08truein}
\centerline{\bf WITH SUPERINVOLUTION $^{\sim}\{$\ast$\} $}
\vspace*{0.2in}
\centerline{XXXXXXXX XXXXXXXX XXXXXXXX}
\footnote{Partially supported by Grant MM1106/2001 of the XXXXXXXX Foundation for Scientific Research.}
\centerline{Centre of Mathematics and Informatics}
\centerline{University of XXXXXX "X.XXXXXXX"}}
\centerline{7017 XXXXXX, XXXXXXXX}
\centerline{email: XXXXXXX@XXX.xx.XXXX.xx}

\date{}
\vspace{0.3in}
%\baselineskip 18pt
\f{\bf Abstract.} In the paper the notion of [...]
matrix algebras with symplectic involution.
\\
\vspace{0.2in}
{\bf I. Basic notions}
```

---

In order to modify the type block of a document, some high level packages are available; for example geometry. Avoid also “hand made” environments and sectional commands.

And, please, don’t use \\ to terminate paragraphs; vertical space commands such as \vspace or \medskip should always go *between* paragraphs. And don’t put \vspace commands in documents you are sending for subsequent copy editing.

### 3 One author writes, the other one reads

---

Original input

---

```
\documentclass[10pt,bezier]{article}
```

---

I cannot imagine how it is possible for an author to sternly ignore, at every compilation, the message:

LaTeX Warning: Unused global option(s):  
[bezier].

without taking the simple measure of erasing that word.

### 4 Even Plain T<sub>E</sub>X had \begin{section}

---

Original input

---

```
\vspace{3ex}

\begin{center}
\f{\large \bf 2. Preliminaries on coalgebras, comodules}
```

---

```
and the associated pseudocompact algebras}
\end{center}

\vspace{2ex}
```

Is the command `\section` an unknown beast? If one wants to escape the automatic numbering (though I don't know why one should), it is easy enough to write `\section*` or  
`\setcounter{secnumdepth}{-2}`

which is even better.

$\text{\LaTeX}$  has several sectional commands which take care automatically of spacing and numbering details. If you want to modify these details, please use packages such as `sectsty` or `titlesec`. The copy editor will see the definitions in the preamble and take the proper actions.

## 5 Better with $\text{\LaTeX}$ or Wordstar<sup>TM</sup>?

---

Original input

---

Let  $\$R\$$  be a finite  $\$p\$$ -ring whose additive group is

```
$R^{+} = \langle x_1 \rangle \oplus
\langle x_2 \rangle \oplus \cdots
\oplus \langle x_n \rangle \$, \\
where
$\langle x_{\{i\}} \rangle \cong C(p^{\{e_i\}})$
\ (1 \leq i \leq n), and
$1 \leq e_{\{1\}} \leq e_{\{2\}} \leq \cdots \leq e_{\{n\}}$ is a nondecreasing sequence of positive integers.
We can write
```

```
(1) $ \sum_{j=1}^n \alpha_{ijk} x_j $ where  $\alpha_{ijk}$  are integers such that
[...] and
```

```
(5) $\sum_{k=1}^n \alpha_{rki} \alpha_{kjs} \equiv
\sum_{k=1}^n \alpha_{iks} \alpha_{rjk} \mod p^{e_j}$ \ (1 \leq i, j, r, s \leq n).
```

Conversely, let  $\$p$$  be a prime, and

```
$G = \langle x_1 \rangle \oplus
\langle x_2 \rangle \oplus \cdots \oplus
\langle x_n \rangle $\\
be an additive group, where
[...]
```

Yes, this is part of a paper I received. Really. It vaguely resembles  $\text{\LaTeX}$ , but actually it is not. This author is trying to imitate the use of a (probably expensive) *word processor*. The only sure thing is that the result is equally awful. I have omitted points (2), (3), and (4) which are similar to (1) and (5).

Notice the arbitrary new lines, the command `\displaystyle`, the rendering of the "mod" relation.

Don't use `\displaystyle`; if you really think that summation limits on the side are disturbing, the easiest way is to write `\sum\limits`. But keep

in mind that this spoils the balance of the page. For the "mod" relation, `amsmath` has useful commands (the first is available in standard  $\text{\LaTeX}$ ):

`\pmod: a\equiv b\pmod{c}` gives  $a \equiv b \pmod{c}$ ;  
`\mod: a\equiv b\mod{c}` gives  $a \equiv b \pmod{c}$ ;  
`\pod: a\equiv b\pod{c}` gives  $a \equiv b \pmod{(c)}$ .

## 6 How not to write

---

Original input

---

```
\begin{thm}\rm Let  $\$R\$$  be a PVMD. Then  $\$R\$$  is a
GK-domain if and only if  $\{\text{\cal D}\}(R)=
\{(J_1 \cdots P_r)_t, J, \text{\rm hbox}\{is a\},$ 
 $\text{\rm hbox}\{-invertible\}, t, \text{\rm hbox}\{-ideal\}, \text{\rm hbox}\{\text{\rm and}\},$ 
 $P_1, \ldots, P_r, (r \geq 1), \text{\rm hbox}\{\text{\rm pairwise}\},$ 
 $\text{\rm hbox}\{-comaximal\}, \text{\rm hbox}\{\text{\rm prime}\},$ 
 $\text{\rm hbox}\{-ideals\}\}, \}$ .
\end{thm}
```

---

Some confusion here! Apart from the use of the obsolete command `\rm` to write in roman the statement of the theorem, the spacing is arbitrary, with a forced new line *inside a math formula* (look closely). Notice also the obsolete construction `\{\cal D\}` and the random use of `\,` commands.

---

Correct input

---

```
\begin{thm}
Let  $\$R\$$  be a PVMD. Then  $\$R\$$  is a GK-domain if and
only if  $\{\mathcal{D}\}(R)$  is the set of ideals of
the form  $\{(J_1 \cdots P_r)_t\}$ , where  $J$  is a
$-invertible $-ideal, and  $P_1, \ldots, P_r$ 
 $\text{\rm textup}\{()\text{\rm rgei}\}\text{\rm textup}\{\}$  are pairwise
$-comaximal prime $-ideals.
\end{thm}
```

---

It was not only an example of terrible mathematical writing. Search for clarity, above all. Some more words for the mathematically inclined: avoid overlong descriptions of sets, where the reader has to try hard in order to find the exit. Better say: "let  $X$  be the set...".

Parentheses in a math paper should always be upright. Most publisher have special fonts for this, so it is not necessary to write `\textup{()}` all the time. A good copy editor is usually able to spot and correct them, in any case.

## 7 Help!

---

Original input

---

```
{\bf Proof:} Let  $\$g \in G_{H=B} +$ 
\sum\limits_{\{b \in \mathbb{N}^*\}}
\sum\limits_{\{n < w\}} R_{\{b\}} y_{\{b\}} b^{\{n\}} \$.
Then there exist a finite subset  $\{N\}$  of  $\mathbb{N}^*$ ,
 $\{b \in B, a_{\{b,n\}} \in R_{\{b\}}\}$  \\
 $\{b \in N, n \leq k\}$  such that
\centerline{$g = \sum\limits_{\{b \in N\}} \sum\limits_{\{n \leq k\}} a_{\{b,n\}} y_{\{b\}} b^{\{n\}}$}
Since  $y_{\{b\}} = \frac{p^k}{p^n} y_{\{b\}} b^{\{k\}}$  in
 $B' \setminus B$  this expression reduces to
\centerline{$g = \sum\limits_{\{b \in N\}} a_{\{b\}} y_{\{b\}} b^{\{k\}}$}
for some  $a_{\{b\}} \in R_{\{b\}}$  ( $b \in N$ ), \\
 $b \in B'$ .
```

---

```
Putting $N=\{\b{in} N'\}, |\ , a_{\b{not}=0}\} \ 
(N\not=\emptyset\text{ for } g \notin B)$ the 
conclusion of the lemma follows since 
$[y_{\b{b}}]_1\cap[y_{\b{b}}]_1$ is 
finite for $\b{b}\not=\b{b}'$ by 
Corollary\ref{predcor}(iii). $\hfill\square$
```

I wanted to reproduce a pretty large extract of this paper just to show how it is possible to write unthinkable things.

It seems the author is sufficiently acquainted with L<sup>A</sup>T<sub>E</sub>X to know the difference between `\notin` and `\not\in` (which must not be used); however the same author ignores the existence of the `displaymath` environment and emulates it with a complicated construction with `\centerline` (an unsupported command).

Another horror is worth noting: this author has the habit to use “shorthands” for the greek letters (`\b` stands for `\beta` and so on; see later on). This has the consequence of obfuscating the code, making it difficult to spot things on the manuscript, especially when a “`b`” and a “`\b`” are near to each other.

For the mathematically inclined: please note

---

Original input

---

```
$b\in B,\ k\in w,\ a_{\b{b},n}\in R_{\b{b}}\ 
\ (\b{b}\in N',\ ,n\leq k)$
```

---

where *five* separated formulas are joined into one! Keep every formula segregated in the proper way.

I find particularly exhilarating the following:

---

Original input

---

```
Putting $N=\{\b{in} N'\}, |\ , a_{\b{not}=0}\} \ 
(N\not=\emptyset\text{ for } g \notin B)$ the 
[...]
```

---

which, of course, should have been written

---

Correct input

---

```
Putting $N=\{\b{beta}\in N' \mid a_{\b{beta}\not=0}\} \ 
($N\not=\emptyset$ for $g \notin B$), the\ldots
```

---

The relation symbol `\mid` is unknown to most authors, it seems.

## 8 I'm drowning!

---

Original input

---

```
\item Let $$ be the Diophantine monoid\\
$$ = \{ x\in N^3 \mid 2x_1 + 5x_2 = 3x_3 \} . \\ 
As already seen, $$ is generated by \\
$g_1 = (3, 0, 2), g_2 = (0, 3, 5), \\
g_3 = (1, 2, 4), g_4 = (2, 1, 3) . \\
One verifies easily that representation by $g_1$ \\
and $g_2$ is unique and the Cale representations \\
of $g_3$ and $g_4$ by base $Q = \{g_1, g_2\}$ are\\
$3g_3 = g_1 + 2g_2 \ \backslash \boxed{and} \ 3g_4 = 2g_1 + g_2 . \\
It follows that $k = 2, l = 4$ and $m = 3, \\
m'_1 = m'_2 = 3, m'_3 = m'_4 = 1$.
```

---

The author is in total havoc: arbitrary spacing, unseparated formulas, chaotically forced new lines. All in a single `\item` in a list. When you send a pa-

per for subsequent typesetting, don't worry about “Overfull hbox” messages; it is the task of the copy editor to reshape the paper for the final format. Concentrate on the writing, not on the form. But try to write correct code, first of all.

## 9 Is it L<sup>A</sup>T<sub>E</sub>X?

---

Original input

---

```
%%% THIS IS A LATEX FILE %%%%
\documentclass[10pt]{article}
```

---

```
\newtheorem{thm}{Theorem}[section]
\newtheorem{cor}[thm]{Corollary}
\newtheorem{lem}[thm]{Lemma}
\newtheorem{prop}[thm]{Proposition}
\newtheorem{ex}[thm]{Example}
\newtheorem{rem}[thm]{Remark}
%%%%%
\baselineskip=14pt \hsize=5in
\vsize=8in \voffset=5truemm
\hfuzz=10pt
\def\ind{\hspace{0.125in}\relax}
\parindent=0pt
\pagestyle{plain}
```

---

This file announces itself as a L<sup>A</sup>T<sub>E</sub>X document. We can easily see this is a lie. Let's see how it goes along.

---

Original input

---

```
\begin{document}
```

---

```
\title{ \large FACTORIZATION OF DIVISORIAL IDEALS \\
IN A GENERALIZED KRULL DOMAIN} \medskip
\author{\bf XXXX Xx XXXXXXXX\\
\smallskip \\
\rm \\
\small Department of Mathematics\\
\small XXXXXXXX XXXXXXXXX XX XXXXXXXXXXXX \\
\small X.X. XXX 000, XXXX XXXXXX, XXXXXXXX\\
\small \texttt{\small XXXXXXXX@XXXXXX.XX.XX}}
```

---

```
\date{ }
\maketitle
\bigskip
\null \hspace{6cm} {\it Dedicated to Robert W. Gilmer}%
\skip0.66in
```

---

A simple `center` environment could have saved the trouble with the dedication: why set 6 cm of space without knowing the line width? And why mix imperial and metric systems? Recall that AMS classes have a `\dedicatory` command to be given before `\maketitle`.

It is difficult to find so many horrors in such a little space, you will be thinking. Well, no: someone is able to do better, we'll see. However, here is the beginning of the paper.

---

Original input

---

```
\section{\normalsize INTRODUCTION}
\ind Dedekind domains are characterized as the class
of Pr\"ufer domains in which each nonzero ideal is a
finite product of prime ideals. The PVMD-analogue
of this factorization property holds for
```

---

[...] property for ideals in Krull domains to generalized Krull domains, in the same spirit of a work on generalized Dedekind domains by Gabelli and Popescu [8]. \par \ind A generalized Krull domain (GK-domain for short) is a PVMD such that \$P\not=(P^2)\_t\$, for each \$t\$-prime ideal \$P\$, and each nonzero principal ideal has only finitely many minimal (\$t\$)-primes (cf. [5, Theorem 3.9]). GK-domains of \$t\$-dimension one coincide with the class of Krull domains. For more details see [5].\par

Apparently the author doesn't like that the first paragraph after a section title is not indented. So a wonderful idea came to his mind.

---

Original input

```
\def\ind{\hskip 0.125in\relax}
\parindent0pt
```

---

This is the wonderful idea! Writing \par everywhere and beginning all paragraphs with \ind. Where does the \hskip 0.125in\relax come from, I don't know. Let's see how one could have achieved a 'slightly better' result:

---

Correct input

```
\usepackage{indentfirst}
```

---

Notice also the error in cf. [5, Theorem 3.9] which should be written with the automatic commands and the non-breaking space:

---

Correct input

```
cf.~\cite[Theorem~3.9]{a-label}
```

---

But these are subtleties.

## 10 A touch of class

---

Original input

```
\title[\tiny\upshape\rmfamily Modules induced from
a normal subgroup of prime index]{}

\begin{center}\large\sffamily\mdseries
Modules induced from a normal subgroup
of prime index
\end{center}

\author{\sffamily X.,X. XXXXX}
```

---

Trying to modify the typesetting of their documents is common among authors. This piece of evidence is hilarious.

## 11 Definition frenzy

---

Original input

```
\def\Ass{{\rm Ass}}
\def\alpha{\alpha}
\def\lambda{\lambda}
```

---

These are only a few definitions set by an author. We all know, of course, that they are all wrong. Some times ago, an author who used to redefine \c wrote to me asking to solve a problem: the name of an author in the bibliography required a cedilla

and, strangely, it appeared that there was no way to make L<sup>A</sup>T<sub>E</sub>X do this.

"Wait! Writing \alpha is too long!" objects the author. The answers to this objection can be: (1) use a smart editor; (2) the L<sup>A</sup>T<sub>E</sub>X source file is more readable. Emacs is not the only smart editor, there are many on every platform and some of them are free. For GNU/Linux there is Kile.

Regarding the definitions, consult a manual in order to find the better way to set them; for a "log" type operator, use the amsmath package and the command \DeclareMathOperator:

---

Correct input

```
\DeclareMathOperator{\Ass}{Ass}
```

---

I should say that this operator is common in Commutative Algebra: it denotes the set of 'associated ideals'. It is not correct to define an operator like that as \textrm{Ass}, because the spacing is excessive and in some cases wrong; compare \log x and \log(xy).

A paper I worked on recently had 250 lines of various definitions. Only 38 remained. When you send a work to others, ensure to erase from the preamble all unused commands. Most of all, ensure that all loaded packages are included in the standard distributions, otherwise send them along with the paper.

## 12 Theorems

---

Original input

```
\newtheorem{satz}{Theorem}[section]
\newtheorem{prop}[satz]{Proposition}
\newtheorem{cor}[satz]{Corollary}
\newtheorem{lem}[satz]{Lemma}
\newtheorem{rem}[satz]{Remark}
\newtheorem{ex}[satz]{Example}
\newtheorem{exas}[satz]{Examples}

\newcommand{\Bth}[1]{\smallskip \begin{satz}
\label{#1} \begin{rm}}
\newcommand{\Eth}{\end{rm} \end{satz}}
```

```
\newcommand{\Bcor}[1]{\smallskip \begin{cor}
\label{#1} \begin{rm}}
\newcommand{\Ecor}{\end{rm} \end{cor} \medskip}
```

```
\newcommand{\Bex}[1]{\smallskip \begin{ex}
\label{#1} \begin{rm}}
\newcommand{\Eex}{\end{ex} \medskip}
[...]
```

---

There should be some obscure reasons why a \medskip is needed after a corollary, but not after a theorem. And to impose a completely useless \smallskip before a statement. There were similar definitions of pseudo-environments for the other kinds of statements.

A common typographical practice is to set statements of theorems in italics, in order to distinguish

them visually from the normal text: they are important and ought to be found easily. It is acceptable, of course, that someone prefers to set them in upright roman type. This author, at least, groups every definition in the preamble, so that the copy editor can easily change the format.

However, the point of this example is not only the format of the statements. What is the difference in typing between

```
\Bth{mylabel}
$2+2=4$.
\Eth
and
\begin{satz}\label{mylabel}
$2+2=4$.
\end{satz}
```

excluding from consideration the number of characters? I see many advantages to the second form. For instance, you can search your document for labels using a well defined key (i.e., `\label`). I am not a fan of completely structured documents *à la* XML, though they have their importance; but a typed paper is not the same as an XML document. With properly used environments, you add structure to the document and so it becomes easier to thumb through it.

A smart text editor is able to produce a skeleton of any environment you need with only a few keystrokes.

'What about the upright type?' you ask. Well, using `amsthm` or `ntheorem` it is very easy.

---

Correct input

```
\usepackage{amsthm}
\theoremstyle{definition}
\newtheorem{satz}{Theorem}
\newtheorem{ex*}[satz]{Example}
\newenvironment{ex}{\begin{ex*}}{\qed\end{ex*}}
```

---

There is also a solution, a bit more complex, if you want to be able to say also `\qedhere` in an `ex` environment.

