

# TUGboat

Volume 28, Number 1 / 2007  
Practical T<sub>E</sub>X 2006 Conference Proceedings

<b>General Delivery</b>	3	Karl Berry / <i>From the president</i>
	4	Barbara Beeton / <i>Editorial comments</i> Erratum: <i>TUGboat</i> 27:1 (EuroT <sub>E</sub> X proceedings); A new Korean T <sub>E</sub> X Society; L <sup>A</sup> T <sub>E</sub> X goes to the movies; Some <i>TUGboat</i> staff changes
<b>Warnings</b>	4	Donald E. Knuth / <i>T<sub>E</sub>X's infinite glue is projective</i>
<b>Software &amp; Tools</b>	5	Oleg Parashchenko / <i>T<sub>E</sub>XML: Resurrecting T<sub>E</sub>X in the XML world</i>
	11	Barbara Beeton and Idris Hamid / <i>Oriental T<sub>E</sub>X: A new direction in scholarly complex-script typesetting</i>
<b>Hints &amp; Tricks</b>	12	Peter Wilson / <i>Glistenings: stringing along; loops</i>
	15	Mark LaPlante / <i>The treasure chest</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	20	Ignacio Llopis Tortosa and María José Castro Bleda / <i>paperT<sub>E</sub>X: Creating newspapers using L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub></i>
	24	L <sup>A</sup> T <sub>E</sub> X Project Team / <i>L<sup>A</sup>T<sub>E</sub>X news, issue 17</i>
<b>Practical T<sub>E</sub>X 2006</b>	26	Conference program, delegates, and sponsors
<b>Keynote</b>	29	Barbara Beeton / <i>How to create a T<sub>E</sub>X journal: A personal journey</i>
<b>Publishing</b>	49	David Walden / <i>A lifetime as an amateur compositor</i>
	61	Elizabeth Dearborn / <i>T<sub>E</sub>X and medicine</i>
<b>Teaching &amp; Training</b>	65	Jon Breitenbucher / <i>L<sup>A</sup>T<sub>E</sub>X at a liberal arts college</i>
	70	Boris Veytsman / <i>Design of presentations: Notes on principles and L<sup>A</sup>T<sub>E</sub>X implementation</i>
<b>Software &amp; Tools</b>	77	Boris Veytsman and Maria Shmilevich / <i>Automatic report generation with Web, T<sub>E</sub>X and SQL</i>
	80	Bob Neveln and Bob Alps / <i>Writing and checking complete proofs in T<sub>E</sub>X</i>
<b>Graphics</b>	84	Troy Henderson / <i>A beginner's guide to MetaPost for creating high-quality graphics</i>
	91	Andrew Mertz and William Slough / <i>Graphics with PGF and TikZ</i>
	100	Boris Veytsman and Leila Akhmadeeva / <i>Drawing medical pedigree trees with T<sub>E</sub>X and PSTricks</i>
<b>Tutorials</b>	110	Peter Flynn / <i>Rolling your own Document Class: Using L<sup>A</sup>T<sub>E</sub>X to keep away from the Dark Side</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	124	Jim Hefferon / <i>L<sup>A</sup>T<sub>E</sub>X resources</i>
	126	Peter Flom / <i>L<sup>A</sup>T<sub>E</sub>X for academics and researchers who (think they) don't need it</i>
	129	Federico Garcia / <i>Hypertext capabilities with pdf L<sup>A</sup>T<sub>E</sub>X</i>
	133	Kaveh Bazargan and CV Radhakrishnan / <i>Removing vertical stretch — mimicking traditional typesetting with T<sub>E</sub>X</i>
<b>Abstracts</b>	137	Abstracts (Adams, Garcia, Höppner, Hummel, Moye, Peter, Wetmore)
<b>News</b>	138	Calendar
	139	TUG 2007 announcement
	140	EuroBachoT <sub>E</sub> X 2007 announcement
<b>TUG Business</b>	141	Steve Peter / <i>TUG 2007 election report</i>
	145	David Walden / <i>Financial statements for 2006</i>
	146	Institutional members
	147	TUG membership form
<b>Advertisements</b>	148	T <sub>E</sub> X consulting and production services

## **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**

2007 dues for individual members are as follows:

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

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

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

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is \$95 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 a discounted group rate and 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 © 2007 T<sub>E</sub>X Users Group.

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

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

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

Printed in U.S.A.

## **Board of Directors**

Donald Knuth, *Grand Wizard of T<sub>E</sub>X-arcana*<sup>†</sup>  
Karl Berry, *President*<sup>\*</sup>  
Kaja Christiansen\*, *Vice President*  
David Walden\*, *Treasurer*  
Susan DeMeritt\*, *Secretary*  
Barbara Beeton  
Jon Breitenbucher  
Steve Grathwohl  
Jim Hefferon  
Klaus H $\ddot{o}$ ppner  
Ross Moore  
Arthur Ogawa  
Steve Peter  
Cheryl Ponchin  
Samuel Rhoads  
Philip Taylor  
Raymond Goucher, *Founding Executive Director*<sup>†</sup>  
Hermann Zapf, *Wizard of Fonts*<sup>†</sup>

<sup>\*</sup>member of executive committee

<sup>†</sup>honorary

See <http://tug.org/board.html> for past board members.

### **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/>

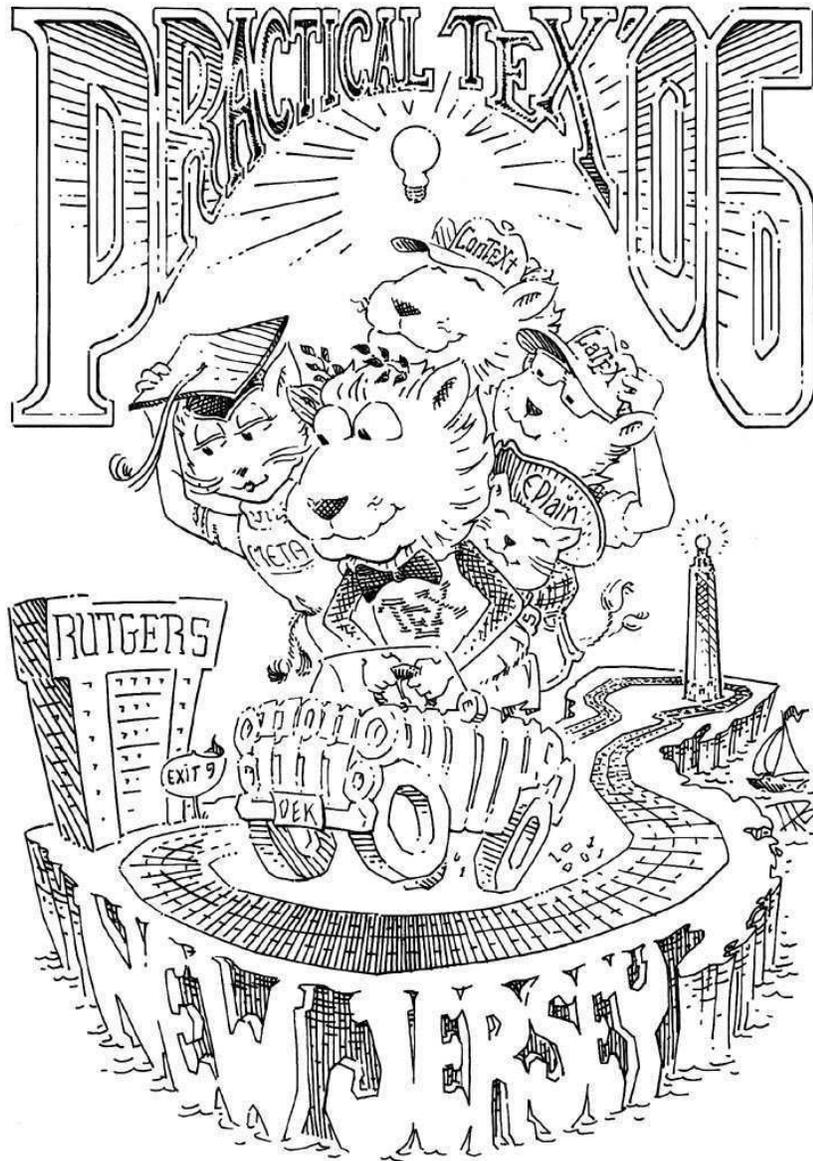
### **Have a suggestion? Problems not resolved?**

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

[printing date: March 2007]

# TUGBOAT

The Communications of the T<sub>E</sub>X Users Group



Volume 28, Number 1, 2007

Practical T<sub>E</sub>X 2006 Conference Proceedings

## **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**

2007 dues for individual members are as follows:

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

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

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

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is \$95 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 a discounted group rate and 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 © 2007 T<sub>E</sub>X Users Group.

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

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

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

Printed in U.S.A.

## **Board of Directors**

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

<sup>\*</sup>member of executive committee

<sup>†</sup>honorary

See <http://tug.org/board.html> for past board members.

### **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/>

### **Have a suggestion? Problems not resolved?**

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

[printing date: March 2007]

**Practical T<sub>E</sub>X 2006 Proceedings**

Rutgers, the State University of New Jersey

Piscataway, NJ, USA

July 30–August 1, 2006

# TUGBOAT

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

TUGBOAT EDITOR      BARBARA BEETON

PROCEEDINGS EDITOR      KARL BERRY

VOLUME 28, NUMBER 1

PORTLAND

•

OREGON

•

2007

U.S.A.

## **TUGboat**

This issue (Vol. 28, No. 1) is the first issue of the 2007 volume year. It combines the Practical T<sub>E</sub>X 2006 conference proceedings with regular material. Vol. 28, No. 2 is expected to be a regular issue, and No. 3 will contain the TUG 2007 (San Diego) 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 one issue of conference proceedings in 2007. Deadlines for presentation proposals (send to the conference committee) and the final papers:

- TUG 2007: abstracts April 23, 2007;  
preprints June 22, 2007;  
papers August 17, 2007.

Links, locations, and more information about all 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 T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X, are available from CTAN and the *TUGboat* web site. We also accept submissions using ConT<sub>E</sub>Xt.

Effective with the 2005 volume year, submission of a new manuscript implies permission to publish the article, if accepted, on the *TUGboat* web site, as well as in print. 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*  
Karl Berry, *Production Manager*  
Christina Thiele, *Associate Editor*,  
*Topics in the Humanities*

### **Production Team**

Barbara Beeton, Karl Berry (Manager),  
Kaja Christiansen, Robin Fairbairns, 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 T<sub>E</sub>X 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.  
T<sub>E</sub>X and  $\mathcal{A}\mathcal{M}\mathcal{S}$ -T<sub>E</sub>X are trademarks of the American  
Mathematical Society.

## General Delivery

### From the President

Karl Berry

### TeX Collection 2007

This issue of *TUGboat* should reach members' mailboxes at about the same time as the TeX Collection 2007 software on DVD and CD. The 2007 software consists of the same major components as the last release. For those who may not be familiar with it, here is a rundown of what's on the DVD:

- TeX Live, a cross-platform distribution including precompiled binaries for many systems;
- MacTeX, which adds a native MacOSX installer, the TeXShop front end, and other features to the base TeX Live;
- proTeXt, based on the MiKTeX Windows distribution with a guided installation;
- a snapshot of the CTAN archive.

A compressed variant of TeX Live, named *inst*, will be shipped on CD. This reduced version includes the same set of packages, but has precompiled binaries for only three systems: `i386-linux`, `powerpc-darwin`, and `win32`. It is provided for those who cannot read a DVD.

Since the software is being released in early 2007 rather than 2006, we will be sending it to both 2006 and 2007 members of TUG (except for those who explicitly opted not to receive it).

For more information and project web pages, please see <http://tug.org/texcollection>.

As we reach the end of another release cycle, I'd like to reiterate that the TeX Collection is a massive effort, done entirely by volunteers. We are grateful to the hundreds of people involved, from all parts of the TeX world: the contributors uploading new packages to CTAN, the CTAN maintainers for providing a central repository to draw from, the people building the binaries on a wide variety of platforms, those helping test the results, the developers maintaining and enhancing the software upon which it all rests, and the user group members keeping the infrastructure provided by TUG and all the TeX user groups viable through their support. Thanks to all.

### 2007 TeX conferences

2007 will see two major TeX conferences. First, a combined EuroTeX and BachoTeX in Bachotek, Poland, from April 28 to May 2. The call for papers deadline will have passed by the time this is

published, but please see the web site for registration and information: <http://www.gust.org.pl/conferences/EuroBachoTeX2007>.

Second, TUG 2007 in San Diego, California, from July 17–20. The deadline for presentation proposals is April 23. The conference theme is *Practicing TeX*, and I'd like to especially invite "ordinary" users and authors to attend and/or speak. The mix of attendees and presentations at the Practical TeX conferences in recent years has been gratifying and, I believe, fruitful for all, and I hope that that will continue at this annual TUG meeting.

I am very happy that Peter Wilson of his own Herries Press has accepted our invitation to be the keynote speaker at TUG 2007. Over the many years of his involvement with TeX, Peter has created the major memoir package, the archaic and bookhands font collections, the Glistings column for *TUGboat*, and much more. He talks about his background and interests in his interview at <http://tug.org/interviews>.

For registration and accommodation information (inexpensive on-campus housing is available), the call for papers, and more, please visit the conference web site at <http://tug.org/tug2007>. I hope to see you in sunny San Diego.

---

### Editorial comments

Barbara Beeton

Here we are in a new year, with a lot to catch up. Almost all material published last year was from conferences: EuroTeX 2005 (compiled jointly by DANTE and GUTenberg, both of whom were celebrating their 16th year, and sent to TUG members in lieu of one issue of *TUGboat*), EuroTeX 2006 (an issue that did contain some "regular" material), and TUG 2006. That didn't leave room for the proceedings of Practical TeX 2006, so we have included them in the present issue. Once again, we have an issue that combines papers presented at a meeting and ordinary articles.

As Karl has said, there won't be a separate Practical TeX meeting this year, so we expect that there will be one regular issue this year.

### Erratum: *TUGboat* 27:1 (EuroTeX proceedings)

An old version of an article by Siep Kroonenberg ("Managing a network TeX installation under Windows") was printed inadvertently in issue 27:1 (pp. 22–27) last year. The correct version is on line at <http://tug.org/tugboat>.

The more significant changes are:

- Credits: the paper originally appeared in slightly different form in NTG MAPS no. 33 (2005).
- Converters: the to-be-written GUI converter is now available and can be downloaded at <http://tex.aanhet.net/epspdf/>
- Disappearing filetypes: for this problem, a good workaround has been found.

We regret the error.

### A new Korean T<sub>E</sub>X Society

In January of this year, the on-line Korean T<sub>E</sub>X community (the Korean T<sub>E</sub>X Users Group) founded a new “off-line” community called the Korean T<sub>E</sub>X Society (KTS). KTS has members and will hold an annual meeting and conference.

The Society will also publish a new journal, *The Asian Journal of T<sub>E</sub>X*; the first issue is expected to appear around the end of April 2007. The editorial board consists of several well-known T<sub>E</sub>Xnicians, Prof. Haruhiko Okumura (Japan), Hàn Th<sub>ế</sub> Thành (Vietnam), CV Radhakrishnan (India), Werner Lemberg, and Jin-Hwan Cho (Korea).

### L<sup>A</sup>T<sub>E</sub>X goes to the movies

In the 2005 movie *Stealth*, an artificial intelligence system goes awry. The trivia listing for the film includes this tidbit:

When Keith Orbit is looking at the code for the AI, we can see that the code is written in LaTeX, which is a language for typesetting mathematics much as HTML is used on the Internet for typesetting web pages.

[www.imdb.com/title/tt0382992/trivia](http://www.imdb.com/title/tt0382992/trivia)

Incidentally, the movie was one of the biggest failures ever at the box office. Seems about par with their understanding that L<sup>A</sup>T<sub>E</sub>X, while the code may be pleasing to look at, is totally unsuitable for the basis of an AI system. Yes, we’re aware that at least one interpreter (for Basic<sup>1</sup>) has been written in T<sub>E</sub>X, but really!

(Thanks to Elizabeth Dearborn for unearthing this trivium and sending it to `texhax`.)

### Some TUGboat staff changes

If you pay attention to the TUGboat masthead, you will notice some changes. The most significant has been noted previously: Mimi Burbank, our Production Editor until November 2005, retired from her position at Florida State, and from the TUGboat

Board. She is missed, and we wish her well in her new life in Uganda. Karl Berry, in addition to all his other efforts as TUG President and chief cook and bottle washer for the T<sub>E</sub>X Live effort, has taken over as Production Editor. Thanks, Karl.

Two long-time Associate Editors have also gone on to other pursuits: Victor Eijkhout (Macros) and Alan Hoenig (Fonts). During their tenure, their knowledge and expertise were responsible for maintaining the high standards of their respective columns, and we are grateful for their contributions.

◇ Barbara Beeton  
American Mathematical Society  
201 Charles Street  
Providence, RI 02904 USA  
`bnb (at) ams dot org`

## Warnings

### T<sub>E</sub>X’s infinite glue is projective

Donald Knuth

T<sub>E</sub>X wizards might be interested in a phenomenon that could be considered an anomaly, but I choose to declare it a “feature.” Consider the two boxes

```
\hbox to 100pt{\hskip 0pt plus -100pt}
\hbox to 100pt{\hskip 0pt plus -1fil}
```

The first box is considered underfull, with a badness of 10000, because the total stretchability is negative. But the second box is perfectly fine, with a badness of 0, because the total stretchability is infinite. If you are tracing, the boxes are

```
\hbox(0.0+0.0)x100.0, glue set -1.0
.\glue 0.0 plus -100.0
\hbox(0.0+0.0)x100.0, glue set -100.0fil
.\glue 0.0 plus -1.0fil
```

within T<sub>E</sub>X’s gullet.

### References

- [1] *The T<sub>E</sub>Xbook*, page 97, although the case  $r < 0$  isn’t explicitly mentioned there.
- [2] *T<sub>E</sub>X: The Program*, sections 186, 646, 658, 659, 664, 665, 673, 676, 796, 852, 1007.

◇ Donald Knuth  
Stanford University

<sup>1</sup> Andrew Marc Greene, “BAS<sub>I</sub>X: An interpreter written in T<sub>E</sub>X”, *TUGboat* 11:3, pp. 381-392

# Software & Tools

## TeXML: Resurrecting TeX in the XML world

Oleg Parashchenko

### 1 Foreword

TeXML is an XML syntax for TeX, L<sup>A</sup>TeX and ConTeXt. This definition is extremely correct, but I dislike its formality. Instead, I prefer the following.