```
\makeatletter
\newenvironment{ex}
{\pushQED{\qed}\begin{ex*}}
{\popQED\end{ex*}\@endpefalse}
\makeatother
```

### 13 This author knows what to do!

---

Original input

```
% Number math displays as subsubsections
%
\renewcommand{\theequation}{\thesubsubsection}
\newcommand{\eqlabel}[1]{%
  \refstepcounter{subsubsection} \label{#1} }
% Recall: \eqref{A} refers to the equation
% labeled "A", with the label A enclosed
% in parentheses.
```

---

```
%% Number displayed statements as
%% subsubsections
\newcommand{\dsp}{%
  \refstepcounter{subsubsection}
  \begin{trivlist}
  \item[] (\thesubsubsection)
  \hangindent=2em \hangafter 1 }

%%
\newcommand{\ndsp}{\end{trivlist} \smallskip}
%% Usage:
%%   \dsp \label{unforgettable}
%%   <text to be displayed with equation-number>
%%   \ndsp
```

---

No, he doesn't. The above excerpt resembles a professional L<sup>A</sup>T<sub>E</sub>X program, but its purpose was only to number equations inside sections. You can achieve the same result far more easily and simply with the `amsmath` package and

---

Correct input

```
\numberwithin{equation}{section}
```

---

Don't overuse the sectional commands; and also don't use subsubsections if you don't have subsections!

To be honest, the article contained also

---

Original input

```
\newtheorem{Theorem}[subsection]{Theorem}
\newtheorem{Lemma}[subsection]{Lemma}
```

---

and so on, so it had subsections! Of course, it is easier to achieve the same result with

---

Correct input

```
\newtheorem{Theorem}{Theorem}[section]
\newtheorem{Lemma}[Theorem]{Lemma}
\numberwithin{equation}{Theorem}
```

---

For the `dsp` environment, a very easy replacement could be

---

Correct input

```
\newenvironment{dsp}
  {\refstepcounter{equation}\begin{trivlist}
  \item[] (\theequation)
  \end{trivlist}}
```

---

It is not advisable to tamper with `\hangindent` and `\hangafter` if you don't know what you are doing. The spacing chosen by the author was arbitrary, to say the least.

## 14 Groups

---

Original input

```
\newcommand{\supp}[1]{%
  \mbox{$\left[ \, , \, #1 \, , \, \right]$}}
\newcommand{\suppl}[1]{%
  \mbox{$\left[ \, , \, #1 \, , \, \right]_\lambda$}}
\newcommand{\suppa}[1]{%
  \mbox{$\left[ \, , \, #1 \, , \, \right]_A$}}
\newcommand{\norm}[1]{%
  \mbox{$\parallel #1 \parallel$}}
\newcommand{\norma}[1]{%
  \mbox{$\parallel #1 \parallel_A$}}
```

---

The purpose of putting all these things inside a `\mbox` is an unconceivable mystery. Note the astounding weirdness of using a relation symbol such as `\parallel` instead of the correct delimiter `\|`. If the definition is without `\left` and `\right`, it would be better to use the pair `\lVert`—`\rVert` available with `amsmath`.

More: all variation must be defined in abstract terms, so that it is sufficient to change only one of them, if needed.

---

Correct input

```
\newcommand{\supp}[1]{\left[\begin{array}{c} \left.\rule{0pt}{1.5ex}\right. \\[-0.9ex] \supp{#1} \end{array}\right]}
\newcommand{\suppl}[1]{\left[\begin{array}{c} \left.\rule{0pt}{1.5ex}\right. \\[-0.9ex] \supp{#1} \lambda \end{array}\right]}
\newcommand{\suppa}[1]{\left[\begin{array}{c} \left.\rule{0pt}{1.5ex}\right. \\[-0.9ex] \supp{#1}_A \end{array}\right]}
\newcommand{\norm}[1]{\left[\begin{array}{c} \left.\rule{0pt}{1.5ex}\right. \\[-0.9ex] \left|\rule{0pt}{1.5ex}\right. \\[-0.9ex] \norm{#1} \end{array}\right]}
\newcommand{\norma}[1]{\left[\begin{array}{c} \left.\rule{0pt}{1.5ex}\right. \\[-0.9ex] \left|\rule{0pt}{1.5ex}\right. \\[-0.9ex] \norm{#1}_A \end{array}\right]}
```

---

## 15 Uniformity

---

Original input

```
\{\gbar = g+U: g\in G\}
\$ \supp{g}=\{(\alpha,i)\in\mathbb{N}\times\rho,\\ \quad |g-\alpha_i|\neq 0\} \$
\$ G^{\alpha}=\{g\in G, \forall i, \norm{g}<\alpha_i\} \quad (\alpha<\mathbb{N}). \$
```

---

These three excerpts didn't show in the same page, but they were *in one and the same paper*. Three different notations are used to denote one and the same thing.

Don't do this yourself, try to be consistent. It can be convenient to define a suitable command, for example

```
\newcommand{\set}[2]{\{ \mid , #1 \mid \#2 \mid , \}}
```

## 16 Two consecutive paragraphs

---

Original input

```
Recall [ComRo, page 5] that a Tychonoff space is called $\omega$-compact provided every its countable subset lies in a compact subspace.
```

---

```
Recall $\mbox{\rm{[U1, p. 39]}$ that a topological ring $R$ is said to be $\omega$-compact if:
```

I'm not blaming the use of the English language, of course. You can spot strange ways of inserting similar objects, in this case bibliographical citations. The use of `\cite` allows for avoiding the problem that I think was at the origin of the mysterious `\mbox{\rm{...}}`: the fact that in statements of theorems, the citation (when done by hand) comes up in italics. Verify, as an exercise, that `\cite` yields the citation in roman.

---

Correct input

```
Recall that a Tychonoff space is called $\omega$-compact $\cite[p.~5]{ComRo}$ when each of its countable subsets lies in a compact subspace.
```

---

```
Recall that a topological ring $R$ is said to be
```

---

```
\left.\rule{0pt}{1.5ex}\right. \cite[p.~39]{U1} if ...
```

---

Don't forget the non-breakable space; use `\emph` instead of explicit font change commands.

## 17 Bibliographies

---

Original input

```
\section*{References}
\begin{itemize}
\item[[CR90]] C.W.\,Curtis and I.\,Reiner, Methods of Representation Theory: with Applications to Finite Groups and Orders, Vol. 1, Classic Library Edn, John Wiley and Sons, 1990.
```

---

The `thebibliography` environment does what the author painfully wants to achieve by hand and much more better, because it sets up a list taking into account the width of the label given as an argument.

Note the pathetic `[[[CR90]]]` and the arbitrary spacing between initials and surnames. I prefer to space normally the initials, others recommend a thin space. Act consistently.

---

Correct input

```
\begin{thebibliography}{CR90}
\bibitem[CR90]{CR90} C. W. Curtis and I. Reiner, Methods of Representation Theory: with Applications to Finite Groups and Orders, Vol.~1, Classic Library Edition, John Wiley and Sons, 1990.
```

---

Another bibliography.

---

Original input

```
\bibitem[HKK]{HKK} C.Y.Hong, N.K.Kim and T.K.Kwak, Ore extensions of Baer and p.p.-rings, J. Pure Appl. Algebra {\bf 151} (2000), 215–226.
\bibitem[JSS]{JSS} Jón drup, p.p.rings and finitely generated flat ideals, Proc.Amer.Soc. {\bf 28} (1971), 431–435.
```

---

Line 1: Fortunately, it is rare to see similar things. Write C.\,Y. Hong, if you prefer, but definitely put a space between initials and surname.

Line 3: In the frenzy of font changing the spaces are lost. Perhaps, using `\textbf` instead of the obsolete `\bf`, it wouldn't have happened.

Line 4: We can't understand what the command `\/` is there for; the name of the cited author is “Jón drup”; probably the author said to himself: “The slash is a kind of accent, it should be done with a slash, shouldn't it?” No.

Line 6: Inconsistent spacing; the name of the journal is wrong; the number intervals must be denoted with the en dash.

---

Correct input

```
\bibitem[HKK]{HKK} C. Y. Hong, N. K. Kim and T. K. Kwak, Ore extensions of Baer and p.p.-rings, J. Pure Appl. Algebra {\textbf{151}} (2000), 215–226.
```

---

```
\bibitem{J}
S. J{\o}ndrup, p.p.rings and finitely generated flat
ideals, Proc. Amer. Math. Soc.
\textbf{28}(1971), 431--435.
```

## 18 Inventing hot water

---

Original input

---

```
\newenvironment{demo}{\par\noindent
{\bf Proof.}}{$\rule{4pt}{8pt}$\par}
```

---

A variation on the theme of proofs. Don't reinvent the wheel—use `amsthm` which also provides the command `\qedhere` to solve the puzzling situations when a proof ends with a displayed equation.<sup>2</sup>

The proposed definition is *completely wrong*:

1. it doesn't set vertical space before and after the environment.
2. it uses obsolete commands;
3. puts a black mark after the last full stop in the proof, with an awful effect.

## 19 Inventing lukewarm water

---

Original input

---

```
A distance $d$ for $G$ is said to be faithful,
if the following (*) is satisfied.
```

---

```
\vspace{1ex}

(*) \ \ \ If $d(\alpha, \alpha') = 0$, then
$G_{\{\alpha\}}$ and
$G_{\{\alpha'\}}$ are isomorphic.
```

```
\vspace{1ex}

As a matter of form, at least one faithful distance
always exists.
```

---

What to say about this gem? The author is puzzled with a condition to which he doesn't want to give a name, but a symbol. Use `amsmath`.

---

Correct input

---

```
A distance $d$ for $G$ is said to be faithful,
if the following $\theta$ is satisfied.
\begin{equation}\tag{*}
\text{If } d(\alpha, \alpha') = 0, \text{ then } G_{\{\alpha\}}
\text{ and } G_{\{\alpha'\}} \text{ are isomorphic.}
\end{equation}
```

---

Such a label can be referred to in abstract way with `\thetag{*}`; the document class will provide the correct parentheses around the symbol. You can also assign a `\label` to the equation and use `\ref` or `\eqref`.

---

<sup>2</sup> This is an occasion to remember Michael J. Downes, esteemed developer of AMS-LATEX; we all miss him.

## 20 Inventing cold water

---

Original input

---

```
\def\yi{\mbox{\rm (i) }\hspace{0.3em}}
\def\yii{\mbox{\rm (ii) }\hspace{0.3em}}
\def\yiii{\mbox{\rm (iii) }\hspace{0.3em}}
\def\yiv{\mbox{\rm (iv) }\hspace{0.3em}}
\def\yv{\mbox{\rm (v) }\hspace{0.3em}}
%
\def\ya{\mbox{\rm (a) }\hspace{0.3em}}
\def\yb{\mbox{\rm (b) }\hspace{0.3em}}
\def\yc{\mbox{\rm (c) }\hspace{0.3em}}
\def\yd{\mbox{\rm (d) }\hspace{0.3em}}
\def\ye{\mbox{\rm (e) }\hspace{0.3em}}
```

---

The author defines by hand the list making commands. Enough said.

## 21 The search for perfection

---

Original input

---

```
\section{\protect\bigskip Introduction}
```

---



---

Original input

---

```
\section{$\bigskip$Introduction}
```

---

The two examples are drawn from different documents. Their authors wanted to increase the spacing between the title and the text of the section. Clearly they did it in a terrible way; the second example is ridiculous. What a surprise when compiling also the table of contents!

These decisions are best left to the document class, in particular if you are sending the paper for subsequent copy editing. If you feel that the spacing is not satisfying for your personal taste and for personal documents, then use packages such as `sectsty` or `titlesec`.

## 22 Nirvana

---

Original input

---

```
\title{Semigroup$\!$ Rings$\!$ that$\!$ are$\!$ Inside$\!$ Factorial$\!$\\
$\!$and$\!$ their$\!$ Cale$\!$ Representation}
```

---

*Lo giuro sul mio onore.* This is a quotation from Mozart's "Don Giovanni", libretto by Lorenzo Da Ponte, Act I, Scene 4. But it is that scoundrel Don Giovanni who says that, the reader will object. Yes, but trust me that the example is real, I still keep it religiously.

◊ Enrico Gregorio  
Dipartimento di Informatica  
Università di Verona  
Enrico.Gregorio (at) univr.it  
http://profs.sci.univr.it/~gregorio

## Macros

### Eplain 3: Expanded plain $\text{\TeX}$

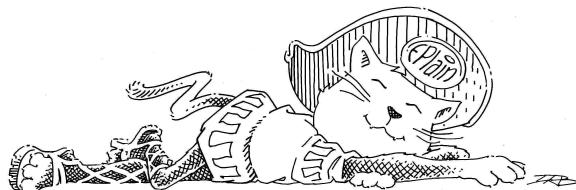
Karl Berry and Oleg Katsiadze

#### Abstract

Eplain is a macro package which extends the definitions in plain  $\text{\TeX}$ . It provides many conveniences which most document writers need, but without forcing any specific typographic style on an author. Since its inception around 1990, Eplain has provided facilities for citations, tables of contents, symbolic cross-references, indexing, and multiple column output.

As of version 3, Eplain also provides support for live hyperlinks in PDF documents and for loading conveniently a few  $\text{\LaTeX}$  packages (notably *graphics.sty* for including images and rotating and scaling text, *color.sty* for colored text, and *url.sty* for typesetting paths and URLs).

This article gives a brief introduction to Eplain for newcomers to the package and then discusses some of the new features in more detail.



#### Introduction

The original plain  $\text{\TeX}$  is more or less a low-level interface to  $\text{\TeX}$  primitives (while providing a few user-oriented macros). It lacks many features which most document writers will reasonably expect, implementation of which requires some experience in  $\text{\TeX}$  macro programming language. A good example is using labels for cross-references. Instead of manually inserting absolute numbers throughout the manuscript, authors like to assign labels to various parts of the document such as sections, figures, etc. When later  $\text{\TeX}$  encounters a cross-reference with a label, it automatically changes the label to the appropriate number. This saves lots of manual work when parts of the document are reordered.

Such features are available in  $\text{\LaTeX}$  (originated by Leslie Lamport), but the approach taken

by  $\text{\LaTeX}$  is much different from that of plain  $\text{\TeX}$ .  $\text{\LaTeX}$  hides many details by providing users with high-level capabilities, making it easier to write documents with predefined styles and harder to produce bad typesetting. A vast number of additional packages has been developed to allow users to customize  $\text{\LaTeX}$ , but desired changes can still sometimes be hard to accomplish.

The Eplain macro package takes another approach. The philosophy of Eplain is to provide functionality which can be used (or not used) as desired, without forcing any typographic style on an author. Thus, the Eplain macro package expands on and extends the definitions in plain  $\text{\TeX}$ , providing features such as symbolic cross-referencing, lists, citations, indexing and many other capabilities. Eplain provides both macros intended to be used directly in documents and macros to be used as tools in developing formats.

Eplain is extensively documented and has an active mailing list, so problems or questions related to using Eplain are usually quickly resolved. (One new user's experience with Eplain is reported in [6].) Also, the development sources are publicly available via <http://sarovar.org/projects/eplain>, so contributing is straightforward. For additional information about Eplain, please visit the Eplain home page, <http://tug.org/eplain>.

#### Eplain at a glance

The simplest way to use Eplain is to put the line

```
\input eplain
```

at or near the beginning of your  $\text{\TeX}$  document. You then need to process your document with the plain-based *tex* (or *pdftex*, *etex*, etc.), *not latex*.

The *eplain.tex* file should already be included in your  $\text{\TeX}$  installation. If not, you can get it from the Eplain home page or from CTAN [1].

Now let's take a quick tour of some of the features provided by Eplain. More detailed discussion can be found in the Eplain manual [2].

**Multiple columns.** Eplain provides the high-level commands *\doublecolumns*, *\triplecolumns*, and *\quadcolumns* to start multiple-column output. Single column output can then be restored with *\singlecolumn*. These macros do not start a new page, so you can have a multiple column passage in the middle of a page.

**Displays.** To obtain left-aligned math displays, Eplain provides the command *\leftdisplays*. The command *\centereddisplays* switches back to the default of centered displays, should you need to.

**Lists.** Eplain provides for arbitrarily nested lists that can be either numbered (`\numberedlist ... \endnumberedlist`) or bulleted (`\unorderedlist ... \endunorderedlist`). In both kinds of lists, you begin an item with `\li`, which accepts a cross-reference label as an optional argument. As with practically everything else, Eplain provides many parameters for customizing the lists.

**Checking for PDF output.** Eplain incorporates `ifpdf.sty`, written by Heiko Oberdiek, which provides an `\ifpdf` switch for detecting whether PDF or DVI is being emitted by `TeX`.

**Verbatim.** Eplain supports both in-line verbatim text (with the construct `\verbatim <verbatim text> \endverbatim`) and typesetting the contents of an entire file verbatim (`\listing{<filename>}`). The verbatim listing produced by `\listing` can have optional line numbers in a customizable format.

**Footnotes.** Eplain extends the definitions of plain `TeX` to support automatically numbered footnotes (`\numberedfootnote`) and to allow general customization of footnote spacing, rules, etc.

**Arrow theoretic diagrams.** Eplain incorporates macros (written by Steven Smith) for drawing commutative diagrams. This capability is similar to that found in `LATEX`'s picture mode for drawing slanted lines and vectors of certain directions, and depends upon the `LATEX` font `line10`.

**Cross-references.** Referring readers to other parts of your document is a basic need for authors; but putting literal page, section, equation, or whatever numbers in the text is of course undesirable.

Eplain therefore supports symbolic cross-referencing, both generically (with `\defineref` to define labels, and `\ref` and its variants to refer to them) and specifically to page numbers (`\xrdef` and `\xref`) and (sub)equations (`\eqdef`, `\eqsubdef`, `\eqref` and variants).

**Citations.** The citation macros provided by Eplain are designed to work with the `BIBTeX` program, written by Oren Patashnik. The macros are defined in a separate source file, `btxmac.tex` (which can also be used on its own, without the rest of Eplain).

Citations are typeset with the `\cite{<label>}` command. The actual bibliography is produced with the `\bibliography` command (which reads the requested `.bib` files), and the bibliography style can be selected by `\bibliographystyle`. Many macros and parameters are provided for fine-tuning the formatting of citations and the bibliography.

**Contents.** Producing a table of contents that is both useful and aesthetic is one of the most difficult design problems in any work. Therefore Eplain does not attempt to solve the design problem; instead, it merely provides helper macros (`\writetocentry` and `\writenumberedtocentry`) for collecting the raw data for a table of contents, using an auxiliary file with extension `.toc` (and the same base name as your document). During the next run of `TeX` on the document, the collected information is read at the place(s) where you call the Eplain-provided `\readtocfile` to produce the table of contents.

To obtain a nicely typeset table of contents you will need to define one macro per contents entry type (chapters, sections, etc.), to specify the styles of the entry types.

This functionality can be reused for other kinds of contents lists, such as lists of figures and lists of tables. Invoking `\definecontentsfile{<abbrev>}` creates a set of the macros with which you can manage your own contents lists, with `<abbrev>` replacing `toc` in the macro names and file name extension of the intermediate file.

**Indexing.** Eplain provides macros to produce raw material for an index in the form accepted by the `MakeIndex` program, and to typeset the sorted index produced by `MakeIndex`.

Besides specifying the basic index entries, the indexing commands facilitate indexing of people's names, creating subentries, "see ..." and "see also ..." entries, page ranges, and entries with page numbers set in a different style (italicized, underlined, etc.). Eplain also supports specifying explicit sort strings; for example, in mathematics, it is usually desirable to typeset  $\pi$  with `$\pi$` and sort it as `pi`.

Setting `\indexproofingtrue` instructs Eplain to typeset index terms in the margin of each page, for help when proofreading.

**Programming definitions.** Numerous helper definitions turned out to be useful when implementing Eplain's user-level features. Many of these are documented and available to Eplain users on the chance that people writing other macros will also find them useful. Among them are: "inner" variants of `\newcount` and friends which can be called inside other macros; a `\for` macro for iteration over a comma-separated list of items; macros for defining general hooks and properties and named environments; macros for managing auxiliary files and manipulating character category codes; and many others.

**Miscellaneous.** Eplain also provides nifty typesetting conveniences such as rules with adjustable default width, height and depth; solid and unfilled boxes of specified dimensions, and boxed text; the time and date in various formats; fractions; long pathnames, electronic mail addresses and URLs (this support comes from `path.sty`, written by Nelson Beebe and Philip Taylor); and various TeX-related logos (from `texnames.sty`, compiled by Nelson Beebe). Finally, when things inevitably go wrong, there are macros for tuning the diagnostic output.

### L<sup>A</sup>T<sub>E</sub>X packages in (E)plain

In version 3.0, which was released in September 2005, Eplain acquired the capability to load *some* L<sup>A</sup>T<sub>E</sub>X packages. You may find this feature of Eplain useful when working with plain TeX, even if you do not plan to use the rest of Eplain. Eplain uses David Carlisle's `miniltx.tex` [4] for this (described below), with extensions to support package options and a few other features.

Of course, most L<sup>A</sup>T<sub>E</sub>X packages don't make sense under plain TeX; the overwhelming majority of L<sup>A</sup>T<sub>E</sub>X packages that have been developed can't be used with Eplain, or plain TeX in general. However, some packages provide general capabilities which in principle are independent of the L<sup>A</sup>T<sub>E</sub>X engine.

The `graphics` bundle is a notable example; it provides for graphics inclusion and also rotating, scaling and coloring of text. These features are not provided by TeX itself; instead the packages must rely on the capabilities of the output driver (typically for PostScript or PDF) to do the job.

These features are just as useful in plain TeX as they are in L<sup>A</sup>T<sub>E</sub>X, but instead of rewriting all the packages in the `graphics` bundle for plain TeX, the L<sup>A</sup>T<sub>E</sub>X3 team developed `miniltx.tex`, a "mini-L<sup>A</sup>T<sub>E</sub>X environment", which provides stubs for or simplified parts of L<sup>A</sup>T<sub>E</sub>X used by the packages in the `graphics` bundle, so that those packages can be loaded in plain TeX after loading `miniltx.tex`.

The definitions in `miniltx.tex` were designed with the `graphics` packages in mind; therefore these definitions are generally not sufficient for loading other packages. Furthermore, `miniltx.tex` provides no support for specifying package options. So Eplain builds on top of `miniltx.tex` to support package options and a few additional packages.

**Loading L<sup>A</sup>T<sub>E</sub>X packages.** To load a L<sup>A</sup>T<sub>E</sub>X package in Eplain, call `\usepackage` (the same name is used in L<sup>A</sup>T<sub>E</sub>X), wrapped in a `\begin{packages} ... \end{packages}`

`\end{packages}` block. This block serves as a substitute for L<sup>A</sup>T<sub>E</sub>X's preamble, so it is best to specify only one such block per TeX job. For example:

```
\begin{packages}
\usepackage{graphicx,color}
\usepackage{url}
\end{packages}
```

will load the `graphicx`, `color` and `url` packages, with no options.

The following L<sup>A</sup>T<sub>E</sub>X packages (all on CTAN) are known to work under Eplain:

- `autopict` (L<sup>A</sup>T<sub>E</sub>X picture mode);
- `color` (color support);
- `graphics`, `graphicx` (graphics inclusion);
- `psfrag` (overlay L<sup>A</sup>T<sub>E</sub>X onto EPS figures);
- `url` (smart line breaking for URLs).

We hope to support other packages in the future.

**Implementation.** For those who may be interested in the TeXnical details, here is an overview of the extensions made by Eplain to `miniltx.tex`.

- We redefine the `\DeclareOption` macro (a no-op in `miniltx.tex`) to save the code implementing the option. Next, we redefine the `\ProcessOptions` and `\ExecuteOptions` macros to execute the code which enables relevant options (they are defined as no-ops by `miniltx.tex`). Also, we implement the star ('\*) option to declare a default option handler, used to process undeclared options.
- We define `\PassOptionsToPackage` (missing in `miniltx.tex`) to let packages pass options to other packages which they load.
- We define `\AtBeginDocument` to accumulate its arguments for execution at the end of the `\begin{packages} ... \end{packages}` block, instead of the immediate execution defined by `miniltx.tex`. Also, `\AtEndOfPackage` (missing from `miniltx.tex`) is defined to delay execution of the argument until after the current package is loaded.
- In `\ProvidesFile` and `\ProvidesPackage`, we define the macros `\ver@<package>.sty` and `\ver@<filename>.ext`; these are used by some packages to detect that a package or a file has been loaded. (When a package is requested which has already been loaded, Eplain currently displays an error message; it does no checking for the legitimate situation of a package having already been loaded with a superset of the options in the second request.)

- `\ProvidesPackage` also checks that the date of the package is not older than the date specified by the user in `\usepackage`.
- `\RequirePackage` is redefined to save all extant parameters before loading other packages, and restore them afterwards. That way, if a recursively loaded package loads other packages, or defines its own options or `\AtEndOfPackage` commands, they do not interfere with the state of the upper-level package.

## Hyperlinks

Another feature of Eplain which is new in version 3 is the ability to create live hyperlinks in PDF documents through `pdftEX` or `dvipdfm(x)`.

**Hyperlink drivers.** Since, in addition to `pdftEX`, there are several `.dvi` processors with the ability to generate PDF files with hyperlinks, the hyperlink support in Eplain has a two-layered structure: 1) hyperlink drivers, which provide low-level hyperlink commands (primitives in the case of `pdftEX` and `\special` commands in case of `.dvi` processors); and 2) user commands, which are the same across the drivers (but supporting different subsets of functionality, depending on the driver's capabilities).

Currently, Eplain has two drivers: `pdftex` for `pdftEX`, and `dvipdfm` for `dvipdfm` and `dvipdfmx`. We hope to add more drivers in the future.

One other pseudo-driver named `nolinks` is beneficial when one wishes to typeset a document both with and without hyperlinks.<sup>1</sup>

Eplain comes with the hyperlink support disabled by default (for various reasons). To enable hyperlinks you specify:

```
\enablehyperlinks [<driver>]
```

<sup>1</sup> Here is the `TEx`nical rationale for the `nolinks` driver (feel free to skip). When a hyperlink is inserted, `TEx` creates a *whatsit* (an internal `TEx` object). Whatsits may introduce legitimate breakpoints at places where none would exist otherwise. Now imagine that you want to generate a PDF document both with and without hyperlinks. Completely disabling the hyperlinks for the latter is not ideal, because then the whatsits will not be generated and the resulting PDF may end up with different page and line breaking than the former. Therefore, it is best to keep hyperlinks enabled, while selecting the `nolinks` driver. This defines all hyperlink commands to produce a whatsit that does nothing (writes to a log file), thus imitating the whatsits from the “real” hyperlink commands. (This trick was borrowed from `color.sty`.)

after you have loaded Eplain in your document. If the optional argument, `[<driver>]`, is omitted, Eplain tries to detect the appropriate driver.

**Implicit hyperlinks.** When hyperlinks are enabled, many Eplain macros automatically start to use hyperlinks in their output. For example, cross-reference macros then render the cross-reference as a hyperlink pointing to the location being referenced. Here is a list of features which create such implicit hyperlinks:

- `\url` (from `url.sty`, see previous section);
- cross-reference macros;
- `BIBTeX` citations;
- numbered and unnumbered lists;
- indexing;
- footnotes.

All macros which create implicit hyperlinks are assigned to one of the so-called *hyperlink groups*, roughly corresponding to the above features, so that parameters such as link border width or colors can be set individually for each group. For example, all equation reference macros are assigned to the ‘eq’ hyperlink group; thus, you can customize parameters for equation hyperlinks without affecting other kinds.

**Explicit hyperlinks.** Sometimes you might need to create a hyperlink explicitly. This is done by first creating a hyperlink destination with the command

```
\hldest{<type>}{<options>}{<label>}
```

Supported destination types and options depend on the selected hyperlink driver, while `<label>` identifies the destination.

Now, to create a link to that destination, use:

```
\hlstart{<type>}{<options>}{<label>}
```

```
...
```

```
\hlend
```

Whatever text you write in the ... becomes a hyperlink pointing to the destination identified by `<label>`. Here again, available link types and options depend on the selected hyperlink driver.

Eplain provides a way to set default destination and link types and options, so that you don't need to specify the same parameters over and over in each call.

**Implementation.** Our first attempt at implementing hyperlinks was to write wrappers around relevant Eplain macros, extending them with hyper-linking capabilities. Although this was functionally implemented and even used to typeset an electronic book, coding of the wrappers turned out to be quite difficult, and adapting the wrappers for other

projects would not have been an easy task. Even worse, many wrappers were fragile, in the sense that they were greatly relying on the internals of Eplain macro definitions; thus, they would have been hard to maintain in future versions.

The logical solution was to add the hyperlink capability directly into Eplain macros. Let's look briefly at the hyperlink implementation in Eplain.

Hyperlink macros in Eplain are structured so that it is relatively easy to add support for new hyperlink engines by writing a new driver. A new driver can be modeled on the existing ones, and, in short, should define macros with certain names so that the driver-independent hyperlink macros can detect the driver, plus define link and destination handlers for each of the types it supports.

Each driver defines default destination and link types and default values for the supported options, to use in the absence of user-specified values in a call to `\hdest` or `\lstart`. Of course, these defaults can be overridden by the user. In addition, the user can set default options and types for each of the link and destination groups, which, when set, will override the global defaults.

Maintaining default options for destination and link groups is a little tricky. We don't want to define a macro per group per option to hold the value, because a lot of T<sub>E</sub>X's memory would be wasted just storing the names of those macros. Instead, for each destination or link group, a list of default options given by the user is saved as a comma-separated list of assignments in the form `<option>=<value>`. This list is consulted whenever a macro from the group needs to create a link or destination. If any option is missing from this list, a global default for this option (defined by the driver or specified by the user) is used.

### An annotated example

Suppose we are using pdft<sub>E</sub>X, have a figure we want to insert, scaled to fit our format, and we want to refer to this figure from the text via a cross-reference label. Also, we want to typeset a URL as a live hyperlink and a line of colored text.

Let's see how this can be done in Eplain. Consider the source file `eplndemo.tex` shown in Example 1, along with its output. Numbers at the beginning of the lines are not part of `eplndemo.tex`, they are merely to ease references in the comments to follow.

In order to compile `eplndemo.tex`, you will need the CTAN lion drawing by Duane Bibby, which you can download from [3]. Place the image

---