Thanks to TeXML, you can reuse your TeX skills in the XML world. With TeXML, XML publishing becomes a case of TeX publishing.

TeXML is a very simple thing. You can learn it in a minute by looking at the examples in the section ‘TeXML tour’. But knowing the syntax isn’t enough.

To feel TeXML, you need to know its past and future, the ideas behind it, and understand the author’s intentions. That’s why the technical stuff is wrapped by the sections with my very subjective view on the topic of XML publishing.

In the most cases, the words ‘TeX’ and ‘L<sup>A</sup>TeX’ are interchangeable, and they mean also any other TeX format.

The author is from the XML world. The TeXML home page is <http://getfo.org/texml/>.

### 2 Why XML, not TeX, why TeX, not XML

The best thing about XML is that everyone knows what it is. XML is ubiquitous now, and especially in the area of technical documentation. Indeed, its parent, SGML, was created to support authoring of technical manuals.

TeX users have different opinions on XML. But nobody rejects the idea of logical markup is very obvious and essential. From the high level point of view, all the markup methods are the same.

What in XML looks like

```
<environment> ...text... </environment>
```

in L<sup>A</sup>TeX looks like this:

```
\begin{environment} ...text... \end{environment}
```

The only difference is notation. But it’s a very important difference. Computers prefer XML, humans prefer L<sup>A</sup>TeX.

Among benefits of logical markup is the possibility of single source publishing, when the same source document can be converted to different output formats. XML is the best choice because XML libraries exist in any practical programming language. On the other hand, the only correct TeX parser is TeX itself, and TeX is locked in its sandbox.

On the other side, the ideal XML world isn’t ideal. How to get PDF from XML? Theory says that you would write an XSLT (W3C, 1999) program which converts XML to XSL-FO (W3C, 2001), and use an XSL-FO formatter which generates PDF from XSL-FO.

XML+XSLT → XSL-FO → PDF. There are two issues: first, tools, which is hopefully temporary; and second, too much automation, which is fatal.

Only a few tools implement XSL-FO in full, and all these tools are commercial, without open source alternatives (the best one is FOP, which is under development), and the W3 Consortium has started work on XSL-FO 2.0.

But the worst is that the joke ‘*automatically*’ means you can’t fix it if something goes wrong applies perfectly to the XSL-FO way. When you need to tune a generated layout, you’ll find that XSL-FO level is too low, and editing XSL-FO isn’t much better than editing PDF. Also you’ll find that XML and XSLT levels are too high and editing here smells bad.

The broken layout isn’t a showstopper in L<sup>A</sup>TeX. Your writings are marked up logically, and when you need typographical tunings, you just use low-level primitives.

Time for a short summary:

- XML is good as a markup language,
- TeX is good for publishing documents.

Why not take the best from both worlds? That is, have sources in XML and publish the documents through TeX. But how?

### 3 XML to TeX — how

When converting XML, there is no better alternative than XSLT. This language is specially designed to convert XML, is based on experiences with the Lisp-like DSSSL language, has a large user and expert base, and has decent support by many tools on many platforms.

Why not Java, or Perl, or Python, or something else? Because XML is alien to them. It’s inconvenient to use the traditional languages for processing XML, for either parsing or converting.

For example, in one project the author worked on a Java application. One procedure was more than 20 lines in size, debugged and enhanced several times, and still couldn’t be compared in functionality with a small XPath (a part of XSLT) expression of several characters.

Worse, the whole library was a partial, poorly documented, limited re-invention of XSLT. I think it’s the doom of any program which converts XML. Instead of using a poor imitation, it’s better to use XSLT itself.

The knowledgeable reader can say that XSLT is a language to convert from XML to XML, not from XML to  $\text{\TeX}$ , and ask if XSLT is still so great to generate  $\text{\TeX}$ .

No, I have to answer, converting XML directly to  $\text{\TeX}$  is nightmare. XSLT is very weak and unbelievably verbose in working with strings, but that's what is required when generating  $\text{\TeX}$  code.

What is expected from a  $\text{\TeX}$  code generator:

- escaping special  $\text{\TeX}$  characters (for example ' $\langle$ ' to ' $\backslash\langle$ ' or, better, to ' $\backslash\text{less}\{\}$ ');
- disjoining ligatures ('---' isn't the long dash in XML, the long dash is the symbol ' $\&\#x2014$ ');)
- mapping from Unicode characters to  $\text{\LaTeX}$  sequences;
- avoiding empty lines, which start a new paragraph in  $\text{\TeX}$ .

And there are common errors when generating  $\text{\TeX}$  code. (See bug databases for such projects as db2latex (Casellas and Devenish, 2004), dbleatex (Guillon, 2006) and others.)

- Opening or closing brace is forgotten.  
 $\langle i \rangle \text{some} \langle /i \rangle \text{ text}$   
 $\rightarrow \{\backslashit \text{ some} \text{ text}$   
instead of  $\{\backslashit \text{ some}\} \text{ text}$ .<sup>1</sup>
- No space after the command name.  
 $\{\backslashitsome\} \text{ text}$
- Space instead of braces.  
 $\text{here is} \langle i \rangle \text{ some} \langle /i \rangle \text{ text}$   
 $\rightarrow \text{here is} \{\backslashit \text{ some}\} \text{ text}$   
instead of  $\text{here is} \{\backslashit\} \text{ some}\} \text{ text}$

If you write a  $\text{\TeX}$  code generator, you should pay attention to everything. You need accuracy and patience, and the work isn't trivial. Therefore you'd prefer to delegate  $\text{\TeX}$ ification from your program to something else.

$\text{\TeX}$ ML is the best and probably the only candidate. You create XML, which is much easier, and then a  $\text{\TeX}$ ML processor converts  $\text{\TeX}$ ML to  $\text{\TeX}$ .

Short summary:

- XSLT is the best tool for converting XML to XML,
- it's better to delegate  $\text{\TeX}$  code generation.

That's why we have  $\text{\TeX}$ ML, an XML syntax for  $\text{\TeX}$ / $\text{\LaTeX}$ / $\text{Con}\text{\TeX}$ t. Conversion from XML to  $\text{\TeX}$  consists of two steps:

- an XSLT program converts XML to  $\text{\TeX}$ ML, and
- a  $\text{\TeX}$ ML processor converts  $\text{\TeX}$ ML to  $\text{\TeX}$ .

$\text{\TeX}$ ML is an XML language with just a few tags, and converting XML to XML is the specialization of

<sup>1</sup> In production we might use  $\backslash\text{textit}\{...\}$ , but for illustrative purposes here I use  $\{\backslashit \dots\}$ .

XSLT; therefore you need only basic knowledge of XSLT to convert XML to  $\text{\TeX}$ .

#### 4 $\text{\TeX}$ ML tour

The  $\text{\TeX}$ ML markup language is minimalistic. Most of the time, you use only three elements: `cmd`, `env` and `group` (the other elements are `pdf`, `math`, `dmath`, `ctrl`, `spec` and `TeXML`).

To get accustomed to  $\text{\TeX}$ ML, it's enough to learn the examples presented in this section. The original paper by Douglas Lovell (Lovell, 1999) is also a good introduction, but it's out of date. For a detailed description of contemporary  $\text{\TeX}$ ML, consult the  $\text{\TeX}$ ML specification (Parashchenko, 2006b).

Installation and usage instructions are on the  $\text{\TeX}$ ML home page: <http://getfo.org/texml/>. A pleasant feature is that it's enough to unpack the distribution package to use  $\text{\TeX}$ ML. The installation procedure isn't required, it's for convenience only.

##### 4.1 Simple $\text{\TeX}$ ML file

An example of a simple  $\text{\TeX}$ ML document:

```
<TeXML>
<TeXML escape="0">
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
</TeXML>
<env name="document">
I'm not afraid of the symbols €,
$, > and others.
</env>
</TeXML>
```

The result of conversion to  $\text{\TeX}$  is the  $\text{\LaTeX}$  document:

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\begin{document}
I'm not afraid of the symbols \^{\},
\textdollar{\}, \textgreater{\} and others.
\end{document}
```

This example demonstrates:

- the root element is `TeXML`,
- $\text{\TeX}$  special symbols are escaped automatically,
- it's possible to disable escaping.

By the way, while preparing the original  $\text{\LaTeX}$  example, I made two errors:

- ' $\backslash\text{textgreater}$ ' instead of ' $\backslash\text{textgreater}\{\}$ ' (result — no space after the symbol ' $\>$ '),
- ' $\backslash\^{\}$ ' instead of ' $\backslash\^{\}\{\}$ ' (result — the circumflex over the comma instead of the symbol ' $\^{\}$ ').

$\text{\TeX}$ ML saves me from such basic errors.

Disabling escaping is not recommended. Usually it's a misuse of T<sub>E</sub>XML. But to keep examples simple, I do use it for creating the L<sup>A</sup>T<sub>E</sub>X header.

## 4.2 More T<sub>E</sub>XML

This document uses more T<sub>E</sub>XML elements:

```
<TeXML>
  <cmd name="documentclass">
    <opt>a4paper</opt>
    <parm>article</parm>
  </cmd>
  ....
  <env name="document">
    Hello, <group><cmd name="it"/>World</group>!
  </env>
</TeXML>
```

After converting to T<sub>E</sub>X, the result is:

```
\documentclass[a4paper]{article} ....
\begin{document}
Hello, {\it}World!
\end{document}
```

This example demonstrates the three most of-ten used T<sub>E</sub>XML elements:

- `cmd` creates a L<sup>A</sup>T<sub>E</sub>X command,
- `env` creates a L<sup>A</sup>T<sub>E</sub>X environment,
- `group` creates a L<sup>A</sup>T<sub>E</sub>X group.

The example also demonstrates how to create the L<sup>A</sup>T<sub>E</sub>X header using regular T<sub>E</sub>XML instead of disabling escaping.

## 4.3 Better layout

This example demonstrates how to tune the layout of a generated L<sup>A</sup>T<sub>E</sub>X code. The result can be made indistinguishable from code written by a human.

In the last example, we got the following L<sup>A</sup>T<sub>E</sub>X document:

```
\documentclass[a4paper]{article} ....
\begin{document}
Hello, {\it}World!
\end{document}
```

A better code layout is:

```
\documentclass[a4paper]{article}
....
\begin{document}
Hello, {\it World}!
\end{document}
```

The source T<sub>E</sub>XML code uses the attributes `n12` and `gr` to tune the layout:

```
<TeXML>
  <cmd name="documentclass" n12="1">
    <opt>a4paper</opt>
    <parm>article</parm>
  </cmd>
```

```
....
  <env name="document">
    Hello, <group>
      <cmd name="it" gr="0"/>World</group>!
  </env>
</TeXML>
```

## 4.4 PDF literal strings

Let's start with the following L<sup>A</sup>T<sub>E</sub>X code:

```
\documentclass{article}
\usepackage[T2A]{fontenc}
\usepackage[koi8-r]{inputenc}
\usepackage{hyperref}
\begin{document}
\section{Заголовок (Title)}
Текст (Text)
\end{document}
```

The code looks fine, but due to the Russian letters, L<sup>A</sup>T<sub>E</sub>X raises the errors:

```
Package hyperref Warning:
Glyph not defined in PD1 encoding,
(hyperref) removing '\CYRZ' on input line 6.
```

For the document above, the solution is to use `\usepackage[unicode]{hyperref}`

But this solution is not generic. For example, for CJK text, it fails with some obscure error like:

```
! Incomplete \ifx; all text was ignored ...
```

I prefer the universal solution that uses Unicode strings for the PDF names:

```
\documentclass{article}
\usepackage[T2A]{fontenc}
\usepackage[koi8-r]{inputenc}
\usepackage[unicode]{hyperref}
\begin{document}
\section{\texorpdfstring{Заголовок (Title)}
}{\004\027\004\060\004\063\004\076\004\073
\004\076\004\062\004\076\004\072\000\040\0
00\050\000\124\000\151\000\164\000\154}}
Текст (Text)
\end{document}
```

Comparing to the previous example, I use

- the option `unicode` for the package `hyperref`,
- the command `texorpdfstring` to assign the name for the PDF bookmark entry.

The content of `texorpdfstring` is created by the T<sub>E</sub>XML command `pdf`:

```
<cmd name="section">
  <parm>
    <cmd name="texorpdfstring">
      <parm>Заголовок (Title)</parm>
      <parm><pdf>Заголовок (Title)</pdf></parm>
    </cmd>
  </parm>
</cmd>
```



(Parashchenko, 2006c) (XSLT+Scheme), one of the Google Summer of Code 2005 projects, presented at the XTech 2006 conference.

TeXML popularity grew, and I started to get contributions. One of the TeXML users, Paul Tremblay, used ConTeXt for publishing. He added ConTeXt support to TeXML, reworked bits of TeXML code and wrote extensive documentation (Tremblay, 2005) on how to imitate XSL-FO constructions in ConTeXt. That's a must-read for those who are interested in the topic.

In June 2006, I collected all the improvements, rewrote documentation, packed the whole as a usual Python package and released version 2.0. No bugs reported till now (March 2007).

## 6 The TeXML processor: present and future

At the moment, the only TeXML processor implementation is written by me in Python. It uses only few standard modules and therefore is portable and can be used anywhere if Python is installed.

The core of the TeXML processor is a stand-alone Python library, therefore TeXML functionality is available to any Python application. It might be that TeXML is available to Java programs using JPython and to .NET programs using IronPython, but checking this has low priority on my long-term TODO list.

TeXML follows the three-step approach to software development: make it work, make it correct, make it fast. TeXML is currently on the second level, 'work correct', so now it's time to improve performance. The processor works much faster than XSLT, but it can be made an order of magnitude faster yet.

The approach is to use finite automata. The current code escapes the output stream character by character. The set of loops, flags and nested conditions adds an overhead to the processing time. By comparison, with automata the only flags are the current state, the current character, and the table of state changes. Overhead per character is minimized.

The second main benefit of automata is that it would make explicit all the rules how to generate correct TeX code with nice layout. At the moment, this knowledge is hidden inside the spaghetti code, that is hard to maintain and modify.

And I'd like to improve some things. For example, the TeXML

```
<cmd name="command"/><ctrl ch="\"/>
```

is translated to

```
\command{}}\
```

I'd prefer to automatically avoid dummy groups:

```
\command\
```

Yet another benefit of using automata is that TeXML could be ported to other languages. The non-trivial TeXML logic, were it written as automata in some well-known format, such as S-expressions or XML, could be automatically translated to a code in any language.

Unfortunately, all these wonderful perspectives are for the far far future. I'm satisfied with the current state of TeXML and prefer to concentrate on other projects.

Creating automata for TeXML could be a good master thesis or even a PhD work. If you know someone who might be interested in this task, don't hesitate to mention TeXML.

## 7 Nice layouts, diff and patch

Probably you've noticed how much attention I devote to the nice layout of the generated code. But what's the benefit except aesthetic?

Before answering, I'd like to note that aesthetic appearance is indeed a benefit. You know the saying, ugly things can't fly. I believe in it. And definitely, nobody is interested in working with the intermediate ugly code which appears in many other XML-to-PDF-through-L<sup>A</sup>T<sub>E</sub>X projects.

Automatically generated PDFs can't be ideal. From time to time, there are layout faults that you'd like to fix. To tune these places, you need to edit the L<sup>A</sup>T<sub>E</sub>X code. When this code is ugly and bad, you might prefer to tolerate the faults instead of fixing them. On the contrary, when the code is human-friendly, you are likely to look into the code and fix the problems.

But the main benefit of human-friendly code is that such code is also `diff`- and `patch`-friendly.

Imagine that you've fixed all the layout faults in the L<sup>A</sup>T<sub>E</sub>X code. Unexpectedly, a proofreader has updated the source XML. How to generate a new PDF, both with your and the proofreader's changes? The naive user has two alternatives:

- detect what's changed in XML and repeat the changes in the L<sup>A</sup>T<sub>E</sub>X code, or
- re-generate PDF and re-apply layout corrections in the L<sup>A</sup>T<sub>E</sub>X code.

Both options are miserable, boring and error-prone. Open source software developers would prefer a better way using `diff` and `patch`.

- Take the initial L<sup>A</sup>T<sub>E</sub>X file, take the current version with the layout fixes, and generate a patch-file using `diff`.
- Generate a new PDF from the new XML.

- Apply the patch-file to the new L<sup>A</sup>T<sub>E</sub>X file and re-generate the PDF.

In most cases, everything goes smoothly and all the changes, from both you and the proofreader, are applied.

Thanks to the good L<sup>A</sup>T<sub>E</sub>X code formatting, as produced by T<sub>E</sub>XML, this way is indeed possible. Instead of saying ‘patch-file’, I prefer to say ‘beauty memory’. It sounds more appealing and descriptive.

To automate this procedure, I developed Consodoc (Parashchenko, 2006a), an XML to PDF publishing tool on top of T<sub>E</sub>XML. The user’s guide for Consodoc is generated by Consodoc itself. Here is an example of the project file:

```
import Consodoc
env = Consodoc.default_process(
  in_file = 'in/guide.xml',
  in_xslt = 'support/guide.xsl'
)
Depends('tmp/guide.pdf', 'support/guide.cls')
```

The project file defines that the source XML file is `in/guide.xml`, T<sub>E</sub>XML is generated by the XSLT program `support/guide.xsl`, and implicitly defines that the patch file is `in/guide.patch`. It also specifies, explicitly and implicitly, the dependencies of the files: if a file is changed, than all the dependent files should be re-generated. To build PDF, just say on the command line: `cdoc`.

Consodoc is a very new product, but it is already usable and successfully passed unit and functional testing. I recommend Consodoc for use in the production environment by early adopters.

## 8 Final words

Publishing XML is still a practical problem, even when the quality of the result isn’t very important. Different approaches are suggested, from using the XSL-FO standard to developing a custom solution, but the Right Thing is still to appear.

The T<sub>E</sub>XML approach is one of the candidates. Instead of inventing something new, it smoothly integrates existing successful technologies and experience. First, it uses T<sub>E</sub>X as the typesetting engine. Second, it uses XSLT as the conversion language.

Third, with the help of the `diff` and `patch` tools, the beauty memory maintains layout corrections of the PDF documents. I’m not aware of any other XML-to-PDF solution with this feature.

The only T<sub>E</sub>XML problem is the lack of sample conversion scripts. But I’ve started work on the T<sub>E</sub>XML stylesheets for DocBook, a popular XML standard for technical books, therefore this problem will be fixed in the near future.

I expect this union — T<sub>E</sub>XML, beauty memory and DocBook T<sub>E</sub>XML stylesheets — will have a big impact on XML publishing, causing restoration of the T<sub>E</sub>X technologies in the modern XML world. Join the T<sub>E</sub>XML movement!

## References

- W3C. “XSL Transformations (XSLT). Version 1.0. W3C Recommendation 16 November 1999”. See <http://www.w3.org/TR/1999/REC-xslt-19991116>, 1999.
- W3C. “Extensible Stylesheet Language (XSL). Version 1.0. W3C Recommendation 15 October 2001”. See <http://www.w3.org/TR/2001/REC-xsl-20011015/>, 2001.
- Casellas, Ramon, and J. Devenish. “Welcome to the DB2L<sup>A</sup>T<sub>E</sub>X XSL Stylesheets”. See <http://db2latex.sourceforge.net/>, 2004.
- Guillon, Benoît. “DocBook to L<sup>A</sup>T<sub>E</sub>X/ConT<sub>E</sub>Xt Publishing”. See <http://dblatex.sourceforge.net/>, 2006.
- Houser, Chris. “T<sub>E</sub>XMLapis”. Available from <http://bluweb.com/us/chouser/proj/texmlapis/>, 2001.
- Lovell, Douglas. “T<sub>E</sub>XML: Typesetting XML with T<sub>E</sub>X”. *TUGboat* **20**(3), 176–183, 1999.
- Parashchenko, Oleg. “sT<sub>E</sub>Xme”. See <http://stexme.sourceforge.net/>, 2004a.
- Parashchenko, Oleg. “T<sub>E</sub>XML: an XML vocabulary for T<sub>E</sub>X”. See <http://getfo.org/texml/thesis.html>, 2004b. Thesis for the First International Conference of Open-Source Developers, Obninsk, Russia.
- Parashchenko, Oleg. “Consodoc publishing server: XML to beautiful documents”. See <http://consodoc.com/>, 2006a.
- Parashchenko, Oleg. “T<sub>E</sub>XML specification”. See <http://getfo.org/texml/spec.html>, 2006b.
- Parashchenko, Oleg. “XSieve: extending XSLT with the roots of XSLT”. See <http://xmlhack.ru/protva/xtech2006-paper.pdf>, 2006c. XTech 2006: Building Web 2.0, 16-19 May 2006, Amsterdam, The Netherlands.
- Tremblay, Paul. “Welcome to context-xml”. See [http://getfo.org/context\\_xml/](http://getfo.org/context_xml/), 2005.

◇ Oleg Parashchenko  
Saint-Petersburg State University,  
7-9, Universitetskaya nab,  
Saint-Petersburg, Russia  
olpa (at) uucode dot com  
<http://uucode.com/>

## Oriental T<sub>E</sub>X: A new direction in scholarly complex-script typesetting

Project by Idris Samawi Hamid

Preservation of much ancient scholarship of non-Western civilization exists only through unedited manuscripts. The manuscripts themselves are often not readily accessible, and the ability to make available accurate and culturally authentic typeset copies is restricted. Particularly for documents in Arabic script, typesetting is hampered by the lack of complete sets of vowel markings and diacritics, crucial for understanding the meaning of these texts.

For scholars working on critical editions of documents in Latin script, T<sub>E</sub>X is the tool of choice. Dr. Idris Samawi Hamid of the Colorado State University Department of Philosophy has received a grant to provide a comparable tool for Arabic scholars.

Dr. Hamid's own fields of specialty include Islamic philosophy, metaphysics, and cosmology. He has prepared critical editions of two major works of Arabic scholarship:

- *Shaykh Aḥmad Aḥsā'ī's Observations in Wisdom: Critical Edition, Translation, Notes, and Glossary*
- *The Mystical Theology of Muḥammad 'Alī Shaḥābādī: A Critical Edition of Rashahāṭu āl-Biḥār, with Notes and Glossary*

Both works have been accepted for publication; what remains is to prepare proper typeset copy ready for printing. This is the impetus for the project and the proposal.

In February 2006, Dr. Hamid submitted a proposal to CSU's Integrated Research Projects Program, requesting a grant to develop and implement extensions to T<sub>E</sub>X to provide such software. The result of this work will be called Oriental T<sub>E</sub>X. The proposal was accepted by the program, and work began in May 2006. Taco Hoekwater is the principal programmer for the project.

Development includes a number of components:

- Extending the data structures of LuaT<sub>E</sub>X and Aleph to handle non-Latin languages and UTF-8 input files;
- Implementing two levels of right-to-left machinery: a global component for handling page elements, and a local component for switching direction in text without disrupting the typesetting process;
- Implementing dynamic ligaturing to accommodate the multiple, contextually-dependent shapes of Arabic letters, and the vertical shifting characteristic of particular written Arabic

dialects; this will be based on concepts already present in Aleph;

- Creating control languages to handle conversion from the input stream to internal character representation, and to manage the context-driven glyph selection; again, many of the basic concepts are present in Aleph, but require adaptation to separate the two distinct processes;
- Developing OpenType font support, enabling use of fonts larger than 256 characters, and providing the mechanism to use such fonts;
- Adding extensions for critical editions, including a line-numbering engine and improved footnote handling;
- Quality control, involving extensive testing to assure that the goals of the development have been met;
- Documentation, a basic reference that is good enough for a skilled macro package programmer to learn how to take advantage of the special features provided by Oriental T<sub>E</sub>X; the goal is to provide user-interface macros that are easy to use, and suitable for typesetting of critical editions.

At TUG 2006, Taco presented a report on the status of the project. (This was particularly appropriate to the venue of the conference, in Marrakesh, Morocco, where other scholars and students reported on their own projects in Arabic typesetting.) As of early November 2006, the first stage was complete; this included support for full Unicode input, and merging of Aleph and pdfT<sub>E</sub>X.

The second stage, comprising support of OpenType fonts and PDF output from Aleph, is essentially complete as of this writing.

The third stage, expected by the time of Bacht<sub>E</sub>X, includes completion of hyphenation patterns, character (case-mapping) tables, end-of-sentence discovery, handling of minimal word size, line justification, and the ligature table. Also in the third phase are the decoupling of characters and glyphs, with separate nodes for Unicode characters, fonts, and glyphs in fonts, as well as revamped paragraph breaking routines and a dynamic font interpolation engine.

The fourth and final phase of the T<sub>E</sub>X extensions, expected by the time of the TUG San Diego meeting, will see two-dimensional line typesetting of Arabic script and improved font handling, completion of the line-numbering engine, and improved footnote handling.

The generous support of the CSU Integrated Research Projects Program is gratefully acknowledged.

## Hints & Tricks

### Glisterings

Peter Wilson

'Tis better to be lowly born  
And range with humble livers in content  
Than to be perked up in a glist'ring grief  
And wear a golden sorrow.

---

*Henry VIII*, WILLIAM SHAKESPEARE

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.

To no one but the Son of Heaven does  
it belong to order ceremonies, to fix the  
measures, and to determine the written  
characters.

---

*The Analects*, CONFUCIUS

### 1 Stringing along

In an earlier column [3] I mentioned that I might continue looking at character strings. Here is some basic code that can be used for examining each character in a simple string:

```
\catcode'\^^G=12
\newcommand*\allchars}[1]{%
  \def\stuff{#1}\ifx\stuff\empty\else
    \@llchars#1^^G\fi
\def\@llchars#1#2^^G{%
  \def\letter{#1}\def\others{#2}%
  \ifx\letter\empty\let\next@gobble
  \else
    \doachar{#1}%
    \ifx\others\empty \let\next@gobble
    \else \let\next\@llchars \fi
  \fi
  \next#2^^G}
\catcode'\^^G=15
```

Here I have used the special character ^^G as a marker for the end of the string. This is normally an invalid character but I temporarily changed its catcode to make it an 'other' character (like @ normally is). The \@gobble macro is part of the L<sup>A</sup>T<sub>E</sub>X kernel; it takes one argument and does nothing with it. Buried inside the code \allchars calls a macro \doachar{<char>} for each character in the string. With this definition

```
\newcommand*\doachar}[1]{\textit{#1}}
```

some examples of \allchars are:

```
\allchars{allchars} -> allchars
\allchars{{\oe}rstead's} -> ærstead's
\allchars{} ->
\allchars{with spaces} -> withspaces
```

The special case of an empty argument is handled in the \allchars macro itself, while everything else is dealt with by \@llchars. This keeps calling itself, grabbing one character from the initial string each time until all are used up, via a process called *tail recursion*, meaning that the last thing that it does is call itself (or effectively do nothing if all the characters have been processed).

Remember that with L<sup>A</sup>T<sub>E</sub>X, if you put any code that includes macros with @ in their names it either has to go in a package file (a .sty file) or be surrounded by the \makeatletter and \makeatother pair of commands.

One unfortunate property of \allchars is that it discards all spaces in the original string. Spaces can be handled by a two-part process. The first part goes through the string word by word, where a word is a set of characters followed by a space. The second part then goes through each word character by character.

First some preliminaries and the main user command \Upeach.

```
\newif\if@newword
\def\checkrelax{\relax}
\catcode'\^^G=12
\newcommand*\Upeach}[1]{%
  \@upeach#1^^G}
\def\@upeach#1^^G{%
  \def\stuff{#1 }%
  \expandafter\getaword\stuff ^^G}
```

The \getaword macro extracts the next word from the string (note the argument delimited by a space). It then calls \getachar with the word as its argument.

```
\long\def\getaword#1 {%
  \@newwordtrue
  \expandafter\getachar#1\relax}
```

\getachar gets the next 'letter' in the word. If it is ^^G then the string is finished. If the letter is the same as \relax then it is a space and the macro must call \getaword to repeat the cycle. Otherwise it has a letter, calls \doUpeach to do something with it, and calls itself again to get the following letter.

```
\def\getachar#1{%
  \def\letter{#1}%
  \if\letter^^G\let\next\relax
  \else
    \ifx\letter\checkrelax
      \let\next\getaword
```

```

\else
  \doUpeach{#1}%
  \let\next\getachar
\fi
\fi
\next}
\catcode'\^^G=15

```

`\getachar` is another example of tail recursion.

The macro `\doUpeach` checks if a new word has just started. If so, it converts its argument into italic uppercase and sets `\@newwordfalse`. If its argument is not the first letter in a word it typesets it in a bold font. Of course this is not a realistic thing to do—it’s merely to demonstrate that all the characters in the string have been examined.

```

\newcommand*\doUpeach}[1]{%
  \if@newword
    \space\textit{\MakeUppercase{#1}}%
    \@newwordfalse
  \else \textbf{#1}\fi}

```

Here are a couple of examples:

```

\Upeach{string with spaces} ->
  String With Spaces
\Upeach{{\oe}rstead’s rule} ->
  Erstead’s Rule

```

These macros work for simple strings but are likely to fail if there are accents or anything else to disturb the even tenor of simple characters. The earlier column [3] gave an indication of how such problems might be resolved. On the other hand, it could be a lot simpler and quicker to change the strings by hand using your normal text editor.

```

Here we go loop de loop.
Here we go loop de li.
Here we go loop de loop
On a Saturday night.

```

---

*Loop de loop, JOHNNY THUNDER?*

## 2 Loops

There are occasions when you need to perform a repetitive action that does not involve string processing.  $\TeX$  provides a `\loop ... \repeat` which can be useful in some circumstances. The general scheme is like this:

```

\loop
  <lots of useful commands>
\if<some condition>
  <more code>
\repeat

```

$\TeX$  processes the commands following the `\loop` and then performs the `\if` test (without any closing `\fi`). If the test is true  $\TeX$  will then process the

`<more code>` and start again with the first batch of commands. If the condition is false it will do whatever comes after the `\repeat`.

$\LaTeX$ , among other internal facilities, provides a mechanism for going through a list of things that are separated by commas (like the option list for a class or package). This scheme looks like:

```

\@for\scratch:=<list>\do{%
  <something with \scratch>}

```

where `\scratch` is some command name and `<list>` is a comma-separated list. It takes each element of the list in turn, defines `\scratch` as that element and then does whatever you tell it to do with it. This continues until the list is exhausted.

It is easier to see how these work with a real example. The following is a very stripped down version of some code from the `memoir` class [2]. It provides a means of putting a list of things into a tabular form without having to worry about signifying the end of each row. The command is:

```

\fillrows{<width>}{<numcols>}{<comma separated list>}

```

which will create a centered tabular form of overall width `<width>` and `<numcols>` columns, with the elements from `<comma separated list>` filling up the tabular row by row (i.e., left to right, top to bottom). I got the initial idea from *TEX for the Impatient* [1] which gave a  $\TeX$  version, filling top to bottom and left to right.

First some counters and lengths, etc., that we need. Be warned, much of the code below you won’t want to know about and I’m not going to try and explain it. In  $\LaTeX$  this is the kind of stuff that is hidden within the `tabular` environment.

```

\newcount\CT@cols      % number of cols
\newcount\@cellstogo   % columns left
\newdimen\CT@col@width % column width
\newtoks\crtok
  \crtok = {\cr}%

```

Now we can start on `\fillrows` itself, which takes three arguments—the overall width, the number of columns, and the list of entries. The first part sets up counters based on the number of columns.

```

\newcommand*\fillrows}[3]{\par\begingroup
  \CT@cols=#2\relax
  \@cellstogo=\CT@cols

```

The next bit defines code that will be called after each entry is put into the tabular; it will insert either a `&` or the internal form of `\`.

```

\def\@endcolactions{%
  \global\advance\@cellstogo\m@ne
  \ifnum\@cellstogo<\@ne
    \global\@cellstogo=\CT@cols
  \the\crtok

```

```

\else
&
\fi}%

```

Calculate the column widths and start off the tabular by defining the preamble (the general layout of the tabular).

```

\CT@col@width=#1
\divide\CT@col@width \CT@cols
\penalty 10000\relax
\noindent
\vskip -\z@
\def\@preamble{%
\begingroup
\let\@sharp\relax

```

Now comes a `\loop... \repeat` going over all but one of the columns, and for each column extending the `\@preamble` by adding some spacing and a `&`.

```

\ifnum\CT@cols>\@ne
\loop
\g@addto@macro{\@preamble}{%
\hb@xt@ \CT@col@width
{\strut\relax\@sharp\hfil} &}%
\advance\CT@cols\m@ne
\ifnum\CT@cols>\@ne
\repeat
\fi

```

The `&` is not required for the last column.

```

\g@addto@macro{\@preamble}{%
\hb@xt@ \CT@col@width
{\strut\relax\@sharp\hfil}}%
\endgroup

```

(The above code sets each column to a fixed width (`\CT@col@width`). Commenting out the two lines that start with `\hb@xt@` will result in each column being set to its natural width, just wide enough for the widest entry in the column.) Now finish up the preliminaries.

```

\let\@sharp ##
\tabskip\fill
\halign to\hsize \bgroup
\tabskip\z@
\@preamble
\tabskip\fill\cr

```

The entries are added to the tabular, using a `\@for` loop to extract each entry from the comma-separated list.

```

\@for\@tempa:=#3\do{%
\@tempa\unskip\space\@endcolactions}%

```

All the entries have been dealt with, so wrap everything up.

```

\the\crtok \egroup \endgroup \par}

```

As a simple example, the code below creates the following tabular:

```

\fillrows{0.7\textwidth}{3}{ one, two,
three, four, five, six, seven}
one two three
four five six
seven

```

And here is the result of another `\fillrows`, this time with five columns set to their natural width.