```

1. \input eplain
2.
3. \begin{packages}
4.   \usepackage{url}[2005/06/27]
5.   \usepackage[dvipsnames]{color}
6.   \usepackage{graphicx}
7. \end{packages}
8.
9. \enablehyperlinks
10. \hlopts{bwidth=0}
11. \hlopts[url]{colormodel=named,%
12.               color=OliveGreen}
13. \nopagenumbers
14. \def\figureword{fig.}
15.
16. \vbox{
17.   \defineref{CTANlion}{1}{figure}
18.   \noindent
19.   \includegraphics[width=200pt]
20.     {ctan_lion_350x350}
21.
22. \noindent Figure~1: lion in the archives.
23. }
24. \medskip
25.
26. See the lion in \ref{CTANlion}.
27.
28. Take me to \url{http://tug.org/eplain}.
29.
30. {\color[rgb]{0.3,0.3,0.3} Paint it gray.}
31.
32. \bye

```

---



Figure 1: lion in the archives.

See the lion in `fig. 1`.

Take me to <http://tug.org/eplain>.

Paint it gray.

---

**Example 1:** Eplain source file and output.

file in the same directory with `eplndemo.tex`, and change to that directory. Now, to produce a PDF, run pdf<sub>T</sub>E<sub>X</sub> (twice):

```
pdftex eplndemo.tex
pdftex eplndemo.tex
```

During the first run, Eplain will write the information about the cross-reference into `eplndemo.aux`, and during the second run this information will be read by Eplain to typeset the reference.

Now, let's see what all these commands mean.

- On line 1, we load Eplain.
- On lines 3–7, we load three L<sup>A</sup>T<sub>E</sub>X packages.
  - `url.sty` provides the `\url` command to conveniently typeset a URL. We request that `url.sty` be the version from June 27, 2005, or later, because earlier versions had problems interacting with plain T<sub>E</sub>X.
  - `color.sty` provides support for colored text; all hyperlinks are automatically colored by Eplain when this package is loaded. We give the `dvipsnames` option because we want to use named colors from the `dvips` graphics driver.
  - Finally, we load `graphicx.sty`, for the macro `\includegraphics`.
- Recall that hyperlinks are off by default. Therefore, we enable them on line 9.
- On lines 10–12, we customize some hyperlink options.
  - First, we set the border width to 0 for all links, to omit the default boxes around links (we prefer colored links).
  - Next, we specify that all links in the `url` hyperlink group (meaning the `\url` command from `url.sty`) should be colored using the named color `OliveGreen`. The default is the dark red shown in the `\ref` output.
  - The % at the end of line 11 prevents T<sub>E</sub>X from converting the end-of-line character into a space token. (We have short source lines simply because of *TUGboat*'s narrow columns.)
- On line 13, we inhibit page numbering for this one-page document. (A plain T<sub>E</sub>X command.)
- On line 14, we define the output word for the cross-reference class `figure`. This word will be prepended by Eplain to references created via `\ref` (read on to see its use).
- Now comes the fun part! On lines 16–23, we create the figure with the CTAN lion image.

- We start by defining a symbolic label so that we can later refer to the figure with `\ref`. The command `\defineref` on line 17 takes the following arguments:

- The cross-reference label (`CTANlion`).
- The reference text (1, the figure number). If you have many figures, you may want to use a count register to keep track of the current figure number, and to use its value here as the reference text.
- The class of the label (`figure`). The label class determines the word placed by `\ref` in front of the reference text; recall that we've defined `\figureword` on line 14. (There is nothing magic about the name `figure`; Eplain just uses whatever is defined.)

In addition to the cross-reference label, `\defineref` creates a hyperlink destination with the same label.

- On lines 18–20, we use the `\includegraphics` command (from the package `graphicx.sty` [5]) to load the image, scaled to the width of 200 pt, and place it in a paragraph of its own, without indentation (`\noindent`).
- On line 22, we put the figure's caption underneath the image.
- Now let's typeset some hyperlinks. First, we use the `\ref` cross-reference command (line 26) to refer to the figure using our chosen cross-reference label (`CTANlion`). Eplain automatically inserts the label's class word (`fig.`, defined on line 14) as part of the link (to make sure the reader does not have to aim too hard).
- Let's now point somewhere outside our document; the Eplain home page seems a good target. On line 28, we use the `\url` command from `url.sty`. Remember that we have customized the color of `url` hyperlinks on lines 11 and 12, so it will be rendered with a color different from the default dark red.
- Finally, we produce a line of colored text on line 30. We use the `\color` command from `color.sty`, specifying an exact RGB color `0.3,0.3,0.3` (which results in gray). To limit the effect of the color command to this single line, we enclose the command and the text in a group.
- On line 32, we say good-bye to T<sub>E</sub>X.

Additional example documents are available from <http://tug.org/eplain/demo>.

## Summary

Eplain tries to make plain T<sub>E</sub>X a more user-friendly environment while at the same time allowing the user to retain complete control. Thus, especially if you are familiar with plain T<sub>E</sub>X, Eplain may be able to save you time and effort in typesetting anything from whole books where you need low-level control to short documents where you need just one or two document-level features.

We appreciate any and all bug reports, suggestions for enhancement, and offers of help; please write via the Eplain mailing list, <http://tug.org/mailman/listinfo/tex-eplain>.

## Acknowledgements

Thanks to Dave Walden, Steve Peter, Greg Black, and Barbara Beeton for reading drafts of this article and substantially improving it. Also thanks to Duane Bibby for the Eplain lion (named T<sub>E</sub>Ximilian; more details and downloadable images are on the Eplain home page), and to Duane and CTAN for the CTAN lion drawing.

## References

- [1] Berry, Karl, Oleg Katsitadze, and Steven Smith. Eplain.  
<http://www.ctan.org/tex-archive/macros/eplain>
- [2] Berry, Karl, Oleg Katsitadze, and Steven Smith. *Eplain: Expanded Plain T<sub>E</sub>X*, 1990–2005. <http://tug.org/eplain/doc/eplain> (online version)  
<http://www.lulu.com/content/113810> (paper version)
- [3] Bibby, Duane. CTAN lion drawing.  
[ftp://tug.ctan.org/ftpmaint/CTAN\\_lion/ctan\\_lion\\_350x350.png](ftp://tug.ctan.org/ftpmaint/CTAN_lion/ctan_lion_350x350.png)
- [4] Carlisle, David. Mini-L<sub>A</sub>T<sub>E</sub>X.  
<http://www.ctan.org/tex-archive/macros/plain/graphics/miniltx.tex>
- [5] Carlisle, David, et al. Graphics bundle.  
<http://www.ctan.org/tex-archive/macros/latex/required/graphics>
- [6] Walden, David. Travels in T<sub>E</sub>X Land: [...] plain T<sub>E</sub>X with Eplain, [...]. *The PracT<sub>E</sub>X Journal*, 2005-4.  
<http://tug.org/pracjourn/2005-4/walden>

◊ Karl Berry  
 88609 Wickizer Lane  
 Bandon, OR 97411  
 USA

◊ Oleg Katsitadze  
 24 Pushkin St., apt. 10  
 Simferopol 95011  
 Ukraine  
 eplain (at) tug.org

## Abstracts

This section contains abstracts from recent publications by other  $\text{\TeX}$  user groups, translated to English where needed. For a complete list of user group publications, see <http://tug.org/pubs.html>.

**Editor's note:** *TeXemplares* is the publication of CervantEX, the Spanish  $\text{\TeX}$  user group. Their web site is <http://www.cervantex.org>.

### ***TeXemplares #7, 2005***

JOAQUÍN ATAZ LÓPEZ, Creación de ficheros  $\text{\LaTeX}$  con GNU-Emacs [Creating  $\text{\LaTeX}$  files with GNU Emacs]; pp. 4–20

GNU Emacs is a very powerful text editor that certain extension packages equip with wide capabilities to deal with  $\text{\LaTeX}$ -type files. Among them are AUC- $\text{\TeX}$ , Ref $\text{\TeX}$ , and BIB $\text{\TeX}$ . [...] What follows is intended to offer a general overview, centered on AUC- $\text{\TeX}$ , because it is the most important package, and on tools for  $\text{\LaTeX}$ , because it is the best known  $\text{\TeX}$ -derived format. However, it should be clear that a large portion of what is said here about  $\text{\LaTeX}$  will also work with other formats derived from  $\text{\TeX}$ .

[Translation of author's introduction (edited)]

JAVIER BEZOS, *The  $\text{\LaTeX}$  Companion*, segunda edición [*The  $\text{\LaTeX}$  Companion*, 2<sup>nd</sup> edition]; pp. 21–23

Review of the second edition of *The  $\text{\LaTeX}$  Companion*.

[Compiled by Federico García]

---

### ***Zpravodaj 14(3-4)–15(2), 2004–2005***

---

**Editor's note:** *Zpravodaj* is the bulletin of  $\mathcal{C}\mathcal{S}\text{TUG}$ , the  $\text{\TeX}$  user group for the Czech and Slovak languages. Their web site is <http://www.cstug.cz>, and the *Zpravodaj* web site is <http://bulletin.cstug.cz>.

### ***Zpravodaj 14(3-4), 2004***

JAROMÍR KUBEN, Dopis nového předsedy  $\mathcal{C}\mathcal{S}\text{TUG}$  [Letter from the new  $\mathcal{C}\mathcal{S}\text{TUG}$  president]; p. 117–118

Zápis z valného shromáždění ze dne 27. 11. 2004 [Report from the  $\mathcal{C}\mathcal{S}\text{TUG}$  general assembly of 27 November 2004]; p. 119–121

Zpráva o činnosti  $\mathcal{C}\mathcal{S}\text{TUG}$  [Report on  $\mathcal{C}\mathcal{S}\text{TUG}$  activities]; p. 121–123

Zpráva o hospodaření za rok 2003 [Financial statement for 2003]; p. 123–124

JIŘÍ KOSEK, DocBook a generování rejstříků [DocBook and back-of-the-book indexes]; p. 125–135

This article summarizes problems that are related to indexes in DocBook. The introductory part describes how to mark up index entries in DocBook and how to then process the document. A new method for generating an internationalized index, respecting the rules of the Czech language, is then presented. At the end of the article, the possibility of including more than one index in a document is mentioned, together with an example of automatic index entry population.

PETR VOPÁLENSKÝ, PETR SOJKA, Multimediální publikování na DVD [Multimedia publishing on DVD]; p. 135–145

Publishing and distribution of multimedia data on DVD is increasingly common. However, for any large publishing project prefabricated solutions are insufficient—it is necessary to find the solution in each particular case.

The article discusses technologies, formats and methods selected and tested for a DVD created for the occasion of the tenth anniversary of the Faculty of Information of Masaryk University, with the name 10@FI. The reader will become acquainted with the method of DVD preparation since its conception and its development up to the creation of the (GNU/Linux) bootable DVD image and its production. The whole project was carried out mainly under the GNU/Linux operating system, using open source programs, with extensive usage of XML technologies and W3C standards (tens of collaborators, almost one hundred co-authors, thousands of linked files, hundreds of images and photos, more than ten minutes of original movies).

PETR OLŠÁK, Novinky v OFS [News in OFS]; p. 145–156

OFS (Olsak's font system) was previously presented at an SLT conference. Nevertheless, the OFS macros for plain  $\text{\TeX}$  were updated significantly during 2004. New features added: tools for on-line font catalogs, font tests including math fonts, improved possibilities of encoding-dependent macros, TX font support, etc. These new features are presented in this article.

PETR OLŠÁK, Projekt OkTeX [OkTeX project]; p. 156–171

OkTeX is a TeX format based on plainTeX and on packages OFS, LANG and IENC. It is an experiment of making a new language environment for plain TeX users, perhaps even more powerful than the well-known Babel package. The new package LANG cooperates with OFS and supports language switching, including declaration of arbitrary font encodings for each language. The IENC package is under development. It allows defining conversions from input encoding to font encoding, cooperating with the LANG package and with the encTeX extension (if the extension is available).

KAREL HORÁK, Jiné rodiny písem pro sazbu matematiky [Different font families for math typesetting]; p. 171–182

The aim of this contribution is to show possibilities that grow more extensive every day, thanks to the creative TeX community. Some time ago I was personally pleased by two well-designed collections of math fonts and characters for Times and Palatino, by Young Ryu. Then I was pleasantly surprised to find a well-designed Fourier collection supplementing the famous Utopia family during the preparation of this article.

PETR SOJKA, Slovenské vzory dělení slov: čas pro změnu? [Slovak hyphenation patterns: A time for change?]; p. 183–189

Word hyphenation, or the algorithmic segmentation of a large number of strings, is a problem tackled more often than it may appear on first sight. The freely available Slovak hyphenation patterns are based only on the definition of syllables, without coverage of many exceptions. We have collected and hyphenated more than one million Slovak word forms and generated new hyphenation patterns for Slovak with the program PatGen. New patterns cover all known exceptions. The result is usable not only in TeX distributions, but also in other systems as OpenOffice. We discuss bootstrapping and stratification techniques used in the patterns' development, and argue for much wider use of these techniques.

JAN PŘICHYSTAL, JIŘÍ RYBIČKA, Webové rozhraní pro sazbu dokumentů [Web interface for document typesetting]; p. 190–195

This article describes the impetus for, and features and facilities of the system ‘TeXonWeb’. This system makes for an easy introduction to the TeX typesetting system, and offers the possibility of creating high quality documents without the need to install TeX, using only a web browser.

MILAN ŠORM, Ligatura aneb začínáme s TeXem [Ligature, or beginning with TeX]; p. 195–200

I have found from my long-term experience with TeX that installation of a TeX distribution on the Windows system, with functioning Czech support, text editor, previewer and optional help or sample styles, is very difficult for most users. I have therefore decided to prepare a simple distribution for learning the principles of TeX, intended for beginners, students of our university and possibly others who need a high-quality typesetting system. The distribution consists of the minimal part of TeX Live needed for PDF generation (`pdftexlive`), spell checking (`ispell`), a freely distributable text editor of my own (designed for TeXing) and prefabricated styles. The aim of the contribution is to present the project, named Ligature, and find volunteers for further cooperative development.

ZDENĚK WAGNER, Skenujeme v Linuxu programem VueScan [Scanning in Linux with the VueScan program]; p. 201–211

The lecture presents the VueScan program (licensed as shareware; the Linux version is free of charge for personal use since 7.6.71). The basic program functions are summarized, and the workflow of using both film and desktop scanners, including scanners with transparent media adapters, is explained. A method of scanner calibration via a standard target is also described. At the end of the lecture, the program SCARSE for conversion of the scanned images from the RGB colorspace to CMYK for prepress is mentioned. The lecture also contains a brief comparison with other scanning programs.

ZDENĚK WAGNER, XML versus TeX, výhody a nevýhody [XML versus TeX, advantages and disadvantages]; p. 211–219

The lecture is a free continuation of the lecture on XML from the previous SLT. It compares features offered by both systems, and explains what is provided by XML mainly to TeXers. It presents thoughts on the cases in which direct preparation of text in TeX is suitable. It discusses methods how to generate files in other formats from both types of source documents and compares the results. The possibility of connection of TeX and XML with databases is also described.

### *Zpravodaj 15(1), 2005*

JAROMÍR KUBEN, Úvodník [Introduction]; p. 1–2

FRANTIŠEK CHVÁLA, O možnostech pdfTeXu [About pdfTeX possibilities]; p. 3–85

The article is intended for pdfTeX beginners. It describes a number of the pdfTeX primitives that

extend the original  $\text{\TeX}$  and presents examples illustrating how to use them for preparing a PDF document.

Primitiva pdf $\text{\TeX}$ U (syntaxe) [pdf $\text{\TeX}$  primitives (syntax)]; p. 86–89

This article reprints the pdf $\text{\TeX}$  documentation file `pdftex-syntax.txt`, including syntax of the new pdf $\text{\TeX}$  primitives. Syntax highlighting is done by ConTeXt via included code.

### *Zpravodaj 15(2-4), 2005*

JAROMÍR KUBEN, Úvodník [Introduction]; p. 93–93

Často kladené otázky o  $\text{\TeX}$ U a odpovědi na ně ( $\mathcal{C}$ TUG FAQ) [Frequently asked questions on  $\text{\TeX}$  and answers to them ( $\mathcal{C}$ TUG FAQ)]; p. 94–331

This special issue contains the translation of the document *Frequently asked questions on  $\text{\TeX}$  and answers to them* to the Czech and Slovak languages. The text was supplemented with explanations of specific problems concerning Czech and Slovak typesetting, microtypographical extensions of pdf $\text{\TeX}$ , usage of commercial Czech and Slovak fonts provided by Storm Type Foundry, and information on how to typeset Hindi and Sanskrit text in the Devanāgarī script.

[Received from Zdeněk Wagner]

---

### *Les Cahiers GUTenberg* Contents of double issue 44–45 (November 2004)

---

Editor's note: *Les Cahiers GUTenberg* is the journal of GUT, the French  $\text{\TeX}$  user group. Their web site is <http://www.gutenberg.eu.org>.

YANNIS HARALAMBOUS, Voyage au centre de  $\text{\TeX}$ : composition, paragraphage, césure [Voyage to the center of  $\text{\TeX}$ : Composition, paragraphs, hyphenation]; pp. 3–53

The author begins his “voyage” by asking why it is that, after all the care that Knuth took to make reading and learning about  $\text{\TeX}$  an enjoyable experience, using wit and charm, information and explanations in small readable chunks, all organised in a way that focusses on teaching rather than just following the order of the code itself—why is it that

so few people have really taken the plunge and followed/read Knuth, one of the truly great minds of the 20th century?

The author then offers to take the reader on a trip (!) down one particular path, all the way down to the very bowels of  $\text{\TeX}$ . Not another overview but a roll-up-your-sleeves encounter. The author proposes to study how a word (as viewed by  $\text{\TeX}$ ) is handled, from just being read by  $\text{\TeX}$ , to its trip through paragraphing and hyphenation, passing by such vast topics as macro expansion, insertions, math mode, and tables, just to name a few. The challenge is to give the reader a taste for more such adventures and a desire to discover hidden treasures along the way.

Perhaps a fanciful introduction, but who says talking about programming can't be playful!

[Summary of author's introductory paragraphs]

YVON HENEL, Comment commenter?

Commentaires et parties optionnelles [How to comment? Comments and optional bits]; pp. 54–82

How to position/place comments in a source file, how to deal with optional bits and obtain/yield/produce a multi-purpose document.

The packages `verbatim`, `comment`, `versions`, and `optional` are presented.

Examples of `docstrip` usage.

[Translation of author's abstract]

TWG-TDS WORKING GROUP, TDS: une structure de répertoires pour les fichiers  $\text{\TeX}$  [TDS: A directory structure for  $\text{\TeX}$  files]; pp. 83–114

This is the unofficial French translation of v.1.1 of the TDS document “A directory structure for  $\text{\TeX}$  files”, which originally appeared in English. This translation, by Jean-Côme Charpentier, is based on the one done originally in 1999 of v.0.9996 of the TDS, in collaboration with Vincent Vaquin.

The text is prefaced by a note from the translator (Charpentier), and concludes with a postface by Fabrice Popineau.

In his translator's preface, Charpentier provides the *raison d'être* for the TDS and its work, in the form of two of the questions that very quickly come to the fore when  $\text{\TeX}$  users begin to explore the possibilities of adding new stuff to their machines: “Where do I put X so that  $\text{\TeX}$  stops complaining that it can't find it?!”, followed very soon after by “Where do I put X so that my machine doesn't become some kind of infernal bazaar?!”