```

That's all folks! Until we
meet again ...

```

## References

- [1] Paul W. Abrahams, Karl Berry, and Kathryn A. Hargreaves. *TEX for the Impatient*. Addison-Wesley, 1990. (Available on CTAN in `info/impatient`).
- [2] Peter Wilson. The memoir class for configurable typesetting, 2004. (Available on CTAN in `latex/macros/contrib/memoir`).
- [3] Peter Wilson. Glistings. *TUGboat*, 26(3):253–255, 2005.

◇ Peter Wilson  
18912 8th Ave. SW  
Normandy Park, WA 98166  
USA  
herries dot press (at)  
earthlink dot net



## The Treasure Chest

Editor's note: This is Mark's final column. The editorial board would like to thank Mark for monitoring the CTAN announcements and compiling this collection since 2004.

This is a large but still incomplete list of the new packages posted to CTAN (<http://www.ctan.org>) in 2006, with descriptions either taken from the announcement or researched, then edited for brevity.

Entries are listed alphabetically within CTAN directories. A few entries which the editors subjectively believed to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions!

Hopefully this column and its companions will help to make CTAN a more accessible resource to the T<sub>E</sub>X community. Comments are welcome, as always.

◇ Mark LaPlante  
166 3rd Avenue West  
Grant, AL 35747  
laplante (at) mac dot com

---

### biblio

**alphabib** in `biblio/bibtex/utils`  
A Bash script allowing addition of alphabetical headers into B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> bibliographies.

**ijqc** in `biblio/bibtex/contrib`  
B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> style for the *International Journal of Quantum Chemistry*.

**jneurosci** in `biblio/bibtex/contrib`  
B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> style for the *Journal of Neuroscience*.

**mad2bib** in `biblio/bibtex/utils`  
Scripts to convert MAD to B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> and UTF-8 to L<sup>A</sup>T<sub>E</sub>X ASCII.

**mrcheckbib** in `biblio/bibtex/utils`  
B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> verification using the AMS MRef database.

**munich** in `biblio/bibtex/contrib`  
The Munich 'name (year)' style.

**rsc.bst** in `biblio/bibtex/contrib/misc`  
B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> styles for RSC journals.

**sort-by-letters** in `biblio/bibtex/contrib`  
Some B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> styles for sorting by initial letters.

**Synapsen** in `biblio/bibtex/utils`  
Java program for managing references.

---

### fonts

**arial** in `fonts/urw`  
An Arial clone in Type 1 format.

**classico** in `fonts/urw`  
URW Classico, a clone of Hermann Zapf's Optima.

**cmll** in `fonts`  
Linear logic symbols for Computer Modern.

**emerald** in `fonts`  
Support for Emerald City Fontwerks fonts.

**enpassant** in `fonts/chess`  
Use free chess fonts from [www.enpassant.dk](http://www.enpassant.dk).

**foekfont** in `fonts`  
Fonts and L<sup>A</sup>T<sub>E</sub>X macros to use the title font of the Mads Foek magazine, <http://madsfoek.dk>.

**fonetika** in `fonts`  
Fonts for the Danish phonetic system Dania, based on URW Palladio and Iwona Condensed.

**indic** in `fonts/ps-type1`  
Indic Type 1 fonts for T<sub>E</sub>X converted from public METAFONT sources.

**JustFontItTE** in `fonts/utilities`  
Windows program with the functionality of standard T<sub>E</sub>X utilities such as `pltotf` and `vftovp`. Can create TFM files from PFB, TTF, OTF, and more.

**linearA** in `fonts`  
LinearA script.

**\*otfinst** in `fonts/utilities`  
Installs OpenType fonts for use in (L<sup>A</sup>)T<sub>E</sub>X systems.

**sarabian** in `fonts/archaic`  
Type 1 fonts for the South Arabian script in use from roughly 1000 to 600 BC.

**starfont** in `fonts/ps-type1`  
Support for the StarFont Sans astrological font.

**\*tex-gyre** in `fonts`  
Extensions of the well-known URW and other free fonts, much as Latin Modern is an extension of Computer Modern.

**thaifonts-scalable** in `fonts/thai`  
A collection of Thai scalable fonts.

**winfonts** in `fonts`  
Support for the Windows core fonts.

**xq** in `fonts`  
Fonts and macros for xiangqi (Chinese chess).

---

### graphics

**AddTeX2Eps** in `graphics`  
Use L<sup>A</sup>T<sub>E</sub>X syntax on EPS figures from Mathematica.

**blockdrawmp** in `graphics/metapost/contrib/macros`  
Block diagrams and bond graphs.

**cmarrows** in `graphics/metapost/contrib/macros`  
MetaPost arrows and braces in CM style.

**gapfill** in `graphics`  
Generate L<sup>A</sup>T<sub>E</sub>X picture environments by parsing PostScript files.

**gastex** in `graphics`  
L<sup>A</sup>T<sub>E</sub>X macros to draw automata, networks, etc., in the L<sup>A</sup>T<sub>E</sub>X picture environment.

[graphics/gastex](#)

- 
- m3D** in `graphics/metapost/contrib/macros`  
Three-dimensional drawing with METAPOST.
- MPStoEPS** in `graphics/metapost/contrib/misc`  
METAPOST to EPS converter.
- pedigree-perl** in `graphics/pstricks/contrib`  
Generate pedigrees in T<sub>E</sub>X from CSV files.
- prerex** in `graphics`  
Produce charts of course nodes linked by arrows representing pre- and co-requisites.
- pst-asr** in `graphics/pstricks/contrib`  
Autosegmental representations, used by linguists.
- pst-code** in `graphics/pstricks/doc/code`  
Original code documentation of PSTricks from Timothy Van Zandt.
- pst-coil** in `graphics/pstricks/contrib`  
Coils and zigzags as lines or node connections.
- pst-dbicons** in `graphics/pstricks/contrib`  
Typesetting ER diagrams in declarative style, using standard database terminology.
- pst-eps** in `graphics/pstricks/contrib`  
Exporting pspicture environments.
- pst-fill** in `graphics/pstricks/contrib`  
Filling and tiling of areas and characters.
- pst-grad** in `graphics/pstricks/contrib`  
Colored gradients.
- pst-ob3d** in `graphics/pstricks/contrib`  
Support for basic 3-dimensional objects.
- pst-pdgr** in `graphics/pstricks/contrib/pedigree`  
Draw medical pedigrees with PSTricks and L<sup>A</sup>T<sub>E</sub>X.
- pst-spectra** in `graphics/pstricks/contrib`  
Draw continuum, emission and absorption spectra.
- \*pst-text** in `graphics/pstricks/contrib`  
Typeset text on a path.
- splines** in `graphics/metapost/contrib/macros`  
MetaPost/METAPOST macros for drawing graphs of cubic splines.
- \*textpath** in `graphics/metapost/contrib/macros`  
Typeset text along a path with the help of L<sup>A</sup>T<sub>E</sub>X, so kerns are preserved and 8-bit input is supported.
- uml** in `graphics/pstricks/contrib`  
Writing UML diagrams in L<sup>A</sup>T<sub>E</sub>X.
- 
- language**
- \*arabi** in `language/arabic`  
Arabic and Farsi support in standard L<sup>A</sup>T<sub>E</sub>X via Babel, using several input encodings, including UTF-8. Includes good-quality free fonts.
- bghyph** in `language/hyphenation`  
Bulgarian hyphenation patterns.
- glhyph** in `language/hyphenation`  
Galician hyphenation patterns.
- hrlatex** in `language/croatian`  
Typical setup for Croatian users.
- mkbangtex** in `language/bengali`  
Preprocessor for BangT<sub>E</sub>X.
- magyar** in `language/hungarian/babel`  
Much improved Hungarian definitions for Babel.
- mexican** in `language`  
Modifications to Spanish for Mexican typography.
- mkpattern** in `language/hyphenation/utils`  
T<sub>E</sub>X program for generating hyphenation patterns.
- staves** in `language`  
Fonts and macros for the “magical” Icelandic staves as well as the runic letters used in Iceland.
- thailatex** in `language/thai`  
Babel-based Thai support with fonts.
- 
- macros/generic**
- dirtree** in `macros/generic/`  
Display directory trees.
- 
- macros/latex/contrib**
- abc** in `macros/latex/contrib`  
Support ABC Plus (`abcplus.sourceforge.net`) music notation in L<sup>A</sup>T<sub>E</sub>X.
- active-conf** in `macros/latex/contrib/conferences`  
Class for typesetting papers at the ACTIVE conference on noise and vibration control.
- akktex** in `macros/latex/contrib`  
Collection of new document classes and packages, easing creation of math documents in particular.
- arabicfront** in `macros/latex/contrib/bezoz`  
Number pages in Arabic numerals starting with the front matter.
- authoraftertitle** in `macros/latex/contrib`  
Make title, author and date available after `\maketitle`.
- auto-pst-pdf** in `macros/latex/contrib`  
Wrapper for `pst-pdf` with some `psfrag` features.
- boxhandler** in `macros/latex/contrib`  
Managing boxed objects such as tables and figures, with caption customization.
- bussproofs** in `macros/latex/contrib`  
Construction of proof trees in the style of the sequent calculus and other systems.
- 
- info**
- \*Free\_Math\_Font\_Survey** in `info`  
A survey of the free math fonts available for use with L<sup>A</sup>T<sub>E</sub>X. Vietnamese translation in the `vn` sub-directory.
- MemoirChapStyles** in `info`  
Memoir chapter style showcase.
- MFwL** in `info`  
*Making Friends with L<sup>A</sup>T<sub>E</sub>X*, a L<sup>A</sup>T<sub>E</sub>X introduction.
- l2picfaq** in `info/l2picfaq/german`  
Questions about L<sup>A</sup>T<sub>E</sub>X and pictures, and answers in the form of sample code.
- \*visualFAQ** in `info`  
A visual interface to the UK T<sub>E</sub>X FAQ.

- \*cellspace** in `macros/latex/contrib`  
Minimal spacing of table cells to avoid touching `\hlines`.
- centernot** in `macros/latex/contrib/oberdiek`  
Horizontally center a `\not` symbol on its argument.
- chessboard** in `macros/latex/contrib`  
Typeset chessboards.
- classicthesis** in `macros/latex/contrib`  
Thesis style in homage to *The Elements of Typographic Style*.
- cool** in `macros/latex/contrib`  
Content Oriented L<sup>A</sup>T<sub>E</sub>X, retaining mathematical meaning in addition to the typesetting instructions.
- coollist** in `macros/latex/contrib`  
Manipulation of lists.
- coolstr** in `macros/latex/contrib`  
Manipulation of strings (sequences of characters, not tokens).
- cooltooltips** in `macros/latex/contrib`  
Associate a pop-up window and tooltip with a PDF hyperlink.
- coverpage** in `macros/latex/contrib`  
Supplement scientific papers with a cover page.
- dateiliste** in `macros/latex/contrib`  
Typeset the file list (like `\listfiles`), and more.
- digiconfigs** in `macros/latex/contrib`  
Small package for easily writing ‘configurations’, i.e., special  $\circ + \bullet$  matrices.
- disser** in `macros/latex/contrib`  
Document class and templates for dissertations in Russian.
- draftwatermark** in `macros/latex/contrib`  
Typeset a light gray watermark of user-defined text on the first page or every page of a document.
- duerer-latex** in `macros/latex/contrib`  
L<sup>A</sup>T<sub>E</sub>X support for the *duerer* fonts.
- dvdcoll** in `macros/latex/contrib`  
Typeset DVD (or similar) collection overviews.
- dyntrree** in `macros/latex/contrib`  
Typeset Dynkin Tree Diagrams from group theory.
- eskdX** in `macros/latex/contrib`  
Implement Russian standards for designers.
- everypage** in `macros/latex/contrib`  
Provides hooks to be run on each page of a document.
- exceltex** in `macros/latex/contrib`  
Access Excel files from L<sup>A</sup>T<sub>E</sub>X.
- extpfeil** in `macros/latex/contrib`  
Extensible arrows and commands to create new ones.
- faktor** in `macros/latex/contrib`  
Typeset quotient structures.
- flippdf** in `macros/latex/contrib`  
Produce a “mirrored” version of the document, for typesetting on transparent film.
- forloop** in `macros/latex/contrib`  
Provides the command `\forloop`.
- gmdoc** in `macros/latex/contrib`  
Documenting packages with minimal markup and hyperlinks. Needs the other new packages `gmiflink`, `gmverb`, and `gmutils` from the same author.
- gnuplottex** in `macros/latex/contrib`  
Generate and include Gnuplot (<http://gnuplot.info>) graphs in L<sup>A</sup>T<sub>E</sub>X.
- gu** in `macros/latex/contrib`  
Typeset crystallographic group-subgroup-schemes in Bärnighausen formalism.
- hyperxmp** in `macros/latex/contrib`  
Embed XMP metadata within a L<sup>A</sup>T<sub>E</sub>X document.
- icsv** in `macros/latex/contrib/conferences`  
Class for typesetting papers at the ICSV conference.
- isodoc** in `macros/latex/contrib`  
Producing customizable letters, invoices, etc.
- labelcas** in `macros/latex/contrib`  
Tests and branches depending on label existence.
- lewis** in `macros/latex/contrib`  
Rudimentary support for drawing Lewis Structures.
- lingtrees** in `macros/latex/contrib`  
Linguistic tree macros and preprocessor.
- lsc** in `macros/latex/contrib`  
Typesetting Live Sequence Charts, similar to `msc`.
- marginnote** in `macros/latex/contrib`  
Marginal notes where `\marginpar` fails.
- minipage-marginpar** in `macros/latex/contrib`  
Allow `\marginpar` commands outside minipages; an alternative approach to `marginnote`.
- moderncv** in `macros/latex/contrib`  
Typesetting modern curriculum vitae, in classic and casual styles.
- nih** in `macros/latex/contrib`  
Class for grant applications to the US government National Institutes of Health.
- noitcruL** in `macros/latex/contrib`  
Underlining with italic correction (in math).
- opteng** in `macros/latex/contrib`  
Manuscript template for *SPIE Optical Engineering* and *OE Letters*, including page length estimates.
- pandora-latex** in `macros/latex/contrib`  
L<sup>A</sup>T<sub>E</sub>X support for the *pandora* fonts.
- papercdcase** in `macros/latex/contrib`  
Origami-style folding paper CD cases.
- pauldoc** in `macros/latex/contrib`  
Helpers for German-language L<sup>A</sup>T<sub>E</sub>X package documentation.
- pax** in `macros/latex/contrib`  
Avoid pdfL<sup>A</sup>T<sub>E</sub>X’s stripping of PDF annotations in included PDF files, via an external Java program.
- pinlabel** in `macros/latex/contrib`  
Attach formatted T<sub>E</sub>X labels to new or existing figures and diagrams.
- \*pracjourn** in `macros/latex/contrib`  
Document class for *The PracT<sub>E</sub>X Journal*, <http://tug.org/pracjourn>.

- punk-latex** in `macros/latex/contrib`  
 L<sup>A</sup>T<sub>E</sub>X support for the **punk** fonts.
- qtree** in `macros/latex/contrib`  
 Typeset tree diagrams, especially for linguistics.
- randbild** in `macros/latex/contrib`  
 Put small plots of curves in the margin.
- \*randtext** in `macros/latex/contrib`  
 Typeset obfuscated text to foil spam email address harvesters. Supersedes `switcheml`.
- ratex** in `macros/latex/contrib`  
 Producing German legal documents, e.g., lawsuits.
- recipecard** in `macros/latex/contrib`  
 Typesets recipes into note card sized boxes.
- robustcommand** in `macros/latex/contrib`  
 Variant of `\DeclareRobustCommand` which checks if the command has already been defined.
- screenplay** in `macros/latex/contrib`  
 Screenplay formatting as recommended by the Academy of Motion Picture Arts and Sciences.
- seqsplit** in `macros/latex/contrib`  
 Splitting long sequences of letters without spaces, such as DNA, RNA, proteins, etc.
- showexpl** in `macros/latex/contrib`  
 Show L<sup>A</sup>T<sub>E</sub>X code and typeset result from one source.
- sikumna** in `macros/latex/contrib/lyx`  
 Extended LyX layout style for use in Sikumna, <http://www.sikumuna.co.il>.
- simplewick** in `macros/latex/contrib`  
 Drawing Wick contractions above and below expressions in math mode.
- spotcolor** in `macros/latex/contrib`  
 Include spot colors in pdfL<sup>A</sup>T<sub>E</sub>X output.
- stackel** in `macros/latex/contrib/oberdiek`  
 Adds optional argument to `\stackrel` for including something below the relation, and defines `\stackbin` for binary symbols.
- stellenbosch** in `macros/latex/contrib`  
 Typesetting dissertations, theses and reports of the University of Stellenbosch, South Africa.
- \*sudokubundle** in `macros/latex/contrib`  
 Typesetting, solving, and creating Sudoku puzzles; over 50 puzzles are included.
- sugconf** in `macros/latex/contrib/conferences`  
 Document class for SAS user group proceedings.
- susy** in `macros/latex/contrib`  
 Support for supersymmetry-related documents.
- svn-multi** in `macros/latex/contrib`  
 Typesetting Subversion keywords, including documents with multiple source files.
- syntrace** in `macros/latex/contrib`  
 Support for tracing trees created with the `syntree` package.
- tabto** in `macros/latex/contrib`  
 Tabbing to fixed positions in a paragraph.
- technica** in `macros/latex/contrib`  
 A suite of packages for typesetting literary texts.
- telprint** in `macros/latex/contrib/oberdiek`  
 Formatting German telephone numbers.
- tengwarscript** in `macros/latex/contrib`  
 Mid-level access to the `tengwar` fonts with good default output.
- thesis-titlepage-fhAC** in `macros/latex/contrib`  
 Title page for Fachschule Aachen theses.
- thuthesis** in `macros/latex/contrib`  
 Thesis template for Tsinghua University.
- titlepage-uni-dortmund** in `macros/latex/contrib`  
 Title page layout for the University of Dortmund.
- toptesi** in `macros/latex/contrib`  
 Bundle for thesis typesetting in Italy, supporting any language.
- wordlike** in `macros/latex/contrib`  
 Simulates standard word processor layout and fonts.
- xifthen** in `macros/latex/contrib`  
 Extended if-then features.
- xyling** in `macros/latex/contrib`  
 Draw syntactic trees and other linguistic constructs, based on X<sub>Y</sub>-pic.
- xytree** in `macros/latex/contrib`  
 Tree drawing package using X<sub>Y</sub>-pic.
- 
- macros/latex/exptl**
- \*biblatex** in `macros/latex/exptl`  
 Reimplementation of the bibliographic facilities provided by L<sup>A</sup>T<sub>E</sub>X in conjunction with B<sub>I</sub>B<sub>T</sub>E<sub>X</sub>: B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> is used only for sorting and producing labels, and bibliography formatting is controlled entirely by T<sub>E</sub>X macros, not B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> style files. Still experimental.
- xbase** in `macros/latex/exptl`  
 Provides L<sup>A</sup>T<sub>E</sub>X3 packages `xparse` and `template`.
- 
- macros/omega**
- fontch** in `macros/omega/latex/contrib`  
 Typeset Malayalam using Omega.
- 
- macros/xetex**
- euenc** in `macros/xelatex/latex`  
 Unicode font encoding definitions for X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X.
- grchyp** in `macros/xetex/hyphenation`  
 Unicode hyphenation patterns for ancient Greek.
- ifxetex** in `macros/xetex/latex`  
 Provides the `\ifxetex` conditional.
- sanhyph** in `macros/xetex/hyphenation`  
 Unicode hyphenation patterns for Prakrit and Sanskrit in Devanagari, Bengali, Kannada, Malayalam and Telugu scripts.
- xetex-greek** in `macros/xetex/hyphenation`  
 Standard Greek hyphenation patterns adapted for X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X.
- \*xlxtra** in `macros/xetex/latex`  
 Additional L<sup>A</sup>T<sub>E</sub>X features for X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X.

---

**support**
**cms4talks in support**

A Java-based content management system for talks written in L<sup>A</sup>T<sub>E</sub>X.

**dinbrief-gui in support**

Graphical interface for the dinbrief T<sub>E</sub>X package.

**\*escapeTeXt in support**

Modular Python program to massage plain text so it can likely pass through L<sup>A</sup>T<sub>E</sub>X.

**gentabtex in support**

Table rendering engine written in Python.

**orderer in support**

Reorder references in a L<sup>A</sup>T<sub>E</sub>X document by order of citation.

**pkfix in support**

Replaces resolution-dependent bitmapped fonts in a dvips-produced PostScript file with the corresponding resolution-independent vector fonts.

**rfile in support**

The Ruby font installer library (RFILE) attempts to manage installation of T<sub>E</sub>X fonts.

**word-to-latex in support**

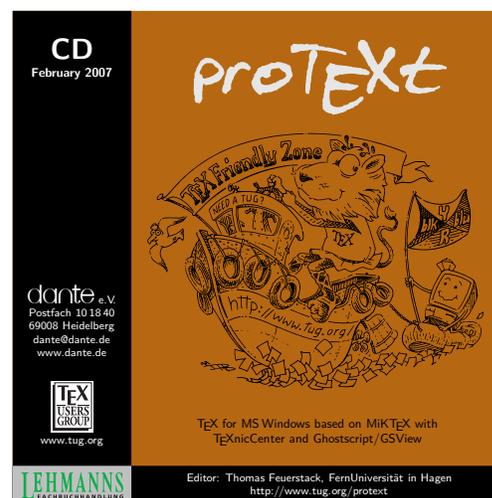
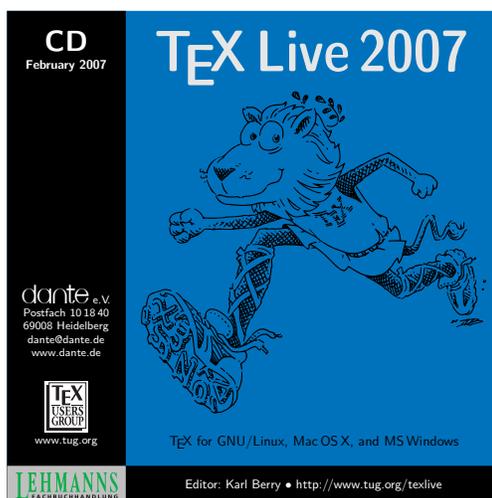
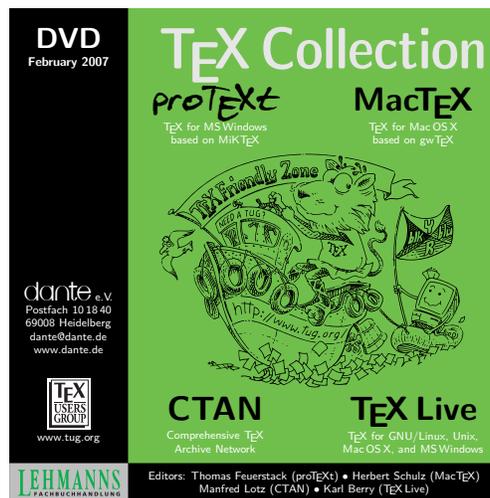
Windows program to convert Microsoft Word documents to L<sup>A</sup>T<sub>E</sub>X or XML.

---

**systems**
**visualtex in systems/win32**

Visual T<sub>E</sub>X text editor.

We end this installment of Treasure Chest with the covers from the newly released T<sub>E</sub>X Collection 2007, now being mailed to user group members. See <http://tug.org/texcollection> for more information.





## paper $\TeX$ : Creating newspapers using $\LaTeX 2\epsilon$

Ignacio Llopis Tortosa and  
María José Castro Bleda

### Abstract

The first author has been working in an Internet newspaper office for a year. He was asked to create a special printable edition of an on-line newspaper. The idea was to get an easy-to-print newspaper containing the main daily news. The solution involved two things: a new  $\LaTeX$  class, that we called paper $\TeX$ , and a Perl-based system that gets all the information from the database and composes a new  $\LaTeX$  document.

### 1 Overall

The final system consists of a Perl script which controls the entire process, from the data collection through the document compilation using PDF $\LaTeX$ . There is also a web wizard that lets the user set up the news and the information that he or she would like to show at the newspaper. The configuration file stores the SQL queries which take the information from the database, thus it can be kept for a long time.

Together with this Perl script, there is a new  $\LaTeX 2\epsilon$  class called paper $\TeX$ ,<sup>1</sup> specially created for this purpose. This class provides many macros to create a document in a newspaper style. It has a front page and the inside part that contains all the news we would like to include. Every piece of news appears just below the one before. Headings use the entire page width and the text can be split into several columns. paper $\TeX$  also provides commands for adding outstanding titles, images, timestamps, etc.

Our last page includes an automatically generated Sudoku, a cultural agenda and a humor drawing. Finally, the system is scheduled to run several times a day creating the PDF version of the newspaper so that users can easily download it. Neither manual design help nor computer programming help are needed. The system determines where images can be placed, if there is enough place for a new piece of text and so on.

You can see a comparison between the newspaper web site and the output generated automatically by the Perl script in Figure 1. The main news items are the same as the ones in the web site and their contents appear in the inner pages of the PDF newspaper.

### 2 Why $\LaTeX$ ?

When creating newspapers it is well known you have many high specific applications which do the work. Most of these applications let you create a newspaper and publish it in several ways at the same time: print version, HTML version, etc. *Protec* has three different publishing systems called Millenium, Edicomp and Arcano.<sup>2</sup> *Unisys* has a system called Hermes.<sup>3</sup> Of course there is also Adobe InDesign and Quark Express but this system does something different than designing layouts.

Why use  $\LaTeX$ ? First of all, as  $\LaTeX$  users, we wanted to use it for handling this big project. Secondly,  $\LaTeX$  is a tag based language which lets you create documents without taking care of design — quite the opposite of Quark Express and InDesign. The idea was for the system to take care of the design, from simple text input. Thirdly,  $\LaTeX$  is a free and open source application and a huge number of packages have been written for it.

Once the system was proposed, we had to prove that it was a good option and it could be used for creating newspapers. We created several documents using the `multicol` package, and we added images and capital letters using the `lettrine` and `graphics` packages. We also designed documents using the `textpos` package, placing items in any desired spot. By this time, it was clear that  $\LaTeX$  could do the job.

### 3 paper $\TeX$ : A new class for creating newspapers

The first idea was to look for something similar. Has anyone tried this before? We posted the question in some forums and mailing lists with disappointing results. But someone told us about the `newsltr` class and the  $\TeX$  capability to handle newsletters. This class was a good starting point but we had some problems. The class was created for plain  $\TeX$  and that was a problem for including  $\LaTeX$  packages, and other matters such as embedding Spanish characters directly or correct hyphenation.

So we decided to develop a new  $\LaTeX$  class for creating newspapers. This class provides commands

<sup>1</sup> Freely available from CTAN in <http://www.ctan.org/tex-archive/macros/latex/contrib/papertex>

<sup>2</sup> <http://www.protecmedia.com/>

<sup>3</sup> <http://www.unisys.com/>



Figure 1: The newspaper web site and the self-generated PDF using paperT<sub>E</sub>X and Perl.

to create a new paperT<sub>E</sub>X journal: from the front page until the last.

To set up a new document, you have to load the class as usual and use its own commands to define the contents. paperT<sub>E</sub>X also includes many style macros which the user can customize as desired: font sizes and styles, colors, headings, etc.

### 3.1 The front page

The front page is quite individualized since it was designed using the `textpos` package that has the capability to place things at absolute positions on the page. We did this because the front page needed to have a different style from the rest of the newspaper, being the first thing a reader sees. It includes a main image or photo, three news blocks, an index which links to the inner news, the weather forecast for a locality and some information about the editor. It has also a banner heading like every newspaper does.

Just after the `\begin{document}`, you can start filling the front page inside its own environment. The main image and each of the main news items have their own commands which get all the information: image path and caption, heading, subheading, opening paragraph, section and time stamp. The table of contents has its own environment and the only thing we have to do is to add entries using the `\indexitem` macro. This command requires two parameters: a short text and a reference to a piece of

news inside the newspaper which allows paperT<sub>E</sub>X to calculate the exact page.

The weather forecast block has three different positions to specify three different weather conditions. Each weather item has an image, maximum and minimum temperatures and a short description. Finally, the editor block includes the usual contact names, email, logo, etc.

As mentioned above, paperT<sub>E</sub>X includes a set of macros that you can redefine to change default format and layout. For example, redefining the front page logo is as simple as this:

```
\renewcommand{\logo}{
\mylogo{\includegraphics[width=\textwidth]{img}}
}
```

The `\mylogo` command removes the paperT<sub>E</sub>X default logo and changes heading elements' positions to make your new logo fit well. Other style aspects are even easier to redefine. You can find all of them in the paperT<sub>E</sub>X manual [1].

### 3.2 The news pages

Once you have introduced all the front page information, the next thing to do is to include all the news in your new paperT<sub>E</sub>X. News items are easy to include and they can have different shapes: *news*, *editorial* and *short news*. Each one of these types has its own environment definition.

To include a normal news item you use the *news* environment, which needs five parameters: number

```

\begin{news}{3}
{...NEWS HEADING...}
{...News subheading...}
{SPORTS}
{1}
\authorandplace{Name Surname}{Place}
\image{img}{Image caption}
\noindent\timestamp{08:25}
Lorem ipsum dolor sit amet, ...
... platea dictumst.
\end{news}

```

Figure 2: Example of a compilation of news items.

of columns, heading, subheading, section name and reference id. The main news text must appear inside the environment and we can also use the following commands within the text in order to add other useful information:

- `\authorandplace`: inserts the name of the editor and where the news happened. To be used at the beginning of the text.
- `\timestamp`: inserts the time and a separator just before the text. It should be used at the beginning of the text.
- `\image`: inserts an image within the text. Since the `multicol` package does not support floating elements, this macro inserts the image only if there is enough space, otherwise you can get images outside the page boundaries.
- `\columntitle`: inserts a single column title or heading, using one of five different shapes via the `fancybox`<sup>4</sup> package: `shadowbox`, `doublebox`, `ovalbox`, `Ovalbox` and `lines`.
- `\expandedtitle`: similar to `\columntitle` except the text extends across the entire page, above all the news columns.

Using all these commands a news source code and its respective result would look like the example in Figure 2.

Editorial news and short news require fewer parameters but are generally similar to the news environment. The `paperTEX` manual [1] has more information.

Once the news items are included, when compiling `paperTEX` will create the corresponding PDF

<sup>4</sup> You can find more information in the `fancybox` package manual available on CTAN.



bookmarks inside each section. In order to generate a new group of items we can use the `\newsection` command which takes the section name as a parameter. From this point, all news PDF bookmarks will be grouped under this section name. Another useful and simple command is `\newsep`, which draws a thin line between two items as a separator.

#### 4 The core system

When the `paperTEX` class was finished, we implemented a new system to carry out the entire newspaper creation process. We decided to use Perl as programming language because we are familiar with it, and because Perl has a lot of modules freely available on CPAN.<sup>5</sup>

Before coding, it was very important to select which news items have to be included each time that `paperTEX` is run. The idea was to define these items using the SQL queries which get all the information from the database; thus, `paperTEX` would not need to be configured each time. The only time when it is necessary to modify something is if the editor of the newspaper wants to change the source of a piece of news or add or remove particular content.

First of all, a configuration file was created for the Perl script to get all SQL queries, execute them, and extract the useful information from all fields. It worked well enough, but was not very user-friendly. Therefore we decided to create a web wizard which gets all the information from the user. This application also lets the users change any parameter or SQL instruction. After this, we asked the editor of the newspaper to give us the list of news he wanted

<sup>5</sup> The Comprehensive Perl Archive Network (<http://www.cpan.org/>).

to appear in the paper $\TeX$  newspaper; then we ran the web wizard and filled in all SQL queries and database information. This config file has not been changed in more than six months and is working well.

The news items are stored in HTML format, so we created a script to get all the items included in the config file, convert them to  $\LaTeX$ , create a new paper $\TeX$  document and, in the end, compile it using PDF $\LaTeX$ . We also decided to add a final page including an events list, an automatically generated Sudoku and a humour drawing. To develop this script some Perl modules were used. For example:

- `HTML::Latex` by Peter Thatcher, which converts any HTML text to  $\LaTeX$  in every way you like. This module was very important for us because all the news items were stored in HTML format just as they were published on the website.
- `Weather::Com` by Thomas Schnuecker, which retrieves the weather forecast included on the paper $\TeX$  front page.
- `Games::Sudoku::Component` by Kenichi Ishigaki, which creates and solves a new Sudoku each time paper $\TeX$  is created. This module was very useful (after few modifications) with the `sudoku`  $\LaTeX$  package by Paul Abraham.

When executing the final script, we had some problems composing the newspaper front page, because the database contained texts that could be longer than the space available. To prevent text overflow, a function to trim texts at certain position (always after a period) was implemented. Some tests in order to get the right threshold were made. Although the perfect solution would have been to have short texts, the system is working quite well as it stands. Another problem was that, at first, the editor of the newspaper office did not want to hyphenate headings in the front page. So we tried to avoid hyphenation, but we got very bad results when there were long words which ran off the page.

Finally, we installed  $\LaTeX$  and the script on the company server and use a *cron*-based application to schedule execution three times a day. We also linked the PDF output in the newspaper web site.

## 5 Conclusions

The goal of this project was to create a special printable edition of an on-line Spanish newspaper. The solution involved a new  $\LaTeX$  class, that we called paper $\TeX$ , and a Perl-based system that automatically extracts, composes and creates a final PDF

file. This system works off-line and does not need any human assistance in order to generate the publication several times a day. It is also worth emphasizing that this system is working today with the same configuration as in August 2006. The final application is running for a Spanish on-line newspaper called `Panorama-Actual.es` and it creates a publication which name is *papelDigital*. You can freely access them through <http://www.panorama-actual.es/pdigital/>.

Finally, there is another system in Spain which does something similar to the application described in this document. It appeared at the beginning of this project but, as far as we know, it does not use  $\LaTeX$  at all. The publication is called *24 Horas*<sup>6</sup> and it is used by EL PAIS, one of the most widely-circulated newspapers in Spain.

## Acknowledgments

We would like to gratefully acknowledge the support of Robert Fuster, from the Polytechnic University of Valencia (Spain). Thanks for reading the various drafts and making useful suggestions.

We would also especially like to thank the members and regular participants on the Spanish  $\TeX$  user list *es-tex* list. They have contributed with comments during the whole development process. Finally, the first author would like to thank Mar for always being *there*.

## References

- [1] Ignacio Llopis Tortosa. paper $\TeX$  class: Creating newspapers using  $\LaTeX$ . Technical Report DSIC-II/03/07, Department Sistemas Informáticos y Computación, Polytechnic University of Valencia, 2006.

- ◊ Ignacio Llopis Tortosa  
Department Sistemas Informáticos  
y Computación  
Univ. Politécnic de Valencia  
Camino de Vera s/n  
46071 Valencia, Spain  
lloptor (at) gmail dot com
- ◊ María José Castro Bleda  
Department Sistemas Informáticos  
y Computación  
Univ. Politécnic de Valencia  
Camino de Vera s/n  
46071 Valencia, Spain  
mcastro (at) dsic dot upv dot es

<sup>6</sup> *24 Horas* is free to download through <http://www.elpais.com/24horas/>

# L<sup>A</sup>T<sub>E</sub>X News

Issue 17, December 2005

## Project licence news

The L<sup>A</sup>T<sub>E</sub>X Project Public License has been updated slightly so that it is now version 1.3c. In the warranty section the phrase “unless required by applicable law” has been reinstated, having got lost at some point. Also, it now contains three clarifications: of the difference between “maintained” and “author-maintained”; of the term “Base Interpreter”; and when clause 6b and 6d shall not apply.

Following requests, we now also provide the text of the licence as a L<sup>A</sup>T<sub>E</sub>X document (in the file `lpp1.tex`). This file can be processed either as a stand-alone document or it can be included (without any modification) into another L<sup>A</sup>T<sub>E</sub>X document, e.g., as an appendix, using `\input` or `\include`.

## New guide on font encodings

Way back in 1995 work was started on a guide to document the officially allocated L<sup>A</sup>T<sub>E</sub>X font encoding names. However, for one reason or another this guide (named *L<sup>A</sup>T<sub>E</sub>X font encodings*) was, until now, not added to the distribution. It describes the major 7-bit and 8-bit font encodings used in the L<sup>A</sup>T<sub>E</sub>X world and explains the restrictions required of conforming text font encodings. It also lists all the ‘encoding specific commands’ (the LICR or L<sup>A</sup>T<sub>E</sub>X Internal Character Representation) for characters supported by the encodings `OT1` and `T1`.

When the file `encguide.tex` is processed by L<sup>A</sup>T<sub>E</sub>X, it will attempt to typeset an encoding table for each encoding it describes. For this to be possible, L<sup>A</sup>T<sub>E</sub>X must be able to find `.tfm` files for a representative example font for each encoding. If L<sup>A</sup>T<sub>E</sub>X cannot find such a file then a warning is issued and the corresponding table is omitted.

## Robust commands in math

The font changing commands in text-mode have been robust commands for years, but the same has not been true for the math versions such as `\mathbf`. While the math-mode commands worked correctly in section heads, they could cause problems in other places such as index entries. With this release, these math-mode commands are now robust in the same way as their text-mode counterparts.

## Updates of required packages

Several of the packages in the tools bundle have been updated for this release.

The `xspace` package has some new features. One is an interface for adding and removing the exceptions it knows about and another is that it works with active characters. These remove problems of incompatibility with the `babel` system.

In *L<sup>A</sup>T<sub>E</sub>X News 16* we announced that some packages might begin to take advantage of  $\epsilon$ -T<sub>E</sub>X extensions on systems where these are available: and the latest version of `xspace` does just that. Note also that `fixltx2e` will make use of the facilities in  $\epsilon$ -T<sub>E</sub>X whenever these are present (see below).

The `calc` package has also been given an update with a few extra commands. The commands `\maxof` and `\minof`, each with two brace-delimited arguments, provide the usual numeric max and min operations. The commands `\settototalheight` and `\totalheightof` work like `\settoheight` and `\heightof`. There are also some internal improvements to make `calc` work with some more primitive T<sub>E</sub>X constructs, such as `\ifcase`.

The `varioref` package has acquired a few more default strings but there are still a number of languages for which good strings are still missing.

The `showkeys` package has also been updated slightly to work with more recent developments in `varioref`. Also, it now provides an easy way to define the look of the printed labels with the command `\showkeyslabelformat`.

## Work on L<sup>A</sup>T<sub>E</sub>X fixes

The package known as `fixltx2e` has three new additions. A new command `\textsubscript` has been added as a complement to the command `\textsuperscript` in the kernel. Secondly, a new form of `\DeclareMathSizes` that allows all of its arguments to have a dimension suffix. This means you can now use expressions such as `\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}`.

The third new addition is the robust command `\TextOrMath` which takes two arguments and executes one of them when typesetting in text or math mode respectively. This command also takes advantage of  $\epsilon$ -T<sub>E</sub>X extensions if available; more specifically, when the  $\epsilon$ -T<sub>E</sub>X extensions are available, it does not destroy kerning between previous letters and the text to be

typeset. The command is also used internally in `fixltx2e` to resolve a problem with `\fnsymbol`.

Also, further work has been done on reimplementing the command `\addpenalty`, which is used internally in several places: we hope it is an improvement!

## The graphics bundle

The graphics bundle now supports the `dvipdfmx` post-processor and Jonathan Kew's XETEX program. By support we mean that the graphics packages recognize the new options `xetex` and `dvipdfmx` but we do not distribute the respective driver files.

This leads elegantly to a description of the new policy concerning such driver files in the graphics bundle. Most driver files for our graphics packages are maintained by the developers of the associated post-processor or T<sub>E</sub>X programs. The teams developing these packages are working very hard: their rapid development offers a stark contrast to the current schedule of L<sup>A</sup>T<sub>E</sub>X releases. It is therefore no longer practical for the L<sup>A</sup>T<sub>E</sub>X Team to be responsible for distributing the latest versions of these driver files.

Therefore the installation files for graphics have been split: there is now `graphics.ins` to install the package files and `graphics-drivers.ins` for the driver files (located in `drivers.dtx`). There is no need to install all those provided in the file `drivers.dtx`.

Please also note that, as requested by the maintainers of PStricks, we have removed the package `pstcol` as current versions of PStricks make it obsolete.

## Future development

The title of this section is a little misleading as it actually describes *current* development. In 1998 the `expl3` bundle of packages was put on CTAN to demonstrate a possible L<sup>A</sup>T<sub>E</sub>X3 programming environment. These packages have been lying dormant for some time while the L<sup>A</sup>T<sub>E</sub>X Project Team were preoccupied by other things such as developing the experimental packages `xor`, `template`, etc., (and also writing that indispensable and encyclopaedic volume, *The L<sup>A</sup>T<sub>E</sub>X Companion – 2nd edition*).

In October 2004 work on this code base was resumed with the goal of some day turning it into a kernel for L<sup>A</sup>T<sub>E</sub>X3. This work can now also make full use of the widely accepted  $\epsilon$ -T<sub>E</sub>X extensions. Currently two areas are central to this work.

- Extending the kernel code of L<sup>A</sup>T<sub>E</sub>X3.
- Converting the experimental packages such as `xor`, `template` to use the new syntax internally.

Beware! Development of `expl3` is happening so fast that the descriptions above might be out of date when you read this! If you wish to see what's going on then go to <http://www.latex-project.org/code.html> where you can download fully working code (we hope!).

# Practical T<sub>E</sub>X 2006

L<sup>A</sup>T<sub>E</sub>X workshop: July 25–28, 2006 ■ Conference: July 30–August 1, 2006  
Rutgers, the State University of New Jersey (Busch Campus)  
Piscataway, New Jersey, USA

## Sponsors

**T<sub>E</sub>X Users Group ■ Rutgers ■ DANTE e.V.**

Carleton Production Centre ■ Design Science ■ O'Reilly Media, Inc.  
PCT<sub>E</sub>X, Inc. ■ River Valley Technologies

*Thanks also to all the speakers, teachers, and participants, without whom there would be no conference. Special thanks to Barbara Mastrian and Steve Peter for all their help with local coordination, and Gerree Pecht, for providing additional local information. Also to Wendy McKay for organizing the Mac OS X gatherings and Duane Bibby for the (as always) excellent drawing.*

## Conference committee

Barbara Mastrian ■ Cheryl Ponchin ■ Karl Berry ■ Robin Laakso ■ Steve Peter ■ Sue DeMeritt

## Workshop participants

Abbes Bahri, Rutgers  
Yael Goldberg, Rutgers  
Alice Leonhardt, Rutgers  
Barbara Mastrian, Rutgers  