To address this, two things have to happen: first, all that stuff has to get organised in some kind of logical way so that users can find things; second, it would be good if everyone could be encouraged to organise things in the same way.

Not very lengthy, the TDS document also provides explanations as to why certain choices are made, and what the consequences are of making this or that choice. By the end, the reader has not only a sense of what the TDS is all about, but it also then forces one to think more consciously about what kinds of packages and such are being added, and where they would logically be best stored.

Charpentier concludes rather wryly that as long as the TDS documentation was only in English, French-speaking users could avoid dealing with the whole issue of organisation and structure. With this translation, they no longer have that excuse to hide behind.

\* \* \*

Where Charpentier's preface introduces the TDS document as it stood in 1999, Popineau's postface muses on its real-life implications from the perspective of 2004. The two texts therefore nicely bracket the TDS text itself. Fabrice Popineau, well known for having developed the *T<sub>E</sub>X* implementation for PCs called fp*T<sub>E</sub>X*, is a most appropriate person, then, to write an appreciation, as it were, of the TDS.

The postface begins by setting the context of the time, before the TDS came into existence, and brings the reader then through to the present day, where factors of great importance in the 1990s are less significant. Which leads to the question of just how useful and pertinent the TDS is today. While Popineau believes that, in its present form, it is much less critical than previously, his postface then proceeds to present several suggestions which would see more clearly defined (and thus more strict) requirements of packages, which would then make it possible to automatically manage the adding, updating, removing, and validating of packages.

[Summary of preface and postface]

GYÖNGYI BUJDOSO AND FERENC WETTL,  
Adapter *T<sub>E</sub>X* à la langue hongroise [On the  
Localization of *T<sub>E</sub>X* in Hungary]; p. 115

This paper deals with the present and future of the localization of *T<sub>E</sub>X* in Hungary. The authors review some of the necessary tools for preparing Hungarian documents, and especially the improvements needed to make *T<sub>E</sub>X* more usable in Hungary. Some of the work has been done, and a short 'to do' list will be presented for work to be done in the near future. The problems stemming from the specialities of Hungarian grammar (e.g., hyphenation, handling definite articles and suffixes) will be considered as well as the tasks implied by the heritage of Hungarian typography (e.g., page layout).<sup>1</sup>

[Author's abstract (edited)]

[Compiled by Christina Thiele]

<sup>1</sup> This text originally appeared in English in the 2002 TUG proceedings (*TUGboat*, 23(1), pp. 21–26). The French translation was done by Jean-Michel Huffen.

## MAPS 32, Spring 2005

MAPS is the publication of NTG, the Dutch language *T<sub>E</sub>X* user group. Their web site is <http://www.ntg.nl>.

WYBO DEKKER, Redactieel [From the editor];  
pp. 1–2

Overview of the issue's contents and errata.

PIET VAN OOSTRUM, Een uittreksel uit de recente bijdragen in het CTAN archief [A selection from the recent contributions to the CTAN archive];  
pp. 3–5

This article describes a number of recent contributions to the CTAN archive. The selection is based on what I find interesting and what I think others will find interesting. It is thus a personal choice. There is no intention of giving a complete overview. Consider this a kind of menu to whet the appetite of the curious.

[Translation of author's abstract]

CTAN TEAM, CTAN plans; pp. 6–8

The readers of *TUGboat* likely know the Comprehensive *T<sub>E</sub>X* Archive Network as a great pile of *T<sub>E</sub>X* stuff. That is, it is full of *T<sub>E</sub>X* materials, and it is great, but it is also something of a pile—a bit of a mess. We will sketch some plans for improving CTAN. As part of that, we will outline its architecture, history, and some present issues.

[Authors's abstract]

METAPOST TEAM, METAPOST developments—  
Spring 2005; pp. 9–13

The METAPOST development team is pleased to announce version 0.9 of METAPOST. This article documents the changes since the previous version, and provides a roadmap for future development.

[Authors's abstract]

ADAM T. LINDSAY, Font Variants; pp. 14–15

Font Variants are a new (meta-)feature in ConTeXt, offering the opportunity of easier access to advanced or unusual font features, without the hassle of using full typescript switches. This article briefly runs through the basic theory and practice of so-called font variants, and gives a few strategies for adapting it for your own uses.

[Author's abstract]

ADAM T. LINDSAY, Euler in use; pp. 16–25

The Euler math font was designed by Hermann Zapf. ConTeXt support was limited until now. We

show how to use the Euler-VM L<sup>A</sup>T<sub>E</sub>X package in combination with some new math definitions and type-scripts to give a more informal look to your equations.

[Author's abstract]

TACO HOEKWATER, Lettrines for ConTeXt; pp. 26–28

The ConTeXt module *lettri* is a port of the L<sup>A</sup>T<sub>E</sub>X package *letrine* by Daniel Flipo that provides a way to typeset dropped capitals at the beginning of paragraphs.

[Author's abstract]

STEVE PETER, T<sub>E</sub>X and linguistics; pp. 29–34

T<sub>E</sub>X has long been associated with mathematics and “hard” sciences such as physics. But even during the early days of T<sub>E</sub>X, linguists were attracted to the system, and today a growing number of them are turning to T<sub>E</sub>X (L<sup>A</sup>T<sub>E</sub>X, ConTeXt). Aside from the general advantages of T<sub>E</sub>X for producing academic papers, it offers linguists largely intuitive means for dealing with often complex notational issues. In this paper, an abbreviated version of my Practical T<sub>E</sub>X 2004 talk, I show some notational issues and their solutions in T<sub>E</sub>X. [Author's abstract]

TACO HOEKWATER, Controlling Acrobat Reader under X11; p. 35

The command-line programs *pdfopen* and *pdfclose* allow you to control the X Window System version of Adobe Acrobat Reader from the command line or from within a script. [Author's abstract]

OSCAR BOOT and FRANS ABSIL, Met XMLvan database naar L<sup>A</sup>T<sub>E</sub>X [From database to L<sup>A</sup>T<sub>E</sub>X via XML]; pp. 36–43

In this article we discuss the automated production of the technical science bachelor's degree program book at the Higher Defense Institute (HDO in Dutch). An important aspect is the conversion of Microsoft access database to L<sup>A</sup>T<sub>E</sub>X tables and appendices in the final documentation: XML files are used as an intermediate step.

[Translation of authors's abstract]

HENDRI HONDORP, Bundeling van conferentieverlagen [Creation of conference proceedings]; pp. 44–49

This article describes how proceedings for a conference or workshop can be produced with pdfL<sup>A</sup>T<sub>E</sub>X and the packages *pdfpages*, *fancyhdr*, and *hyperref*.

[Translation of author's abstract]

FRANS GODDIJN and HENDRI ADRIAENS, ...three, two, one . . . ; pp. 50–51

This article briefly describes the seemingly trivial task of numbering items in a list with decreas-

ing numbers, starting with the number equal to the amount of items in the list. [Author's abstract]

KAREL H. WESSELING, Compiling METAPOST figures under ConTeXt; pp. 52–55

To teach yourself METAPOST, the book “Learning METAPOST by Doing” by André Heck is probably unsurpassed. However, the examples therein are processed on Unix using L<sup>A</sup>T<sub>E</sub>X. ConTeXt users have a bit of detective work to do before they can have successful compilations. If you are new to ConTeXt, the lines below may help save your a few hours of experimenting. These instructions were extracted from the MetaFun manual by Hans Hagen (from chapters 1, 2 and 3), and from a small macro that he once gave me that makes it possible to use the *graph* package by John Hobby.

[Author's abstract]

ANDRÉ HECK, Learning METAPOST by doing; pp. 56–117

This course is only meant as a short, hands-on introduction to METAPOST for newcomers who want to produce rather simple graphics. The main objective is to get students started with METAPOST on a Unix platform. [Author's abstract]

JANUSZ NOWACKI, Antykwa Toruńska; pp. 118–132

Presentation of the Antykwa Toruńska typeface designed by Zygfryd Gardzielewski and digitized by GUST, the Polish T<sub>E</sub>X Users Group.

ERNST VAN DER STORM, Variabele faxdocumenten aanmaken in L<sup>A</sup>T<sub>E</sub>X [Generating variable fax documents with L<sup>A</sup>T<sub>E</sub>X]; pp. 133–137

I describe how L<sup>A</sup>T<sub>E</sub>X is used as a link between an existing company application and a fax server; data is exported automatically to a L<sup>A</sup>T<sub>E</sub>X document.

[Translation of author's abstract]

ERNST VAN DER STORM, Stroomdiagrammen maken met flow [Making flow charts with flow]; pp. 138–140

flow is a handy program to make flow charts; by means of \write18, it can be called from within a L<sup>A</sup>T<sub>E</sub>X document so that possible modifications appear automatically.

[Translation of author's abstract]

TACO HOEKWATER, Verslag EuroT<sub>E</sub>X 2005 [Report on EuroT<sub>E</sub>X 2005]; pp. 141–148

A report on the 16th annual EuroT<sub>E</sub>X conference, held in Pont-à-Mousson, March 7–11, 2005.

[Translation of author's abstract]

[Compiled by Steve Peter]

---

### **The PracTEX Journal 2005-1–2005-4**

*The PracTEX Journal* is an online publication of the TeX Users Group. Its web site is <http://tug.org/pracjourn>. All articles are available there.

### **The PracTEX Journal 2005-1, 2005-01-15**

LANCE CARNES, Welcome

STEVE GRATHWOHL, Invitation to PracTEX'05

LANCE CARNES, Highlights of the PracTEX'04 conference

JIM HEFFERON, CTAN for Starters

Newcomers to TeX can have trouble finding their way around. As with many other community-supported projects, beginners can get the sense that only insiders or old-timers can get the tool to do its magic. One of the secrets to TeX success is to know where to find resources on the Internet that you can use. This article guides you through finding and using resources from the TeX community's archive, the Comprehensive TeX Archive Network (CTAN).

DAVID ALLEN, Screen presentations, manuscripts, and posters from the same L<sup>A</sup>T<sub>E</sub>X source

This paper describes how to format the same document in any of three different styles: screen, manuscript and poster. The screen style is used to format output for a computer screen or LCD projector presentation, the manuscript package is used for printed publication, and the poster package will generate a conference poster presentation. These styles are used with the L<sup>A</sup>T<sub>E</sub>X article class. This paper gives the implementation details of the three styles, and also shows examples of their usage.

JENNY LEVINE, Label replacement in graphics

In this paper I show how graphics are manipulated to our (*Duke Mathematical Journal*) style. I give some examples and a step-by-step approach to assessing a figure file, removing its labels, and placing new ones using *graphicx* and *overpic*.

This is done to maintain a consistent style. Our labels should be in a compatible font and should match the look of the journal as well as that of the article (size, placement, emphasis, etc.).

DOUGLAS WAUD AND TIM NULL,  
\begin{here}{}% getting started

This is the first in a series of columns designed for the beginner or non-expert TeX or L<sup>A</sup>T<sub>E</sub>X user. This first column is divided into two main parts: the installation of TeX on your computer, and creating a simple L<sup>A</sup>T<sub>E</sub>X document. The installation section gives an overview of installing TeX on a

computer system, and provides a list of detailed resources for installation on different computers and operating systems. The section on creating a sample L<sup>A</sup>T<sub>E</sub>X document shows the basic elements of a L<sup>A</sup>T<sub>E</sub>X source file and guides the reader step-by-step in creating a document.

### STEVE PETER, \starttext % Practical ConTeXt

This column will introduce the use of ConTeXt, a powerful document creation system. In this first installment the reader will learn how to obtain a working ConTeXt system, how to write a simple "Hello, World" document, and how to adjust to the position of page numbers in a document. Following this is a demonstration of some unique features of ConTeXt, and finally a list of places to find more information.

DAVID WALDEN, Travels in TeX Land: Tweaking L<sup>A</sup>T<sub>E</sub>X

The author uses L<sup>A</sup>T<sub>E</sub>X extensively, but is not an expert by any means. His work requires that L<sup>A</sup>T<sub>E</sub>X do things differently than it does "out of the box". He may need new capabilities that do not already exist in L<sup>A</sup>T<sub>E</sub>X, to modify slightly existing L<sup>A</sup>T<sub>E</sub>X capabilities, or to give L<sup>A</sup>T<sub>E</sub>X a different look and feel.

This note sketches some of what the author would have liked to have found sketched in one place (rather than having to hunt in books or on the Web) when he was first trying to tweak L<sup>A</sup>T<sub>E</sub>X to do different things: finding an appropriate package or class, creating a new command or environment to do something new, creating a new command or environment to do something differently than L<sup>A</sup>T<sub>E</sub>X already does it, and changing an existing command or environment.

THE EDITORS, Ask Nelly:

- What is ConTeXt?
- What is L<sup>A</sup>T<sub>E</sub>X3?

PETER FLYNN, In my opinion: L<sup>A</sup>T<sub>E</sub>X myths and realities

### **The PracTEX Journal 2005-2, 2005-04-15**

LANCE CARNES, From the Editor

FROM READERS, Feedback

CHRISTINA THIELE, News from Around

This new section is about news from other user groups, news about TeX, and anything else that seems 'new'.

This first column is modelled on the old "News from Around", which appeared in *TTN* (*TeX and*

*TUG News*), a TUG newsletter (hardcopy) which circulated from 1991 till 1995.

There are two main sections: User Group News and *TeX* News. Under User Group News, there is a report on Greek *TeX* users and on the TUG 2004 conference in Greece; news from the Icelandic *TeX* users, who recently launched their local newsletter; a report from the Polish *TeX* users on their upcoming annual Bacho*TeX* conference; and an update from the UK *TeX* users. There is a report from the international *TeX* Users Group on elections, on the upcoming conferences in North Carolina and China, and on two new projects: the Interview Corner and the TUG Heritage Project.

*TeX* News includes items on Y&Y Inc., the *L<sup>A</sup>TeX* 3 project, and accessing CTAN more easily.

STEVE GRATHWOHL, Invitation to Prac*TeX*'05

PETER L. FLOM, A *L<sup>A</sup>TeX* Fledgling Struggles to Take Flight

I'm writing with two groups in mind: Beginners, and people who write for beginners. I'd like to offer both groups some perspective from someone who is just a little way along the path. I'd like to let the true beginners know that it is possible to learn *L<sup>A</sup>TeX*; after only a few months of intermittent use, I can do a lot—I have written entire articles in *L<sup>A</sup>TeX*, some of them with quite complicated organizational structure and with fairly intimidating formulas; I've also started doing some presentations in *L<sup>A</sup>TeX*, using the *beamer* package. If I can do it, you can. I'd like to give the teachers the perspective of a recent beginner, so that their efforts can have maximum reward; when I consider that so many people contribute to *L<sup>A</sup>TeX*, often without any monetary reward, I imagine that those people would like to have their efforts help as many people as possible to use *L<sup>A</sup>TeX* easily and well.

A. SCHREMMER, So, you are running Mac OS X and want to try *L<sup>A</sup>TeX*

This brief guide explains how to install and use the TeXShop (*L<sup>A</sup>TeX*) system on Mac OS X. If you follow the step-by-step installation instructions you should have the system running in short order. The second part of the article leads the reader through composing and typesetting a first *L<sup>A</sup>TeX* document.

WILL ROBERTSON, Square cells: an array cooking lesson

In this article, various features of the array package are described and used to create an environment that typesets tabular material with exactly square-shaped cells (useful in showing, for example, magic squares of numbers in which all rows, col-

umns and diagonals sum to an equal value). Along the way, some intermediate-level concepts in *L<sup>A</sup>TeX* programming are also shown. This article should be of interest to anyone who would like to read about the process of constructing new macros in *L<sup>A</sup>TeX*.

THOMAS A. SCHMITZ, Integrating TrueType Fonts into Con*TeX*t

*TeX* has a reputation for being difficult when it comes to font management. Many people (mainly those who haven't used any flavor of *TeX* in a long time) still think that only Computer Modern is available for typesetting in *TeX*, and there is a consistent rumor that integrating fonts is terribly difficult. While it involves a lot of steps, most of it is handled by automated tools and can be done even by inexperienced users. This tutorial will give step-by-step instructions on how to integrate TrueType fonts with your Con*TeX*t-installation.

ADAM T. LINDSAY, OpenType installation basics for Con*TeX*t

This article examines the basic steps necessary for OpenType font installation, with a focus on Con*TeX*t-oriented tools. Along the way, I will give overviews of the general font installation workflow and of the *TeX* font font installation script. The article assumes a fair amount of confidence at the command line and a properly-configured Con*TeX*t installation.

D.P. STORY, Creating Online Tests with eqExam

Have you ever wanted a simple way of creating an online test or quiz for your students? This paper describes one approach I have taken that uses a *L<sup>A</sup>TeX*-to-PDF workflow. This method may also be used for surveys and other information gathering. For an example of an eqExam online survey see the *TeX/L<sup>A</sup>TeX* survey at [http://www.math.uakron.edu/~dpstory/eqExam/tex\\_survey.pdf](http://www.math.uakron.edu/~dpstory/eqExam/tex_survey.pdf).

TRISTAN MILLER, Producing beautiful slides with *L<sup>A</sup>TeX*: An introduction to the HA-prosper package

This paper presents HA-Prosper, a *L<sup>A</sup>TeX* package for creating overhead slides. The features of the package and some examples of their use are described. The author also discusses advantages to producing slides with *L<sup>A</sup>TeX* versus the presentation software typically bundled with today's office suites.

PETER FLYNN, What does XML give the *L<sup>A</sup>TeX* user?

*L<sup>A</sup>TeX* users are being faced with the suggestion that they learn Yet Another Markup Language. Where does it all end? If *L<sup>A</sup>TeX* is so smart, why bother? This article is intended to provide some guidance for the confused.

D.P. STORY, Producing a  $\text{\TeX}/\text{\LaTeX}$  Online Survey with the eqExam Package