Jaime Moore, Decision & Sensor Analytics  
Gerree Pecht, Princeton University  
Christina Polans, IEEE

Ira Polans, IEEE  
Sam Roze, Princeton University  
Chirag Shah, IEEE  
Caroline Sheedy, Carnegie-Mellon University  
David Starbuck, IEEE  
Leszita Townsend, Rutgers  
Michele Turansick, Institute of Advanced Study

## Conference participants

William Adams, Mechanicsburg, PA  
Leila Akhmadeeva, Bashkir Medical Univ., Russia  
Bob Alps, Towers Perrin, Chicago, IL  
Tim Arnold, SAS  
Kaveh Bazargan, Focal Image Ltd  
Barbara Beeton, American Mathematical Society  
Karl Berry, T<sub>E</sub>X Users Group  
Jon Breitenbucher, College of Wooster  
Elizabeth Dearborn, Buffalo, NY  
Sue DeMeritt, Center for Communications  
Research, La Jolla, CA  
Ron Fehd, Centers for Disease Control and  
Prevention  
Frances Felluca, INFORMS  
Peter Flom, National Development and Research  
Institutes  
Peter Flynn, Silmaril Consultants  
Federico Garcia, University of Pittsburgh  
Steve Grathwohl, Duke University Press  
Barbara Hamilton, Center for Communications  
Research, Princeton, NJ  
Jim Hefferon, St. Michael's College  
Troy Henderson, US Military Academy  
Klaus Höppner, DANTE e.V.  
Ned Hummel, University of Nebraska  
Mirko Janc, INFORMS  
Jonathan Kew, SIL International

Richard Koch, University of Oregon  
Martha Kummerer, University of Notre Dame  
Robin Laakso, T<sub>E</sub>X Users Group  
Jenny Levine, Duke University Press  
Wendy McKay, Caltech  
Andrew Mertz, Eastern Illinois University  
Stephen Moye, American Mathematical Society  
Bob Neveln, Widener University  
Don Pellegrino, DuPont  
Steve Peter, Beech Stave Press  
Christina Polans, IEEE  
Ira Polans, IEEE  
Cheryl Ponchin, Center for Communications  
Research, Princeton, NJ  
John Rorem, Duke University Press  
Sam Roze, Princeton University  
Herbert Schulz, Naperville, Illinois  
Heidi Sestrich, Carnegie-Mellon University  
Chirag Shah, IEEE  
William Slough, Eastern Illinois University  
David Starbuck, IEEE  
David Tellet, Alexandria, VA  
Larry Thomas, Saint Peter's College  
Boris Veytsman, George Mason University &  
AES ITT Industries  
David Walden, E. Sandwich, MA  
Alan Wetmore, US Army

A general report on Practical T<sub>E</sub>X 2006 appeared in *The PracT<sub>E</sub>X Journal*, issue 2006-3:  
<http://tug.org/pracjourn/2006-3/practex06/>

Here we include a few photos from the conference. More photos appear in the report and at  
<http://tug.org/practicaltex2006/photos/>



Seated: William Slough, Martha Kummerer, Barbara Hamilton, Cheryl Ponchin, Kaveh Bazargan (sprawled), Wendy McKay, Frances Fellucca, John Rorem.

Middle row: Leila Akhmadeeva, Boris Veytsman, Jenny Levine, Heidi Sestrich, Alan Wetmore, Sue DeMeritt, Steve Peter, Jonathan Kew, Peter Flom, Larry Thomas, David Walden, Jim Hefferon, Stephen Moye.

Back row: Bob Neveln, Elizabeth Dearborn, Barbara Beeton, Dick Koch, Andrew Mertz, Herb Schultz, Ron Fehd, Ned Hummel, Jon Breitenbucher, Karl Berry, Klaus Höppner, Steve Grathwohl, Don Pellegrino, Peter Flynn, Mirko Janc.



Barbara Mastrian, local coordinator extraordinaire.



Boris Veytsman, Kaveh Bazargan, Leila Akhmadeeva, Robin Laakso, and Karl Berry.



Pigeons at the top of the Empire State Building.

# Practical T<sub>E</sub>X 2006 — schedule

<b>Tuesday July 25– Friday July 28</b>	9 am–5 pm	<b>L<sup>A</sup>T<sub>E</sub>X workshop</b> <i>led by Sue DeMeritt &amp; Cheryl Ponchin</i>	
	8–9 am	<i>registration</i> (on Tuesday, at Rutgers)	
	10:15–10:30 am	<i>break</i>	
	12–1 pm	<i>lunch</i>	
	3–3:15 pm	<i>break</i>	
<b>Saturday July 29</b>	5–7 pm	<i>registration &amp; reception</i> (at the Clarion hotel, Windsor Ballroom)	
	8–9 am	<i>registration</i> (at Rutgers)	
<b>Sunday July 30</b>	9 am	Karl Berry, T <sub>E</sub> X Users Group	<i>Welcome &amp; introductions</i>
	9:20 am	Barbara Beeton, AMS & TUG	keynote address: <i>How to create a T<sub>E</sub>X journal: A personal journey</i>
	10:20 am	<i>break</i>	
	10:30 am	Peter Flom, NDRI	<i>L<sup>A</sup>T<sub>E</sub>X for social scientists and other people who think they don't need it</i>
	11:10 am	Jim Hefferon, St. Michael's College	<i>L<sup>A</sup>T<sub>E</sub>X resources</i>
	11:50 am	Boris Veytsman, George Mason Univ. & AES ITT Industries	<i>Design of presentations: Notes on principles and T<sub>E</sub>X implementation</i>
	12:30 pm	<i>lunch</i>	
	1:40 pm	Alan Wetmore, US Army	<i>T<sub>E</sub>X and after dinner speaking</i>
	2:20 pm	Steve Peter, Beech Stave Press	<i>Fonts, typefaces, glyphs &amp; sorts</i>
	3 pm	<i>break</i>	
	3:10 pm	Klaus Höppner, DANTE e.V. & TUG	<i>Creation of a PostScript Type 1 logo font with MetaType1</i>
	3:50 pm	William Adams, Mechanicsburg, PA	<i>TypeSpec v2: Typesetting font specimens</i>
	4:30 pm	q & a, Birds of a Feather	
<b>Monday July 31</b>	9 am	Ned Hummel, University of Nebraska	<i>Common macro pitfalls and how to avoid them</i>
	9:40 am	Jonathan Kew, SIL	<i>X<sub>Y</sub>T<sub>E</sub>X font installation and usage</i>
	10:20 am	<i>break</i>	
	10:30 am	Federico Garcia, Univ. of Pittsburgh	<i>Capabilities of PDF interactivity</i>
	11:10 am	Boris Veytsman, GMU & AES ITT	<i>Automatic report generation with Web, T<sub>E</sub>X and SQL</i>
	11:50 am	Kaveh Bazargan, River Valley Technologies	<i>Removing vertical stretch: Mimicking traditional typesetting with T<sub>E</sub>X</i>
	12:30 pm	<i>lunch</i>	
	1:40 pm	David Walden, E. Sandwich, MA	<i>A lifetime as an amateur compositor</i>
	2:20 pm	Troy Henderson, US Military Academy	<i>Creating high-quality technical graphics with MetaPost</i>
	3 pm	<i>break</i>	
	3:10 pm	Andrew Mertz & William Slough, Eastern Illinois University	<i>Graphics with PGF and TikZ</i>
3:50 pm	Jon Breitenbucher, College of Wooster	<i>L<sup>A</sup>T<sub>E</sub>X at a liberal arts college</i>	
4:30 pm	q & a, TUG meeting		
<b>Tuesday August 1</b>	9 am	Peter Flynn, Silmaril Consultants	<i>Rolling your own document class</i>
	9:40 am	Federico Garcia, Univ. of Pittsburgh	<i>L<sup>A</sup>T<sub>E</sub>X and the different bibliography styles</i>
	10:20 am	<i>break</i>	
	10:30 am	Boris Veytsman, GMU & AES ITT	<i>Drawing medical pedigree trees with T<sub>E</sub>X and PStricks</i>
	11:10 am	Elizabeth Dearborn, Buffalo, NY	<i>T<sub>E</sub>X and medicine</i>
	11:50 am	Bob Neveln & Bob Alps, Widener Univ.	<i>Writing and checking complete proofs in T<sub>E</sub>X</i>
	12:30 pm	<i>lunch</i>	
	1:40 pm	Stephen Moye, AMS	<i>A wayward wayfarer's way to T<sub>E</sub>X</i>
	2:20 pm	Steve Peter, Beech Stave Press	<i>Introduction to memoir</i>
	3 pm	<i>break</i>	
	3:10 pm	panel: Barbara Beeton, Peter Flynn, Mirko Janc, Jonathan Kew	moderator: <i>David Walden</i>
	≈ 4 pm	<i>end</i>	
	7 pm	<i>banquet</i> (at the Clarion hotel, Garden Room)	

# How to Create a T<sub>E</sub>X Journal: A Personal Journey

Barbara Beeton

American Mathematical Society  
201 Charles Street  
Providence, RI 02904 USA  
bnb (at) ams dot org

## Abstract

When TUG was first formed, the Internet wasn't generally available; the logical channel for communication with and among TUG's members was on paper. So *TUGboat* came into being.

As T<sub>E</sub>X has matured, the needs of the community have evolved, but paper is still a logical medium for showcasing a typesetting tool.

This talk will introduce high- and low-lights in the history of *TUGboat*, some reasons for choosing its particular format and mode of presentation, several experiments, and lots of my personal experiences as editor.

Editor: A person employed on a newspaper whose business it is to separate the wheat from the chaff, and to see that the chaff is printed.

Elbert Hubbard

Although this epithet was directed at newspaper editors, we've all read material in print that would have been better off left unpublished. As long-time editor of *TUGboat*, I'm sure I've let some chaff slip through, however much I've tried to keep the wheat content high.

I've mostly enjoyed my tenure as editor. However, without the help of a lot of people along the way, we never would have had such a long and interesting voyage. I'll try to give credit where credit is due along the way.

Let's start at the beginning, and proceed from the outside in.

## How I got involved in this madness

TUG came into existence in February 1980 at a meeting held at Stanford University. About 50 people attended. One of the decisions taken at that meeting was to "organize a newsletter". From the minutes of the first steering committee meeting:<sup>1</sup>

Robert Welland agreed to edit the newsletter. The first newsletter will have a report of the meeting and will be distributed free by the AMS upon inquiry about T<sub>E</sub>X. Subsequent newsletters will be by subscription only.

Bob Welland, a math professor at Northeastern University, had no production facilities—but the AMS

did, and the AMS had just undertaken projects to use T<sub>E</sub>X to prepare its administrative publications and to develop an input system (AMS-T<sub>E</sub>X) that would allow mathematicians (or their secretaries) to prepare manuscripts that could be used directly in the composition of AMS journals. This meant that someone was needed in-house at AMS to prepare files

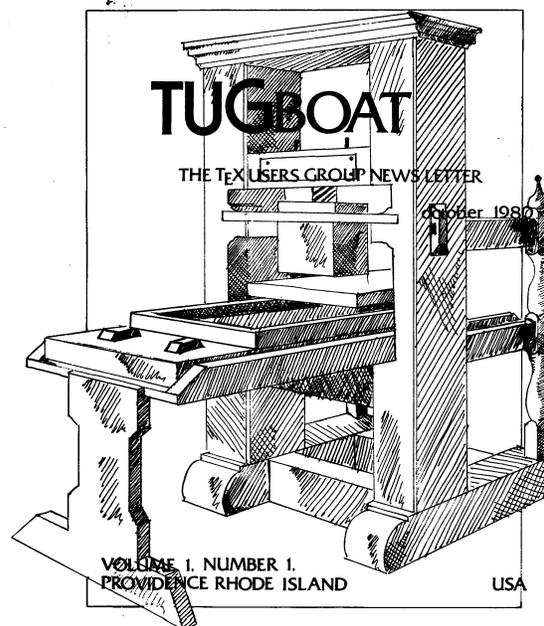


Figure 1: The very first issue—the cover

<sup>1</sup> *TUGboat*, 1:1 (1980), p. 15.

TUGBOAT VOLUME 1, NUMBER 1

Contents  
October 1980

Robert Welland. *Editor's Comments* . . . . . 2

**General Delivery**

Richard Palais. *Message from the Chairman* . . . . . 3

Ellen Swanson. *Publishing & T<sub>E</sub>X* . . . . . 7

Michael Spivak. *A<sub>M</sub>S-T<sub>E</sub>X—"A Very Friendly Product"* . . . . . 10

Robert Morris. *Minutes of the TUG Meeting* . . . . . 12

**Interface Software**

Ignacio Zabala and Luis Trabb-Pardo.  
*The Status of the Pascal Implementation of T<sub>E</sub>X* . . . . . 16

David Fuchs. *The Format of T<sub>E</sub>X's DVI Files* . . . . . 17

Thea Hodge. *University of Minnesota CDC Site Report* . . . . . 19

**Warnings & Limitations**

Barbara Beeton. *Troubles with Trace and Other Oddities* . . . . . 20

**Macros** . . . . . 20

Terry Winograd and Bill Paxton.  
*An Indexing Facility for T<sub>E</sub>X* . . . . . Appendix A

Michael Spivak. *The Joy of T<sub>E</sub>X* . . . . . order form

**Questions & Answers** . . . . . 21

**Letters** . . . . . 21

Gérard Emch and Arnold Pizer . . . . . 22

**Miscellaneous** . . . . . 21

Order Form for *The Joy of T<sub>E</sub>X*

T<sub>E</sub>X Errata

TUG Mailing List

Figure 2: The very first issue — the back cover

The subject of mathematical printing has never been methodically treated, and many details are left to the compositor which should be attended to by the mathematician. Until some mathematician shall turn printer, or some printer mathematician, it is hardly to be hoped that this subject will be properly treated.

Augustus de Morgan  
*Penny Cyclopaedia* (1842),  
on 'Symbols'

# TUGBOAT

THE T<sub>E</sub>X USERS GROUP NEWSLETTER  
EDITOR ROBERT WELLAND

VOLUME 1, NUMBER 1 OCTOBER 1980  
PROVIDENCE RHODE ISLAND U.S.A.

Figure 3: The very first issue — the title page

TUGBOAT  
Volume 4, Number 1 • April 1983

Contents

2 Addresses of Officers, Authors and Others

3 Official Announcements

**General Delivery**

4 Library Subscriptions—What Are They?

4 MICHAEL SPIVAK. Users' Course in A<sub>M</sub>S-T<sub>E</sub>X

4 SAMUEL B. WHEEDEN. TUG Financial Reports

4 TUG Treasurer's Report

5 1983 TUG Budget

**Software**

6 DAVID FUCHS. T<sub>E</sub>Xchax Summary

**Output Devices**

10 CHART. Output Devices and Computers

10 Index to Sample Output from Various Devices

11 NELSON H. F. BEEBE. Low-Cost Downloadable Font Devices

12 PATRICK ION, BILL HALL, HILLA J. THEDFORD. T<sub>E</sub>X on the OSP130

13 Sample Output from Florida Data OSP 130

**Site Reports**

14 DAVID FUCHS. News from All Over

- HP 1000 16 IRENE J. BUNNER AND JOHN D. JOHNSON. T<sub>E</sub>X on the HP-1000

17 RICHARD FURUTA and PIERRE MACKAY. Unix T<sub>E</sub>X Site Report

18 PAVEL CURETS and HOWARD TRUCEY. Forging T<sub>E</sub>X to VAX/UNIX

21 PIERRE MACKAY and RICHARD FURUTA. T<sub>E</sub>X at the University of Washington: Topo-20, Unix, Versatec, and the Monolithic

- VAX/VMS 22 MONTE C. NICHOLS. VAX/VMS Site Report

23 DAVID KRAPP. T<sub>E</sub>X at Calma R&D

- DG MVS/800 23 NORMAN NAUGLE and BART CHILDS. T<sub>E</sub>X at Texas A&M University

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

24 LANCE CARNER. Editor's Introduction

**Fonts**

25 SCOTT GUTHERY. Pictures Are Just Big Letters

26 GEORGIA K. M. TOBIN. Computer Calligraphy

33 Announcement: Fifth ATypI Working Seminar,  
*The Computer and the Hand in Type Design: The Aesthetics and Technology of Digital Letterforms*

**Macros**

33 TUGboat Macro Index

35 BARBARA BEETON. How to Build a *natex*

36 BARBARA BEETON. Determining Habitable Size and Other Quantities

37 AUGUST MOHR. Some Layout Macros

38 ROBERT M. MCCLURE. Testing the Widths of a Font

**Problems**

39 Hanging punctuation

**Letters et alia**

40 MARIA CODE. How to obtain T<sub>E</sub>X82 on tape

40 Announcement: Manipulation de Documents—*Journées Francophones*

**Dreamboat**

41 A. E. SIEGMAN. T<sub>E</sub>X As a Programming Language?

**Advertisements**

43 Textset, Inc.—A Service for T<sub>E</sub>X Users

44 Quality Micro Systems—Lasergrafix 1200

**Miscellaneous**

45 T<sub>E</sub>X82 Order Form

47 Membership Application and Order Form

T<sub>E</sub>X and METAFONT: Errata and Changes — Supplement

TUG Membership List — Supplement

Figure 4: TUGboat 4:1 — the back cover

TUGBOAT  
THE T<sub>E</sub>X USERS GROUP NEWSLETTER

PROVIDENCE • RHODE ISLAND • U.S.A.

Figure 5: TUGboat 4:1 — the cover

The perpetrator of this assignment was Sam Whidden, head of the AMS Information Systems Development department, and the founding treasurer of T<sub>E</sub>X Users Group. He was also my boss. I didn't have a chance.

Sam was also responsible for the name *TUGboat*—the vessel that would convey the organization, TUG, through twisty little passages.

### The covers and title page

The first issue of *TUGboat* appeared in October 1980. Bob Welland found an old press that was allegedly a reproduction of the one used by Gutenberg. He made a pen-and-ink drawing that has graced the cover ever since, going through various adaptations:

- For the first issue, a photograph was made of the background, and an overlay was prepared on clear film using rub-on type for the text (Fig. 1). The contents list was placed on cover 4 (Fig. 2), a practice that has continued with only one exception.
- The title page used the same pasted-up “*TUGboat*”, but everything else was set with T<sub>E</sub>X, including an epigraph (Fig. 3), a practice modeled on use of quotes in *The T<sub>E</sub>Xbook*. Finding suitable quotes has provided me considerable amusement, as well as occasional panic attacks when a deadline was approaching, and nothing had turned up. (I cheerfully accept suggestions for quotes, and must thank Don Knuth in particular for his many contributions.) I believe I've received more comments about the epigraphs than about almost anything else; I'm not sure what this is supposed to imply, but it does show that people at least open the cover and look at the title page.
- In the summer of 1982, I attended a workshop at RISD (the Rhode Island School of Design) on the topic “Design with type”. For one of my projects, I decided to redesign the table of contents—I really don't like the dotted effect. I had two goals (in addition to improving the appearance): to strengthen the association between page number and what appears on the page, and to subdivide the contents into logical subject areas. The new cover 4 design debuted with the first issue of 1983 (Fig. 4). This issue was also the first to have all the cover text (except for the name *TUGboat*) prepared in T<sub>E</sub>X (Fig. 5), with a “pseudo-spine”—rotated text identifying the issue running from top to bottom near the stapled edge. (Later, when issues were large enough to have a real spine, this text was moved there.)

- Bob Welland “retired” from the editor's post as of the end of the 1983 academic year, and, with no obvious candidates clamoring to take over, I became editor with issue 4:2. (I had been doing most of the production work, after all.) I celebrated this occasion by omitting the name of the publication from the title page (Fig. 6). Sigh.

Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?  
 Brian W. Kernighan and P. J. Plauger  
*The Elements of Programming Style*,  
 Second edition, McGraw-Hill, 1978.

THE T<sub>E</sub>X USERS GROUP NEWSLETTER  
 EDITOR BARBARA BEETON

VOLUME 4, NUMBER 2 • SEPTEMBER 1983  
 PROVIDENCE • RHODE ISLAND • U.S.A.

Figure 6: *TUGboat* 4:2—the title page