On January 31, 2005, I published a  $\text{\TeX}/\text{\LaTeX}$  online survey at [http://www.math.uakron.edu/~dpstory/eqExam/tex\\_survey.pdf](http://www.math.uakron.edu/~dpstory/eqExam/tex_survey.pdf) and invited the  $\text{\TeX}$  community, through the `comp.text.tex` forum, to participate in this unofficial survey of  $\text{\TeX}$  usage. The survey was created from a  $\text{\LaTeX}$  source, the final document was in Adobe's Portable Document Format (PDF). The purpose of this article is to describe how the survey was created and to report on some of the results.

TIM NULL, \begin{here} % getting started: A  $\text{\LaTeX}$  Survivor's Guide

This is the second in a series of columns on the preparation of a simple and short  $\text{\LaTeX}$  article. The main topic of discussion is techniques for avoiding and resolving  $\text{\LaTeX}$  errors. It is proposed that working to minimize risk is a good strategy for new  $\text{\LaTeX}$  users. Techniques for reducing risk are offered. The topic for the simple example article will be introduced, and the topic will be related to the philosophy of risk minimization. The information presented in the first begin-here column is reviewed in an included appendix. This material is re-presented in a different and, *hopefully*, clearer format.

STEVE PETER, \starttext % Practical ConTeXt: ConTeXt Text Editors

This column looks at some text editors and  $\text{\TeX}$  editing environments that can make life easier for ConTeXt users. Many text editors have some sort of  $\text{\TeX}$  support built in, and with a bit of coaxing they can be made to play nicely with ConTeXt. This column is a (partial) answer to frequent questions on the ConTeXt mailing list on how to make various  $\text{\LaTeX}$ -oriented text editors usable for ConTeXt.

DAVID WALDEN, Travels in  $\text{\TeX}$  Land: Choosing a  $\text{\TeX}$  Environment for Windows

This column recounts my experiences looking at and thinking about different ways  $\text{\TeX}$  is set up for users to go through the document-composition to typesetting cycle (input and edit, compile, and view and print). First I'll describe my own experience randomly trying various  $\text{\TeX}$  environments. I suspect that some other users have had a similar introduction to  $\text{\TeX}$ ; and perhaps other users have just used the environment that was available at their workplace or school. Then I'll consider some categories for thinking about options in  $\text{\TeX}$  setups. Last, I'll suggest some follow-on steps.

Since I use Microsoft Windows as my computer operating system, this note focuses on environments that are available for Windows.

THE EDITORS, Ask Nelly:

- What do I do with the 2 CDs and 1 DVD from TUG?
- How can I condense math matrices?
- Please explain the different  $\text{\TeX}$  font formats?
- How can I make PowerPoint slides with  $\text{\LaTeX}$ ?

ARTHUR OGAWA, In my opinion:  $\text{\TeX}$ 's Interface Challenges

Despite its well-known advantages for producing fine documents,  $\text{\TeX}$  has five areas with significant usability challenges: document creation, formatting specification, document preview, software installation, and integration with the host system resources. Through improved performance at these interfaces,  $\text{\TeX}$  can become more useful to the average computer user and, ultimately, more popular. After discussing these challenges and needed improvements, I mention certain activities we may undertake right away to prepare for the development of a more user-friendly  $\text{\TeX}$ , and I encourage and welcome further dialog on these issues.

**The Prac $\text{\TeX}$  Journal 2005-3, 2005-07-15**

LANCE CARNES, From the Editor

FROM READERS, Feedback

CHRISTINA THIELE, News from Around

This section is about news from other user groups, news about  $\text{\TeX}$ , and anything else that seems 'new'.

Under User Group News, there is a report on the Italian  $\text{\TeX}$  users and their upcoming conference in Pisa this October. There is a photo report on the Polish User Group's recent Bacho $\text{\TeX}$  conference.

From the international  $\text{\TeX}$  Users Group there are several items: the results of the recent election for TUG president, a summary of the members meeting that took place at the Practical  $\text{\TeX}$  2005 conference in North Carolina in June, the upcoming TUG 2005 conference in Wuhan, China, and updates on TUGboat publishing policies and the  $\text{\TeX}$  Live software release.

COMPILED BY PETER FLOM AND TRISTAN MILLER, Impressions from Prac $\text{\TeX}$ 05

The recent Prac $\text{\TeX}$  2005 conference in Chapel Hill, North Carolina was a great success thanks to the presenters, the attendees, and the local organizer, Steve Grathwohl of Duke University Press. See the post-conference web page at <http://www.tug.org/practex05/>.

[tug.org/practicaltex2005/](http://tug.org/practicaltex2005/) information for more photos and the conference preprints. Several papers from the conference are included in this issue of *The PracTEX Journal*.

#### JOE HOGG, Making a Booklet

This paper describes how a 36-page booklet, *A Botanical Tour of the Los Angeles Zoo and Botanical Gardens* was produced by the Zoo's Docent Botany Committee. The emphasis is on project and typesetting goals and using L<sup>A</sup>T<sub>E</sub>X to achieve them. This was our first typesetting project and, as such, it should be of interest to other T<sub>E</sub>X novices.

#### KLAUS HOEPPNER, Strategies for including graphics in L<sup>A</sup>T<sub>E</sub>X documents

This article presents strategies for including graphics into L<sup>A</sup>T<sub>E</sub>X documents. It shows the usage of the standard **graphics** packages of L<sup>A</sup>T<sub>E</sub>X as well as an introduction to different graphics formats. Some external tools for converting graphics formats are discussed.

#### PETER FLYNN, A categorized search of CTAN

The search functionality accessible through the search page of the Comprehensive T<sub>E</sub>X Archive Network (CTAN) allow you to search in three places: *a)* the CTAN directory structure and its filenames; *b)* Google; or *c)* the Graham Williams catalogue. While each has its advantages, they have a tendency to provide too much information. An new interface to *(a)* is being tested, which shows only direct matches, and categorises the output into different types of files.

#### JON BREITENBUCHER, L<sup>A</sup>T<sub>E</sub>X at a liberal arts college

Does L<sup>A</sup>T<sub>E</sub>X have a place in a liberal arts education? Yes, and in this article I present my reasons for introducing L<sup>A</sup>T<sub>E</sub>X in an undergraduate liberal arts setting. I also present how I introduced L<sup>A</sup>T<sub>E</sub>X, issues that were encountered, and what students and faculty think the impact has been.

#### JOHN BURT, Using **poemscol** for Critical Editions of Poetry

Critical editions are special versions of literary, legal, or historical texts in which the editors have attempted to reconstruct the text as the author wrote it. Editors of critical editions examine such things as the author's manuscripts, the publisher's galley, or other published editions of the same work to gather evidence to make arguments for the readings they propose, sometimes engaging in painstaking literary detective work. Critical editions have special typographical features which make them especially suited for T<sub>E</sub>X. **poemscol** is a critical edition package

for L<sup>A</sup>T<sub>E</sub>X designed around the special requirements of critical editions of poetic texts.

#### D. W. IGNAT, Word to L<sup>A</sup>T<sub>E</sub>X for a Large, Multi-Author Scientific Paper

The scientific journal *Nuclear Fusion* received a manuscript for a large review article in many sections, each formatted in MS Word. The journal's policy for reviews required a translation to L<sup>A</sup>T<sub>E</sub>X, including the transformation of section-based references to a non-repetitive article-based list. Saving Word files in RTF format and using **rtf2latex2e** accomplished the basic translation, and then a perl program was used to get the references into acceptable condition. This approach to conversion succeeded and may be useful to others.

#### PETER FLOM, HANS HAGEN, JOE HOGG, NICOLA TALBOT, PHILIP TAYLOR, CHRISTINA THIELE AND DAVID WALDEN, What is T<sub>E</sub>X?

#### TIM NULL, \begin{here} % getting started: Topic #1: Creating my first L<sup>A</sup>T<sub>E</sub>X article, Part 3

This a continuation of a series on the preparation of a simple and short L<sup>A</sup>T<sub>E</sub>X article. It is the third installment in the series. This column will introduce two sections of a L<sup>A</sup>T<sub>E</sub>X document: the *Preamble* and *Title Page*. Part II of this column offers several contests/exercises to hone your L<sup>A</sup>T<sub>E</sub>X skills.

#### DAVID WALDEN, Travels in T<sub>E</sub>X Land: A Macro, Three Software Packages, and the Trouble with T<sub>E</sub>X

In this column in each issue I muse on my wanderings around the T<sub>E</sub>X world. In this issue, I deal with three *unrelated* topics: I describe how a small macro works that I decided to try to understand, I briefly describe my experiments with three T<sub>E</sub>X-related software packages, and I give my perspective on why lots of people find T<sub>E</sub>X difficult.

#### THE EDITORS, Ask Nelly:

- What is Lyx?
- What are class and style files?

#### THE EDITORS, Distractions: A Pitfalls Contest and a Web Treasure Hunt — prizes awarded!

#### *The PracTEX Journal 2005-4, 2005-11-07*

#### LANCE CARNES, From the Editor: In this issue; Next issue: Fonts

#### FROM READERS, Feedback

PETER FLOM, L<sup>A</sup>T<sub>E</sub>X for academics and researchers who (think they) don't need it

This paper is written for academics and researchers who don't use L<sup>A</sup>T<sub>E</sub>X and wonder why anyone does. People who *do* use L<sup>A</sup>T<sub>E</sub>X (probably all of the readers of the article in this journal) may wish to share the article with their colleagues.

JIM HEFFERON, Minutes in Less Than Hours: Using L<sup>A</sup>T<sub>E</sub>X Resources

In this article, building a new L<sup>A</sup>T<sub>E</sub>X document class is illustrated by developing a class for minutes of meetings.

ANDREW MERTZ AND WILLIAM SLOUGH, Beamer by Example

There are a variety of L<sup>A</sup>T<sub>E</sub>X classes which can be used to produce "overhead slides" for presentations. One of these, beamer, provides flexible and powerful environments which can be used to create slides and PDF-based documents suitable for presentations. Although the class is extensively documented, first-time users may prefer learning about this class using a collection of graduated examples. The examples presented here cover a wide spectrum of use, from the simplest static slides to those with dynamic effects.

M.A. GURAVAGE, A Brochure

This paper shows how typesetting a commercial brochure can be done easily with ConTEXt.

STEVE PETER, \starttext % Swelled rules and MetaPost

This column looks at how to make a typographic character called a "swelled rule" using MetaPost and ConTEXt.

DAVID WALDEN, Travels in T<sub>E</sub>X Land: Word2T<sub>E</sub>X redux, T<sub>E</sub>X2Word, plain T<sub>E</sub>X and Eplain, and playing with "thought breaks"

In this column in each issue I muse on my wanderings around the T<sub>E</sub>X world. In the last issue, I described my trial of Word2T<sub>E</sub>X. In this issue, I first describe a little more of my experience with Word2T<sub>E</sub>X, second describe a trial of TeX2Word, third describe my efforts to try T<sub>E</sub>X itself (not L<sup>A</sup>T<sub>E</sub>X) for the first time, and fourth look at several ways to typographically show a change of subject.

THE EDITORS, Ask Nelly:

- How can I use hyperlinks in documents?
- How do I use the apa style?

THE EDITORS, Distractions: Sudoku; contest winners and answers

## Calendar

**2006**

Mar 1	<b>Deadline for submission of abstracts for EuroTEX 2006</b> (July 5–8). For information, visit <a href="http://matexhu.org/eurotex2006">http://matexhu.org/eurotex2006</a> .	Jun 5 – Jul 28	Rare Book School, University of Virginia, Charlottesville, Virginia. Many one-week courses on type, bookmaking, printing, and related topics. For information, visit <a href="http://www.virginia.edu/oldbooks">http://www.virginia.edu/oldbooks</a> .
Mar 6–10	Rare Book School, University of Virginia, Charlottesville, Virginia. One-week courses on bibliography and electronic texts. For information, visit <a href="http://www.virginia.edu/oldbooks">http://www.virginia.edu/oldbooks</a> .	Jun 30	<b>Deadline for submission of papers for TUG 2006</b> (November 9–11). For information, visit <a href="http://www.tug.org/tug2006/">http://www.tug.org/tug2006/</a> .
Mar 6–8	29 <sup>th</sup> Internationalization and Unicode Conference, San Francisco, California. One keynote will be by Charles Bigelow (of Bigelow & Holmes), on “The Effect of Unicode on Type Design”. For information, visit <a href="http://www.unicodeconference.org">http://www.unicodeconference.org</a> .	Jul 3–4	“Jobbing printing—the stuff of life”, joint conference of the Printing Historical Society and The Ephemera Society, University of Reading, UK. For information, visit <a href="http://www.printinghistoricalsociety.org.uk/events/">http://www.printinghistoricalsociety.org.uk/events/</a> .
Mar 8–10	DANTE 2006, 34 <sup>th</sup> meeting, Freie Universität Berlin, Germany. For information, visit <a href="http://www.dante.de/dante2006">http://www.dante.de/dante2006</a> .	Jul 5–8	16 <sup>th</sup> EuroTEX Conference, “A Hungarian TeX Rhapsody”, Debrecen, Hungary. For information, visit <a href="http://www.matexhu.org/eurotex2006">http://www.matexhu.org/eurotex2006</a> .
Apr 1	<b>Practical TeX 2006</b> (July 30–August 1) abstracts due. For information, visit <a href="http://www.tug.org/practicaltex2006">http://www.tug.org/practicaltex2006</a> .	Jul 5–9	ALLC/ACH-2005, Joint International Conference of the Association for Literary and Linguistic Computing and Association for Computers and the Humanities, “Digital Humanities 2006”, Université Paris-Sorbonne. For information, visit <a href="http://www.allc-ach2006.colloques.paris-sorbonne.fr/">http://www.allc-ach2006.colloques.paris-sorbonne.fr/</a> or the organization web site at <a href="http://www.ach.org">http://www.ach.org</a> .
Apr 5–8	Kitabat 2006: Arabic calligraphy and typography conference, Dubai. For information, visit <a href="http://www.kitabat.org">http://www.kitabat.org</a> .	Jul 11–14	SHARP Conference (Society for the History of Authorship, Reading and Publishing), “Trading Books—Trading Ideas”, The Hague & Leiden, Netherlands. For information, visit <a href="http://sharpweb.org">http://sharpweb.org</a> . In conjunction with the 400th anniversary of Rembrandt’s birth in Leiden; for information, visit <a href="http://www.rembrandt400.com">http://www.rembrandt400.com</a> .
Apr 29 – May 2	BachoTeX 2006, “There Is No Life Without Live TeX”, 14 <sup>th</sup> annual meeting of the Polish TeX Users’ Group, GUST, Bachotek, Brodnica Lake District, Poland. For information, visit <a href="http://www.gust.org.pl/BachoTeX/2006">http://www.gust.org.pl/BachoTeX/2006</a> .		
Jun 30	<b>Deadline for submission of proposals for TUG 2006</b> (November 9–11). For information, visit <a href="http://www.tug.org/tug2006">http://www.tug.org/tug2006</a> .		

*Status as of 1 February 2006*

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](mailto:office@tug.org)). For events sponsored by other organizations, please use the contact address provided.

An updated version of this calendar is online at <http://www.tug.org/calendar/>.

Additional type-related events are listed in the Typophile calendar, at

<http://www.icalx.com/html/typophile/month.php?cal=Typophile>.

**Practical TeX 2006**

Rutgers University, Busch Campus,  
Piscataway, New Jersey.

Jul 25–28	Pre-conference, hands-on L <sup>A</sup> T <sub>E</sub> X workshop.
Jul 30 – Aug 1	A user-oriented conference sponsored by TUG. For information, visit <a href="http://www.tug.org/practicaltex2006">http://www.tug.org/practicaltex2006</a> .
Jul 30 – Aug 3	SIGGRAPH 2006, Boston, Massachusetts. For information, visit <a href="http://www.siggraph.org/s2006/">http://www.siggraph.org/s2006/</a> .
Aug 9–13	TypeCon2006, “The Boston T Party”, Boston, Massachusetts. For information, visit <a href="http://www.typecon.com">http://www.typecon.com</a> .
Sep 1	<b>Practical TeX 2006</b> , papers due for <i>TUGboat</i> publication.
Sep 29–30	American Printing History Association conference, “The Atlantic World of Print in the Age of Franklin”, Philadelphia, Pennsylvania. For information, visit <a href="http://www.printinghistory.org/html/conference">http://www.printinghistory.org/html/conference</a> .
Oct 7	DIY (Do It Yourself) Book Festival, Los Angeles, California. For information, visit <a href="http://www.diyconvention.com/">http://www.diyconvention.com/</a> .
Oct 16–18	Fifth Annual St. Bride Conference, “Fast Type, Slow Type”, Birmingham, England. Celebrating the 300th birthday of John Baskerville. For information, visit <a href="http://www.creativepro.com/story/news/23853.html">http://www.creativepro.com/story/news/23853.html</a> .
Oct 20–22	The Fourth International Conference on the Book, “Save, Change or Discard: Tradition and Innovation in the World of Books”, Emerson College, Boston, Massachusetts. For information, visit <a href="http://book-conference.com">http://book-conference.com</a> .

**TUG 2006**

Digital Typography & Electronic Publishing;  
Localization & Internationalization,  
Marrakesh, Morocco.

Nov 7–8	Pre-conference tutorials.
Nov 9–11	The 27 <sup>th</sup> annual meeting of the TeX Users Group. For information, visit <a href="http://www.tug.org/tug2006">http://www.tug.org/tug2006</a> .
Nov 18	NTG 38 <sup>th</sup> meeting, Zeist, Netherlands. For information, visit <a href="http://www.ntg.nl/bijeen/bijeen38.html">http://www.ntg.nl/bijeen/bijeen38.html</a> .

**TUG Business****TUG financial statements for 2005**

Robin Laakso

This financial report for 2005 has been reviewed by the TUG board but has not been audited. It may change slightly when the final 2005 tax return is filed. TUG’s tax returns are publicly available on our web site: <http://www.tug.org/tax-exempt>.

**Revenue highlights**

TUG income decreased 10 percent for 2005 compared to 2004. Total membership dues were \$91.1K at the end of 2005, compared to \$101.6K in 2004. This represents a decline in membership of approximately 110—from a little more than 1600 TUG members in 2004 to just over 1500 in 2005. The joint membership with NTG increased slightly, while UK-TUG joint membership dropped slightly.

TUG had \$19.4K in income in 2005 from other sources than membership fees. Three areas of particular note:

- TUG store revenue of \$7.4K included sales of:
  - TeX CDs and DVDs;
  - discounted WinEdt licenses, through our arrangement with the WinEdt team;
  - discounted TeXnical books, through our arrangement with the Pearson Publishing Group (which includes Addison-Wesley);
  - the Lucida font collection, through our arrangement with Bigelow & Holmes (which started late in 2005).
- Contribution income from generous TUG members and individuals worldwide increased 6 percent from 2004 to 2005.
- Interest income was down 15 percent in 2005 compared to 2004, mostly due some of our reserves (held in a CD) being used to cover accrued liabilities; notably, paying for the *TUGboat* issues as we get back on schedule.

**Expense highlights**

Payroll and office expenses, software production and mailing, and *TUGboat* production and mailing continue to be the major expense items.

Payroll was down slightly in 2005 from 2004 (as it was from 2003 to 2004).

Software production and mailing was down 10 percent, from \$9K in 2004 to \$8K in 2005. The savings is partially due to excellent prices obtained

in Germany for copying both the DVD and CDs, and partially because fewer members in 2005 resulted in lower postage costs.

*TUGboat* production and mailing at \$18.6K in 2005 included three publications: the Practical TeX 2005 conference proceedings, the TUG 2005 (Wuhan, China) proceedings, and a regular issue to be published in early 2006 for which the estimated expense was accrued in 2005. Fewer pages in the first two issues, combined with smaller print runs, helped bring the cost down significantly in 2005.

Much of the \$2.1K for the “Postage/delivery—members” line item was individually mailing issues of *TUGboat* and software discs, as members join throughout the year.

In 2005, TUG made contributions of \$2,000 to the TUG Bursary, \$1,000 to EuroTeX 2005, \$500 for an Apple developer membership, and miscellaneous donations of \$950.

Netting Revenue, Cost of Goods Sold, and Expenses, TUG had a loss of \$2,986 for the year. This is a 27 percent smaller loss than in 2004.

However, there was an unexpected prior year adjustment of -\$9,784, shown near the bottom of the Profit and Loss comparison. This resulted entirely from the publication in 2005 of the 2004 EuroTeX conference proceedings. This 322 page issue cost TUG just under \$19K to produce and mail, which was over \$11K more than we had anticipated (and therefore accrued) at the end of 2004.

### Balance sheet highlights

We have accrued \$7,000 to produce and mail the last 2005 issue of *TUGboat* which is this present issue being mailed in early 2006. We expect this accrual to be close to the right amount.

The ‘committed donations’ come to TUG specifically designated for the LTeX project, the TeX Development fund, etc.; they have been committed accordingly and are disbursed as the projects progress.

The deferred conference donations came from DANTE in late 2005 for conferences in 2006. The deferred member income came from members who paid their 2006 dues in 2005.

The payroll liabilities are for 2005 state and federal taxes due January 15, 2006.

If you have any questions about TUG’s finances, or if you would like to help with any TUG-related activities, please contact the TUG office.

◊ Robin Laakso  
TUG Executive Director  
[office@tug.org](mailto:office@tug.org)

### TUG 12/31/2005 Balance Sheet

#### Assets

Checking/savings	\$115,994
Accounts receivable	\$635
Other current assets	\$728
Total current assets	<u>\$117,357</u>
Fixed assets	<u>\$5,591</u>
<b>Total assets</b>	<b><u>\$122,948</u></b>

#### Liabilities and Equity

##### Liabilities

Late <i>TUGboat</i> accrual	\$7,000
Committed donations	\$7,005
Deferred conf. donations	\$1,794
Deferred member income	\$1,160
Payroll liabilities	<u>\$1,037</u>
<b>Total liabilities</b>	<b><u>\$17,996</u></b>

##### Equity

Equity as of 1/1/2005	\$117,722
Net income for 2005	<u>-\$12,770</u>
<b>Total equity</b>	<b><u>\$104,952</u></b>

##### Total liabilities and equity

\$122,948

### TUG 2005 (versus 2004) Revenue and Expenses

	2005	2004
<b>Revenue</b>		
Membership dues	\$91,173	\$101,631
Product sales	\$7,410	\$8,259
General contributions	\$7,938	\$7,453
Conferences	\$182	-\$296
Interest	\$3,672	\$4,295
Advertising	\$200	\$950
Other		\$765
<b>Total revenue</b>	<b>\$110,575</b>	<b>\$123,057</b>
<b>Cost of Goods Sold</b>		
<i>TUGboat</i> prod/mail	\$18,626	\$26,242
Software prod/mail	\$8,092	\$8,962
Postage/delivery-members	\$4,874	\$5,111
Conf. expenses (TUG)	\$2,082	\$1,115
Member renewal		
Copy/printing - members	\$300	\$389
<b>Total COGS</b>	<b>\$33,974</b>	<b>\$41,819</b>
<b>Gross profit</b>	<b>\$76,601</b>	<b>\$81,238</b>
<b>Expenses</b>		
Contributions made by TUG	\$4,950	\$8,449
Office overhead	\$13,411	\$12,788
Payroll	\$59,066	\$59,768
Professional fees	\$119	\$2,016
Depreciation	\$2,041	\$2,305
<b>Total expenses</b>	<b>\$79,587</b>	<b>\$85,326</b>
<b>Net ordinary income</b>	<b>-\$2,986</b>	<b>-\$4,088</b>
<b>Prior year adjustment</b>	<b>-\$9,784</b>	<b>-\$260</b>
<b>Net profit</b>	<b>\$12,770</b>	<b>\$4,348</b>

## TeX Development Fund 2005 Report

Karl Berry and Kaja Christiansen

The TeX Development Fund was created by the TeX Users Group in 2003, under the aegis of the TUG Technical Council, to foster growth of TeX-related technical projects. This report lists the two projects funded since the last report, in *TUGboat* 25(2).

We remain most appreciative of the ongoing support from individuals, which have made the recent grants possible.

For application information, the complete list of projects, and more, please see the web site.

◊ Karl Berry and Kaja Christiansen  
<http://tug.org/tc/devfund/>

### 1 Inconsolata

Applicant: Raph Levien, USA, <http://levien.com/type/myfonts/inconsolata.html>.

Amount: US\$1000; acceptance date: 30 Nov 2005.

Completion and release of the Inconsolata font, a monospaced design. Only one style is anticipated (no bold or italic is planned). Glyph coverage is to include Latin 1, 2, and 9, with a few other glyphs useful for TeX to be added upon request.

Metrics and encoding will be tuned to make Inconsolata a possible drop-in replacement for cmtt.

### 2 Malayalam

Applicant: Alex A.J., India,

<http://sarovar.org/projects/malayalam>.

Amount: US\$600; acceptance date: 9 Nov 2005.

The project aims to develop a package for Malayalam typesetting using the Omega system.

## 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 maintains an online list of consultants at <http://tug.org/consultants.html>. If you'd like to be included either in print or online or both, please submit at <https://www.tug.org/consultants/listing.html>, or email us at [consult-admin@tug.org](mailto:consult-admin@tug.org). To place a larger ad in *TUGboat*, please see <http://tug.org/TUGboat/advertising.html>.

#### Kinch, Richard J.

7890 Pebble Beach Ct  
 Lake Worth, FL 33467  
 561-966-8400

Email: [kinch \(at\) truetex.com](mailto:kinch(at)truetex.com)

Publishes TrueTeX, a commercial implementation of TeX and L<sup>A</sup>T<sub>E</sub>X. Custom development for TeX-related software and fonts.

#### Martinez, Mercè Aicart

C/Tarragona 102 4<sup>o</sup> 2<sup>a</sup>  
 08015 Barcelona, Spain  
 +34 932267827

Email: [m.aicart \(at\) menta.net](mailto:m.aicart(at)menta.net)

Web: [http://www.edilatex.com/index\\_eng.html](http://www.edilatex.com/index_eng.html)

We provide, at reasonable low cost, TeX and L<sup>A</sup>T<sub>E</sub>X typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990.

#### MCR Inc.

731 Beta Drive #G  
 Mayfield Village, OH 44143  
 (440) 484-3010; fax: (440) 484-3020  
 Email: [sales \(at\) mcr-inc.com](mailto:sales(at)mcr-inc.com)  
 Web: [www.mcr-inc.com](http://www.mcr-inc.com)

Contract typesetting/printing services.

#### Ogawa, Arthur

40453 Cherokee Oaks Drive  
 Three Rivers, CA 93271-9743  
 (209) 561-4585  
 Email: [arthur.ogawa \(at\) teleport.com](mailto:arthur.ogawa(at)teleport.com)

Bookbuilding services, including design, copyedit, art, and composition; color is my speciality. Custom TeX macros and L<sup>A</sup>T<sub>E</sub>X2<sub>E</sub> document classes and packages. Instruction, support, and consultation for workgroups and authors. Application development in L<sup>A</sup>T<sub>E</sub>X, TeX, SGML, PostScript, Java, and C++. Database and corporate publishing. Extensive references.

#### Peter, Steve

310 Hana Road  
 Edison, NJ 08817  
 +1 (732) 287-5392  
 Email: [speter \(at\) dandy.net](mailto:speter(at)dandy.net)

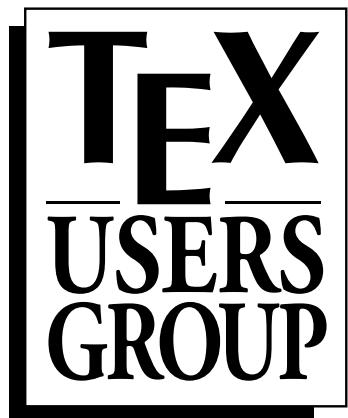
Specializing in foreign language, linguistic, and technical typesetting using TeX, L<sup>A</sup>T<sub>E</sub>X, and ConTeXt, I have typeset books for Oxford University Press, Routledge, and Kluwer, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful 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.

#### Veytsman, Boris

2239 Double Eagle Ct.  
 Reston, VA 20191  
 (703) 860-0013  
 Email: [borisv \(at\) lk.net](mailto:borisv(at)lk.net)  
 Web: <http://users.lk.net/~borisv>

TeX/L<sup>A</sup>T<sub>E</sub>X consulting. Integration with databases, full automated document preparation systems, conversions and more.

# TeX Users Group Membership Form 2006



Promoting the use  
of TeX throughout  
the world.

*mailing address:*

P. O. Box 2311  
Portland, OR 97208-2311 USA

*shipping address:*

1466 NW Naito PKWY, Suite 3141  
Portland, OR 97209-2820 USA

*phone:* +1 503-223-9994

*fax:* +1 206-203-3960

*email:* office@tug.org

*web:* http://www.tug.org

*President* Karl Berry

*Vice-President* Kaja Christiansen

*Treasurer* David Walden

*Secretary* Susan DeMerit

*Executive Director* Robin Laakso

TUG membership rates are listed below. Please check the appropriate boxes and mail the completed form with payment (in US dollars) to the mailing address at left. If paying by credit/debit card, you may alternatively fax the form to the number at left or join online at <http://tug.org/join.html>. The web page also provides more information than we have room for here.

**Status** (check one)  New member  Renewing member

**Automatic membership renewal** in future years

Using the same payment information; just contact office to cancel.

	Rate	Amount
<input type="checkbox"/> <b>Early bird membership</b> for 2006 After May 31, dues are \$85.	\$75	_____
<input type="checkbox"/> <b>Special membership</b> for 2006 You may join at this special rate (\$55 after May 31) if you are a senior (62+), student, new graduate, or from a country with a modest economy. Please circle accordingly.	\$45	_____
<input type="checkbox"/> <b>Subscription</b> for 2006 (non-voting)	\$95	_____
<input type="checkbox"/> <b>Institutional membership</b> for 2006 Includes up to eight individual memberships.	\$500	_____
<input type="checkbox"/> <b>Don't ship any physical benefits</b> ( <i>TUGboat</i> , software) <i>TUGboat</i> and software distributions are available electronically.	deduct \$20	_____
<input type="checkbox"/> <b>Send last year's TeX Collection 2005</b> right away <i>Instead of</i> this year's TeX Collection 2006.	n/a	_____
<input type="checkbox"/> <b>Send CTAN 2006 on CD</b> (shipped on DVD to everyone)	\$15	_____
<b>Purchase last year's materials</b>		
<input type="checkbox"/> <i>TUGboat</i> volume for 2005 (3 issues)	\$20	_____
<input type="checkbox"/> <b>TeX Collection 2005</b> 2 CD's & 1 DVD with proTeXt, MacTeX, TeX Live, CTAN.	\$20	_____
<input type="checkbox"/> CTAN 2005 CD-ROMS	\$15	_____
<b>Voluntary donations</b>		
<input type="checkbox"/> General TUG contribution	_____	_____
<input type="checkbox"/> Bursary Fund contribution	_____	_____
<input type="checkbox"/> TeX Development Fund contribution	_____	_____
<input type="checkbox"/> CTAN contribution	_____	_____
<input type="checkbox"/> L <sup>A</sup> T <sub>E</sub> X 3 contribution	_____	_____
<b>Total \$</b>		_____

*Tax deduction:* The membership fee less \$35 is generally deductible, at least in the US.

*Multi-year orders:* To join for more than one year at this year's rate, just multiply.

**Payment** (check one)  Payment enclosed  Visa/MasterCard/AmEx

Account Number: \_\_\_\_\_ Exp. date: \_\_\_\_\_

Signature: \_\_\_\_\_

**Privacy:** TUG uses your personal information only to send products, publications, notices, and (for voting members) official ballots. TUG does not sell or otherwise provide its membership list to anyone.

Electronic notices will generally reach you much earlier than printed ones. However, you may choose not to receive any email from TUG, if you prefer.

Do not send me any TUG notices via email.

Name \_\_\_\_\_

Department \_\_\_\_\_

Institution \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State/Province \_\_\_\_\_

Postal code \_\_\_\_\_ Country \_\_\_\_\_

Email address \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

Position \_\_\_\_\_ Affiliation \_\_\_\_\_

## Institutional Members

American Mathematical Society,  
*Providence, Rhode Island*

Banca d'Italia,  
*Roma, Italy*

Center for Computing Science,  
*Bowie, Maryland*

Certicom Corp.,  
*Mississauga, Ontario Canada*

CNRS - IDRIS,  
*Orsay, France*

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,  
*Washington/New Mexico, USA*

Masaryk University,  
Faculty of Informatics,  
*Brno, Czechoslovakia*

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

Princeton University,  
Department of Mathematics,  
*Princeton, New Jersey*

Springer-Verlag Heidelberg,  
*Heidelberg, Germany*

Stanford Linear Accelerator  
Center (SLAC),  
*Stanford, California*

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*

Uppsala University,  
*Uppsala, Sweden*

Vanderbilt University,  
*Nashville, Tennessee*

## Practical T<sub>E</sub>X 2006: L<sup>A</sup>T<sub>E</sub>X Workshop and Presentations on

### L<sup>A</sup>T<sub>E</sub>X, T<sub>E</sub>X, ConT<sub>E</sub>Xt, and more

L<sup>A</sup>T<sub>E</sub>X workshop: July 25–28, 2006

Practical T<sub>E</sub>X conference: July 30–August 1, 2006

Rutgers, the State University (Busch Campus)  
Piscataway, New Jersey, USA

<http://tug.org/practicaltex2006>

conferences@tug.org

Keynote address: *Barbara Beeton*,  
American Mathematical Society & T<sub>E</sub>X Users Group

- April 1, 2006 - presentation proposal deadline
- April 15, 2006 - early bird registration deadline
- July 14, 2006 - hotel reservation deadline

Hope to see you there!



Sponsored by the T<sub>E</sub>X Users Group.



## EuroTEX 2006: A Hungarian T<sub>E</sub>X Rhapsody

### Announcement and Call for Papers

The 16<sup>th</sup> EuroTEX meeting, “A Hungarian T<sub>E</sub>X Rhapsody”, will be held in Debrecen, Hungary, between July 5 and 8, 2006. MAT<sub>E</sub>X (the Hungarian T<sub>E</sub>X User Group) together with the University of Debrecen have committed to undertake the conference affairs, and now announce the call for papers. This will be the first international T<sub>E</sub>X conference held in Hungary.

For more information about EuroTEX 2006, please visit <http://www.matexhu.org>.

#### Dates

- March 1, 2006—Deadline for abstracts of presentations;  
e-mail: [eurotex2006@matexhu.org](mailto:eurotex2006@matexhu.org).
- June 1, 2006—Deadline for preprints of papers, for distribution at the conference.
- July 5–8, 2006—Conference.
- July 25, 2006—Deadline for final versions of papers; the proceedings will be published as an issue of *TUGboat*.

#### Topics

Topics include but are not limited to:

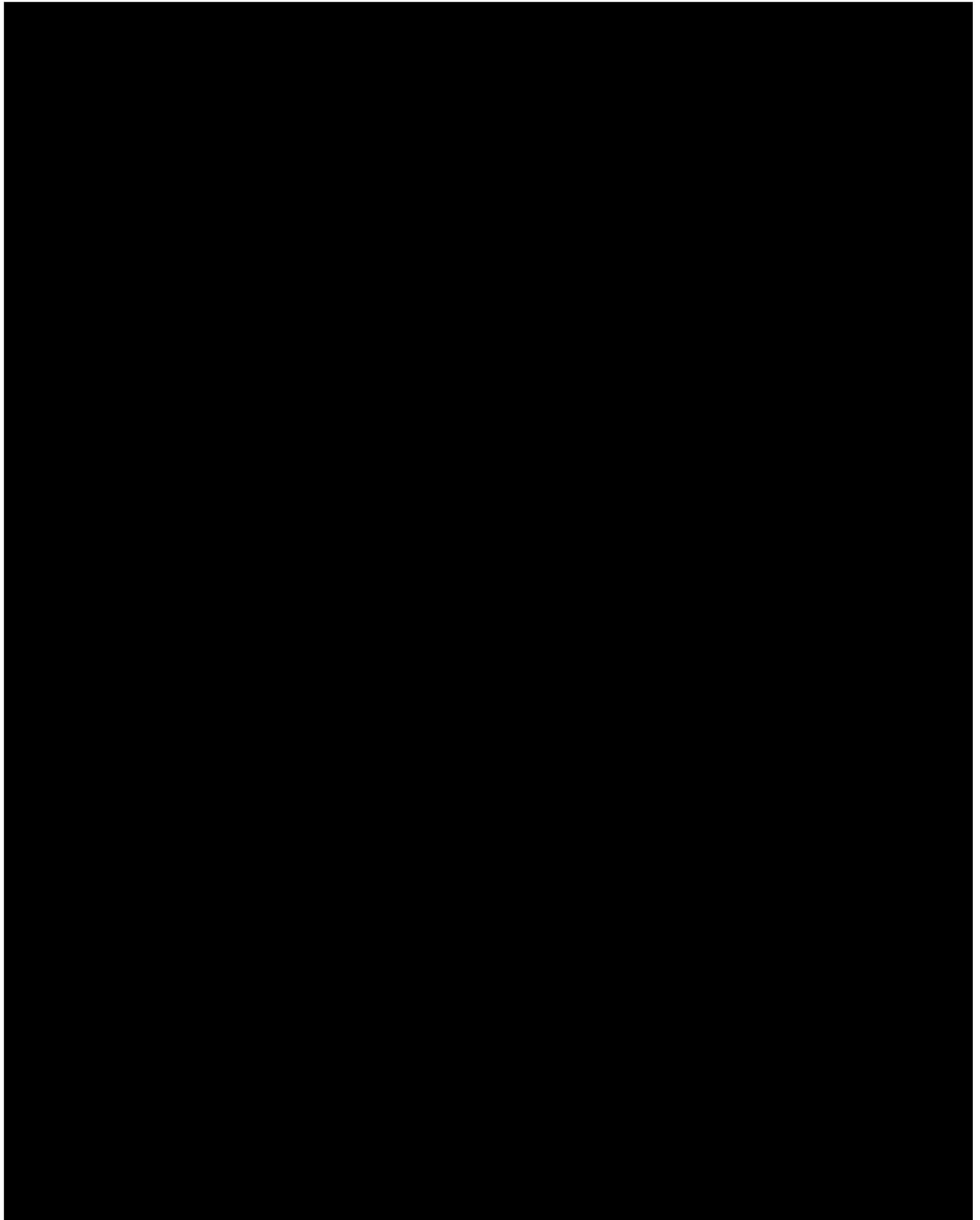
- T<sub>E</sub>X and so many friends, for automated typesetting
- Typography (digital or otherwise)
- Font design and technologies
- Publishing (electronic or otherwise)
- (Re)discovery of the European book tradition

#### Location

The place of the conference is Debrecen, Hungary. Debrecen is a town of universities known as the Calvinist Rome. It is near the biggest Hungarian National Park, Hortobágy, and a famous spa in Hajdúszoboszló.

There will be also free time to give you the opportunity to taste the many types of Hungarian wines, and get to know the tasty special Hungarian dishes. Hungary is a sunny country during summer, an ideal place for making excursions. There are several cultural programs in both Debrecen and Budapest, including jazz and classical music festivals, exhibitions and performances. And we especially invite you to bring your musical instruments to create our own festival!





**Table of Contents** (ordered by difficulty)

**Introductory**

- 180 *Barbara Beeton* / Editorial comments
  - typography and *TUGboat* news
- 179 *Karl Berry* / From the President
  - TUG activities and information for 2005
- 196 *Peter Flynn* / Typographers' Inn
  - superscripted ordinals, e-books et al., report layouts
- 273 *Enrico Gregorio* / Horrors in L<sup>A</sup>T<sub>E</sub>X: How to misuse L<sup>A</sup>T<sub>E</sub>X and make a *copy editor* unhappy
  - examples of some egregious L<sup>A</sup>T<sub>E</sub>X errors, and corrections
- 188 *Jim Hefferon* / Minutes in less than hours: Using L<sup>A</sup>T<sub>E</sub>X resources
  - using existing packages as building blocks for a new class
- 193 *Steve Peter* / `\starttext`: Swelled rules and MetaPost
  - introduction to using MetaPost in ConTeXt
- 183 *Gianluca Pignalberi* / Interview with Donald E. Knuth
  - reprinted from the *Free Software Magazine*
- 241 *Norbert Preining* / T<sub>E</sub>X Live for Debian
  - The T<sub>E</sub>X Live software distribution is now available as Debian packages
- 239 *Herbert Schulz* / The MacT<sub>E</sub>X distribution
  - Overview of a new easy-to-install distribution for Mac OS X, based on gwT<sub>E</sub>X
- 186 *Andrzej Tomaszewski* / Implementing editors' ideas—lots of fun, sometimes even more trouble
  - communication problems in the world of printing; reprinted from BachoT<sub>E</sub>X 2005

**Intermediate**

- 246 *Michael Barr* / Diagxy, a Lego-like diagram package
  - a generic package for building diagrams from combining blocks
- 280 *Karl Berry* and *Oleg Katsiadze* / Eplain 3: Expanded plain T<sub>E</sub>X
  - hyperlinks, graphics, color, and other user-level features for plain T<sub>E</sub>X
- 250 *Troy Henderson* / Embedding fonts in MetaPost output
  - making standalone eps files from MetaPost
- 264 *Mark LaPlante* / The treasure chest
  - new and updated CTAN packages in 2005
- 215 *Will Robertson* / Advanced font features with XgT<sub>E</sub>X—the `fontspec` package
  - accessing built-in Mac OS X fonts from L<sup>A</sup>T<sub>E</sub>X
- 224 *Ioannis Vamvakas* and *Panagiotis Kotopoulos* / ByZL<sup>A</sup>T<sub>E</sub>X: A package for typesetting “Byzantine” music
  - METAFONT fonts, a L<sup>A</sup>T<sub>E</sub>X package, and a historical introduction
- 233 *Gerben Wierda* / i-Installer: The evolution of a T<sub>E</sub>X install on Mac OS X
  - past, present, and future of i-Installer and the associated gwT<sub>E</sub>X (re)distribution
- 199 *Peter Wilson* / The alphabet tree
  - survey of writing systems from Sumeria to the present, profusely illustrated

**Intermediate Plus**

- 268 *Hendri Adriaens* and *Chris Ellison* / powerdot—making presentations with L<sup>A</sup>T<sub>E</sub>X
  - introduction to and implementation of a new class for making slides
- 253 *Peter Wilson* / Glistering
  - empty arguments; clear to even page; capitalizing first characters

**Advanced**

- 243 *Hans Hagen* / Hyphenation patterns in ConTeXt
  - languages and hyphenation files, especially in ConTeXt
- 256 Pearls of T<sub>E</sub>X programming
  - an array of short macros (some easy) with useful applications

**Contents of other T<sub>E</sub>X journals**

- 287 *T<sub>E</sub>Xemplares*: Contents of issue 7 (2005)
- 287 *Zpravodaj*: Contents of issues 14(3-4), 15(1), 15(2) (2004–05)
- 289 *Les Cahiers GUTenberg*: Contents of double issue 44–45 (November 2004)
- 290 *MAPS*: Contents of issue 32 (2005)
- 292 *The PractT<sub>E</sub>X Journal*: Contents of issues 2005-1–2005-4

**Reports and notices**

- 300 *Karl Berry* and *Kaja Christiansen* / T<sub>E</sub>X development fund 2005 report
- 298 *Robin Laakso* / Financial statements for 2005
- 297 Calendar
- 302 Institutional members
- 300 T<sub>E</sub>X consulting and production services
- 301 TUG membership form
- 303 EuroT<sub>E</sub>X 2006 announcement
- 302 Practical T<sub>E</sub>X 2006 announcement
- 304 TUG 2006 announcement