- By 1984, sentiment had been expressed that *TUGboat* should be a representative example of high quality T<sub>E</sub>X composition. Dave Kellerman and Barry Smith volunteered to guest-edit and produce an issue demonstrating this capability. They commissioned a designer and a special cover drawing for this issue, which appeared as the first issue of 1986 (Fig. 7). Along with the change in format, the subtitle was upgraded from “The T<sub>E</sub>X Users Group Newsletter” to “The Communications of the T<sub>E</sub>X Users Group”. The content of the issue was set to a grid, which may be apparent in the layout of the title page (Fig. 8). To avoid the appearance of clutter, the contents list was omitted from cover 4. (Although I understand and sympathize with the goal, I've found the lack of a T-of-C inconvenient, and have taped one to the

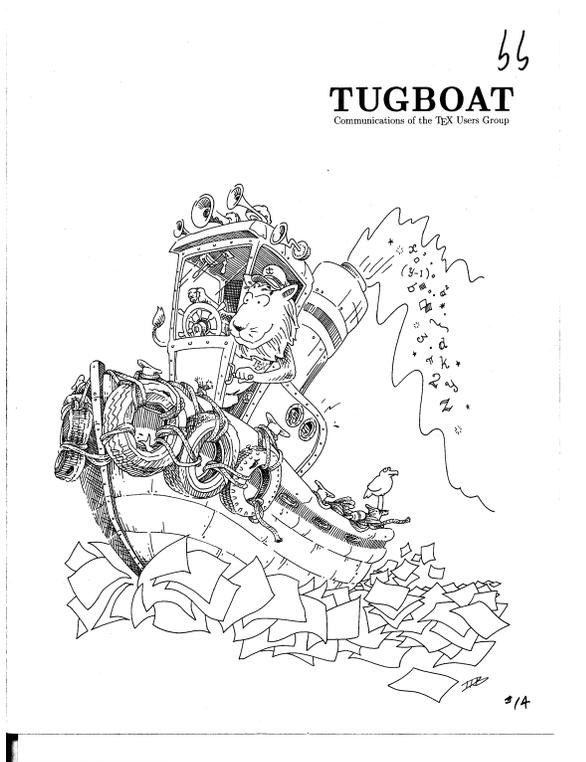


Figure 7: *TUGboat* 7:1 — the cover of the guest-edited issue

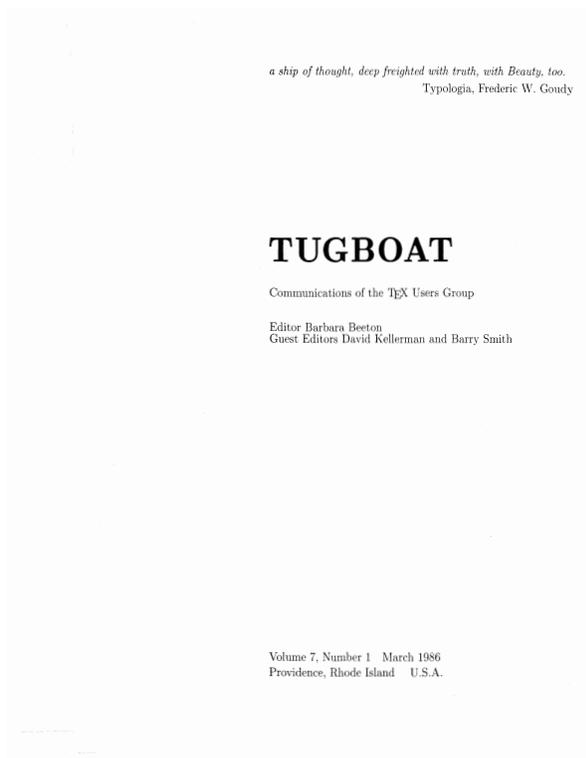


Figure 8: *TUGboat* 7:1 — the title page

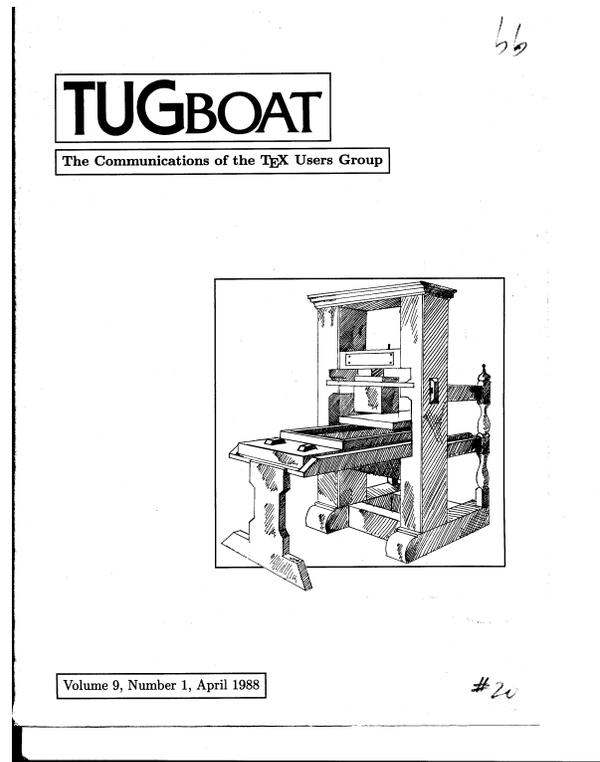


Figure 9: *TUGboat* 9:1 — a new look for the cover

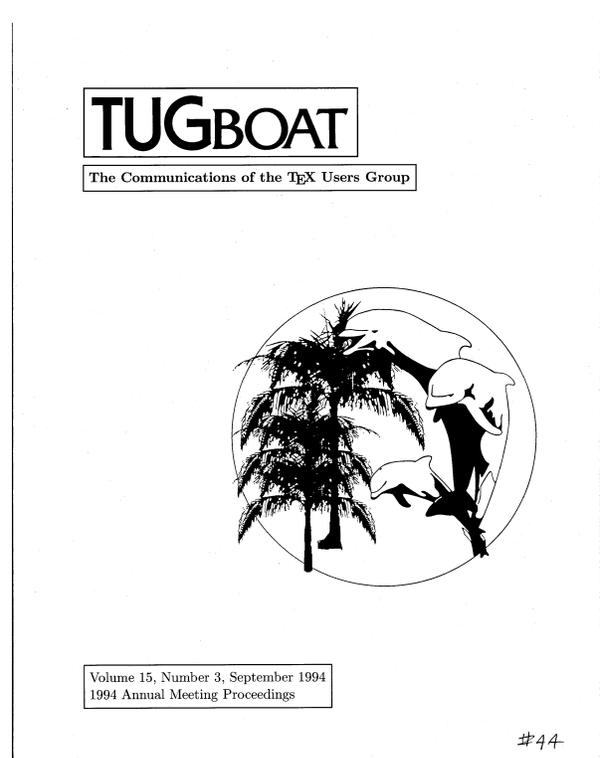


Figure 10: *TUGboat* 15:3 — and a new look for proceedings issues

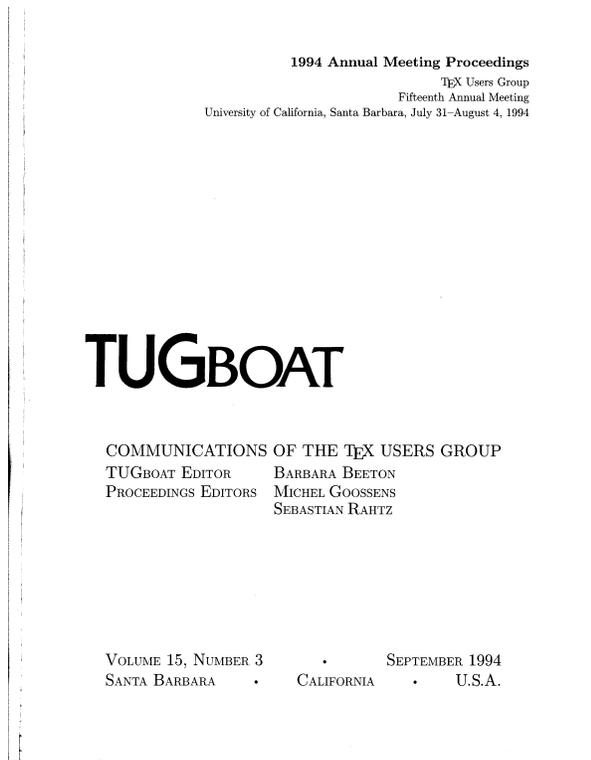


Figure 11: *TUGboat* 15:3—a proceedings issue title page

issue for ease of reference.) More about the content of this issue later.

- The covers and contents reverted to the previous layout with the next issue, and nothing much changed until the first issue of 1988, when Alan Wittbecker, an employee at the newly-relocated TUG office, hired to assist with *TUGboat* production (among other things), reformatted the front cover (Fig. 9), reducing the size of the press drawing and boxing all the other cover elements. Note, however, that the cover drawing and the *TUGboat* name were still pasted up manually for each issue.
- With the first issue of 1989, *TUGboat* permanently got a real spine! No more guessing which one to pull out from a growing run of anonymous grayish covers.
- It gradually became a tradition for annual meetings to have a drawing representing the meeting location. Beginning with the proceedings of the 1994 meeting in Santa Barbara, California, this drawing replaced the press on the cover (Fig. 10). The title page of a proceedings issue is also modified (Fig. 11), substituting the location of the meeting for the epigraph, and

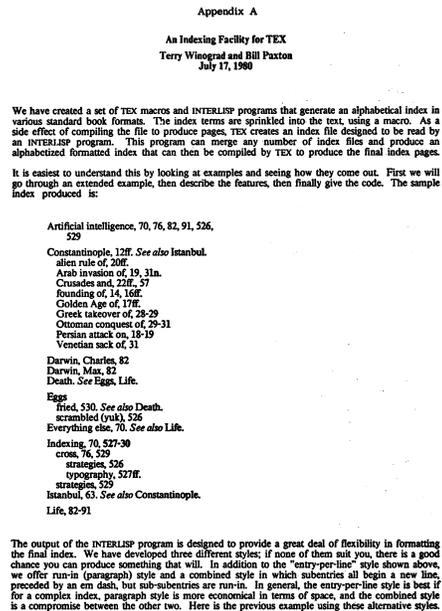


Figure 12: *TUGboat* 1:1, an item reproduced directly from author copy

identifying the proceedings editors who are responsible for the production while I get to rest. This practice continues to the present day.

### General format and layout

*TUGboat* is formatted for US letter-size paper, 8.5 × 11", although it is sometimes trimmed a bit smaller. (The guest-edited issue and several that followed were 8 × 10.5".) This was established at the first issue.

There were several reasons for this decision. First, in the US, authors are used to preparing manuscripts on the paper that is easiest to obtain, and that's letter size. We were hoping to encourage authors to prepare submissions that would be ready to use, and indeed, the first issue contains some items reproduced directly from author-submitted copy (Fig. 12).

The capacity of the press was also a consideration, as was the size of shelves and files. A final product formatted to letter size is readily accommodated by the presses in the AMS print shop; printing is actually done on larger sheets that are then folded and gathered. Anything smaller must be trimmed, which can result in considerable waste.

The material that we expected to publish initially included reports on T<sub>E</sub>X development, news

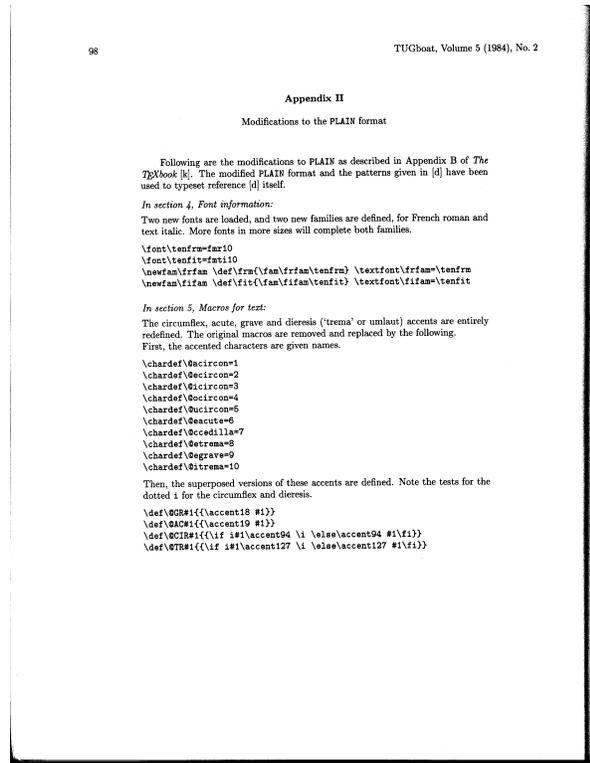


Figure 13: TUGboat 5:2, a single-column page

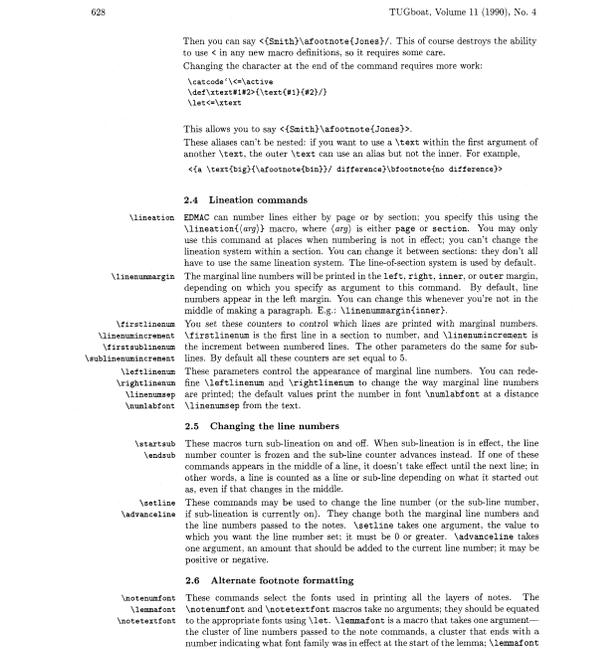


Figure 14: TUGboat 11:4, a doc style page

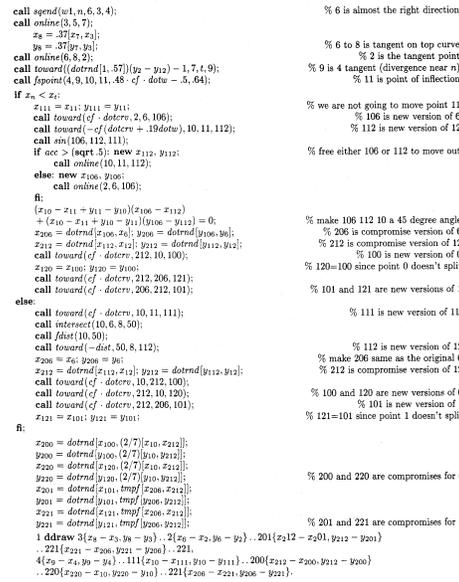


Figure 15: TUGboat 5:2, full-width code in the style of The METAFONTbook



Figure 16: TUGboat 16:2—code takes up space

## B Catalogue of Packages

Table 2: T<sub>E</sub>X Live packages

Package	Collection	Description
2up	generic3	Allows one to print a document two-up, with considerable flexibility as to paper size and layout. It produces a standard dvi file, and does not involve an additional dvi or PostScript filter. It should work with most T <sub>E</sub> X macro packages.
alposter	latex3	A L <sup>A</sup> T <sub>E</sub> X class providing fonts in sizes of 12pt up to 107pt. It also makes sure that in math formulae the symbols appear in the right size. This package also creates a PostScript header file for dvips which assures that the poster will be printed in the right size. Sizes DIN A0, DIN A1, DIN A2 and DIN A3 are also supported.
a4	latex3	Originally for L <sup>A</sup> T <sub>E</sub> X 2.09 but updated for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> . Mostly superseded by native L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> support for A4 paper but define the extra option of widemargins. The geometry package is usually what you are looking for though.
aaai	latex3	AAAI style.
accents	latex3	A package for multiple accents with nice features concerning creation of accents and placement of scripts.
achemos	latex3	L <sup>A</sup> T <sub>E</sub> X and B <sup>H</sup> L <sup>A</sup> T <sub>E</sub> X style for American Chemical Society.
acronym	latex3	This package ensures that all acronyms used in the text are spelled out in full at least once. It also provides an environment to build a list of acronyms.
adeflists	latex3	A class that satisfies the requirements of the Australian Defence Force Academy (a college of the University of New South Wales).
adriat	latex3	Using address lists in L <sup>A</sup> T <sub>E</sub> X.
ae	font3	A set of virtual fonts which simulate T1 coded fonts using the standard CM fonts. The package is called AE fonts (for Almost European). The main use of the package is to produce PDF files using Type1 versions of the CM fonts instead of the bitmapped EC fonts.
aguplus	latex3	Styles for American Geophysical Union.
aias	latex3	A bundle of L <sup>A</sup> T <sub>E</sub> X/B <sup>H</sup> L <sup>A</sup> T <sub>E</sub> X files and sample documents to aid those producing papers and journal articles according to the guidelines of the American Institute of Aeronautics and Astronautics (AIAA).
alates	format3	An extended L <sup>A</sup> T <sub>E</sub> X with better modularity.
alg	latex3	L <sup>A</sup> T <sub>E</sub> X environment for typesetting algorithms.
algorithmx	latex3	Defines a floating algorithm environment designed to work with the algorithmic package.
alpha-ocf2	systems1	System binaries for Alpha running OSF 3.2.
alpha-ocf4	systems1	System binaries for Alpha running OSF 4.0.
allfont	latex3	A generalised replacement for some parts of gensf and mfnfs. Similar to pdfont with the PostScript specific extension.
amiga	systems1	An Amiga port of the complete UNIX-T <sub>E</sub> X system.
amofonts	am2	A set of miscellaneous T <sub>E</sub> X fonts from the American Mathematical Society that augment the standard set normally distributed with T <sub>E</sub> X. The set includes: Extra mathematical symbols; Blackboard bold letters (uppercase only); Fraktur letters; Subscript sizes of bold math italic and bold Greek letters; Subscript sizes of large symbols such as sum and product; Added sizes of the Computer Modern small caps font; Cyrillic fonts (from the University of Washington); Baler math fonts.
amstex	am2	A collection of loosely related files that are distributed together by the American Mathematical Society. These files are miscellaneous enhancements to L <sup>A</sup> T <sub>E</sub> X whose aim is superior information structure of mathematical documents and superior printed output.
answers	latex3	American Mathematical Society plain T <sub>E</sub> X macros.
	latex3	Styles for setting questions (or exercises) and answers.

Figure 17: TUGboat 19:1 — we tried to format the T<sub>E</sub>X Live contents listing to be readable

alposter	Provides fonts in sizes of 12pt up to 107pt. Provides fonts in sizes of 12pt up to 107pt and also makes sure that in math formulae the symbols appear in the right size. Can also create a PostScript header file for dvips which ensures that the poster will be printed in the right size. Supported sizes are DIN A0, DIN A1, DIN A2 and DIN A3.
a2ac	Author: unknown; CTAN location: macros/latex/contrib/supported/alposter AFM to AFM plus Composites. Enables the use of PostScript fonts while typesetting texts in a given language; the presence of more accents themselves (no whole accented characters) is sufficient. The configuration files of the a2ac program are independent of the PostScript font encoding and of the typesetting system encoding. The program may be used to prepare a font for any typesetting system, especially T <sub>E</sub> X.
a4	Author: Petr Ošáň; CTAN location: fonts/utilities/a2ac Support for A4 paper sizes. Provides support for A4 paper sizes, however it is mostly superseded by the adpaper option of L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> and by the geometry package. It does, however also define the extra option of widemargins.
atidclass	Author: Nico Poppelier and Johannes Braune; CTAN location: macros/latex/contrib/supported/atidclass Increases width of printed area of an A4 page. This package provides an option to increase the width of the A4 page. Note however that it is superseded by geometry.
a5	Author: unknown; CTAN location: macros/latex/contrib/other/a5ac Support for A5 paper size. This package provides support for A5 paper sizes. Note however that it is superseded by geometry.
a5com	Author: Mario Wolzko; CTAN location: macros/latex/contrib/other/a5ac Support for A5 paper sizes. Superseded by geometry.
aaai	Author: unknown; CTAN location: macros/latex209/contrib/aaai American Astronomical Society format.
aaates	Author: American Astronomical Society; CTAN location: macros/latex/contrib/supported/aaates Text abbreviations in L <sup>A</sup> T <sub>E</sub> X. A L <sup>A</sup> T <sub>E</sub> X package defining abbreviation macros, which expand to defined text and insert following spaces intelligently, based on context. They can also expand to use thing the first time they are used and another thing on subsequent invocations. Thus they can be abbreviations in two senses, in the source and in the document. Also includes a facility for suffixes like B <sup>H</sup> 000C and G <sup>H</sup> 000A which correctly handles following periods.
abctext	Author: Matt Swift; CTAN location: macros/latex/contrib/supported/frankenstein Notate tunes stored in abc notation. A package to notate tunes stored in an ASCII format (abc notation). One of the most important aims of abc notation, and perhaps one that distinguishes it from most, if not all, computer-readable musical languages is that it can be easily read by humans. The package produces files that can be processed with MusicT <sub>E</sub> X.
abctex	Author: Chris Wolzko; CTAN location: support/abctex Books of abstracts. A L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> class file for making “books of abstracts”, commonly used for conferences. It is based on report class, however \chapter has been redefined and shouldn’t be used.
abstract	Author: Haveli Dima; CTAN location: macros/latex/contrib/other/abstract Control the typesetting of the abstract environment. The abstract package give you control over the typesetting of the abstract environment, and in particular provides for a one column abstract in a two column paper.
abstex	Author: Peter R. Wilson; CTAN location: macros/latex/contrib/supported/abstract No description available.
abstex	Author: unknown Multiple accents. A package for multiple accents with nice features concerning creation of accents and placement of scripts.
abzoo	Author: Javier Bezos; CTAN location: macros/latex/contrib/supported/abzoo Includes makfont, vpl2pt, CSX, Xef, and Normam.def.
acfonts	Author: John Smith; CTAN location: fonts/utilities/acfonts L <sup>A</sup> T <sub>E</sub> X and B <sup>H</sup> L <sup>A</sup> T <sub>E</sub> X style for American Chemical Society.
achemos	Author: Mats Dahlgren; CTAN location: macros/latex/contrib/supported/achemos

Figure 18: TUGboat 21:1, but finally just packed it in as tightly as we could

How to Create a T<sub>E</sub>X Journal: A Personal Journey

about the Users Group and about what users were doing, “sales pitches” (why T<sub>E</sub>X is a Good Thing), examples of things that can be done with T<sub>E</sub>X, and solutions to problems.

While blocks of small type on large pages is not easy to read (most AMS books and journals have a text width of 30pc, about 5in), the letter-size page is wide enough to hold two columns of type that are narrow enough to be read easily, but (almost) wide enough to avoid most formatting problems. So a two-column style was adopted as the basic layout.

## Variations on the theme

But some material simply can’t be shoe-horned into two columns. We’ve already seen one example printed directly from an author submission (Fig. 12).

- Macros are often difficult to disassemble into the narrow measure, so a “medium-width” format was defined, with a 30pc measure, centered horizontally on the page (Fig. 13). This would be used only sparingly, when the density of macro code makes it impossible to reformat to the two-column style.
- Another single-column format, with text narrower than full page width, that is used occasionally is the L<sup>A</sup>T<sub>E</sub>X doc format, where macro code is interspersed with commentary, and a wide left margin is used to place macro names as labels (Fig. 14).
- Other material sometimes calls for use of the full page width, as with the presentation of code emulating, for example, *The METAFONTbook* (Fig. 15).
- Extended code listings have been poured into a full-width page container (Fig. 16) for lack of any better ideas.
- And several iterations of all or part of the Catalogue accompanying a T<sub>E</sub>X Live disk have used a specially formatted full-width presentation to pack as much information as possible onto the page with (Fig. 17) or without (Fig. 18) rules or shading to help guide the eye.

The initial macros to implement these layouts were based on a plain-T<sub>E</sub>X multi-column macro system developed for in-house use at AMS. The original requirements for this system included some interesting features:

- the ability to have as many columns as the data would allow (we’ve used up to 12);
- full-width “banners” can float across the page at top or bottom or anywhere in between;
- partial width insertions can float across just some columns;





the formula  $2x+15$ , the second  $x$  is changed from  $\text{bin\_road}$  to  $\text{ord\_road}$  in 5728. It turns out that thick spaces are inserted after the  $x$  and before the 1 in 'x = 1'; medium spaces are inserted before each  $x$  and '+'; thin spaces are inserted after each comma in 'x, 1'.

31. The behavior of the simpler algorithm, which we may call Brand X, can be deduced from the demerits values (d\*) in the trace output. There is only one reasonable choice, 021, for the first line:

31. When your instructor made up this problem, he said '\tracingparagrpahs=1' so that his transcript file would explain why TeX has broken the paragraph into lines in a particular way. He also said '\pretolerance=1' so that hyphenation would be tried immediately. The output is shown on the next page: use it to determine what line breaks would have been found by a simpler algorithm that breaks one line at a time. (The simpler algorithm finds the breakpoint that yields fewest demerits on the first line, then chooses it and starts over again.)

32. (This exercise takes awhile, but the data structures are especially interesting: the hyphenation algorithm is a nice little part of the program that can be studied in isolation.) The following tables are constructed:

Table with 3 columns: op, char, link. Rows include trs[96] through trs[105], hyf\_distance, hyf\_noun, and hyf\_next.

and there's only one, 022, for the second. But for the third, a line from 022 to 043 (the break after 'para:') has 16725 demerits, which certainly looks worse than the 1225 demerits from 022 to 024. This, however, leads Brand X into a trap, since there's no good way to continue from 024. Similarly, Brand X will choose to go from 027 to 022, and this forces it to 0211 and then infelicitously to 0213 (because the syllable 'break-' is too long to be squeezed in). The resulting paragraph, as typeset by Brand X, looks like this (awful):

Given the word abcdef, it is interesting to watch 1923 produce the hyphenation numbers '0a02b1c0d5' from this trie.

33. The idea is to keep line numbers on the save stack. Scott Douglass has observed that, although TeX is careful to keep cur\_boundary up to date, nothing important is ever done with it; hence the save\_order field in knots/boundary words is not needed, and we have an extra halfword to play with! (The present data structure has fossilized elements left over from old incarnations of TeX.) However, line numbers might get larger than a halfword; it seems better to store them as fullword integers.

This problem requires changes to three parts of the program. First, we can extract 11063 as follows:

```
(Cases of main_control that build boxes and lists 1006) =#
new_node(left_brace, begin_saved(0) := line: incr(save_ptr); new_save_level(simple_group);
end; { the line number is saved for possible use in warning message }
any_mode(begin_group): begin saved(0) := line: incr(save_ptr); new_save_level(semi_simple_group);
end;
any_mode(end_group): if cur_group = semi_simple_group then
begin unsave; decr(save_ptr); { pop unused line number from stack }
end
else off_save;
```

Figure 25: TUGboat 11:4—the flow of this page doesn't follow the usual path

Software

Editor's Introduction
Helmut Jürgensen

The Software Column of this TUGboat issue is a call for help—your contributions or suggestions!

Some of the things we would like to receive are outlined below. If you consider submitting material which you feel would fit into the Software Column but which is not covered by the list, please submit it anyway—or contact me.

TeX has come of age now—everybody said so at the TUG meeting. The main interest of TUG members seems to shift away from "mere" implementation questions gradually and towards "real" typesetting problems. Nevertheless, it also seems that there are still quite a few difficulties arising when embedding the TeX and METAFONT systems into different environments in a user-friendly fashion. A lot of software has been and is being written to handle the problems. In fact, judging from some discussions I had at the TUG '84 meeting, it is written again and again and again. Therefore, I think the TUGboat and this column could help in distributing information and discussing ideas.

Please send descriptions of TeX and METAFONT-related software. They could be short, giving only name, purpose, availability, and—enough—important technical data. They could be long as well, if of sufficiently general interest. In some cases novel software methods may require a more thorough treatment. WEB-related problems would form another area of interest.

In general, information about software used in the implementation of the TeX, METAFONT, and WEB systems, in the adaptation to your environment or your research work, about enhancements to these systems, or software engineering comments related to these systems are welcome.

I should also appreciate receiving suggestions about the column in general, comments on articles, hints about related publications elsewhere, etc.

Please send your contributions and ideas to Helmut Jürgensen
Department of Computer Science
The University of Western Ontario
London, Ontario
Canada N6A 5B7

You can also reach me by phone (519-679-3039) or via memex (deepthroat@uwo-hobbit.jurgensen). Please remember, I need your contributions—otherwise the column will starve—and I must have them no later than two weeks before the general TUGboat deadline.

TeXtensions

How to Run TeX in a French Environment: Hyphenation, Fonts, Typography

Jacques Désarménin

1. Introduction

One of the greatest achievements of text-processing programs—especially TeX—is that they allow anybody to produce beautiful, typeset-like documents by merely typing a correctly prepared manuscript on a terminal, using a keyboard very similar to that of an ordinary typewriter.

As a consequence, before trying to train professional typists in TeX, one must ensure that a replica of the keyboard they are used to will appear on the terminal. In our situation, this means that some extra characters—é, è, ç among others—have to be included where they usually appear on a French keyboard. Some others must be displaced; for example, a and q have to be permuted and the digits retain their positions but must be typed by using the SHIFT key. The solution to these specific problems, and a certain number of improvements to the input of TeX, have been given by D. Foata and Y. Roy in Strasbourg [1].

The main concern in processing French text is hyphenation. The set of patterns that TeX uses to hyphenate English works very poorly in French. The procedures are fundamentally different, not only because the etymology has much more importance in English, but also because the syllabifications do not obey the same rules in both languages. The same word might not be hyphenated the same way in French and in English: ma-gni-fi-cence vs. mag-nif-i-cence. We explain in another article

©Copyright 1984 by Jacques Désarménin

General Delivery

MESSAGE FROM THE PRESIDENT

Pierre MacKay

There is much to celebrate. TeX "came of age" in December, and the TeXbook is out. Which leads to the first and most urgent of several messages, addressed to those members of TUG who may still be using PTEX, TEX80 or one of their offspring. The message is, "STOP!" TEX80 is now an interesting piece of technological archaeology; it was a necessary step along the way, but it does not offer anything like the power of genuine TeX, and it can never improve. Change now, and you will face a slightly painful but rather brief effort of macro conversion, rather like getting a tooth fixed. Be brave. Get it over with. You will be glad you did.

This leads to a second message. In the next year or so, we are likely to see the appearance of TeX offspring. Some will be the enhancements that have deliberately been allowed for in the final modules of the TeX WEB file, and some may be more like HalfTeX, PartsTeX, RathaTeX and HarlotTeX. There may be good arguments for some of them, but there should be no arguments about the legitimacy of genuine versions of TeX. The test of the real thing lies in a file called TRIP-TEX. When you run TRIP through your newly compiled TeX, you get a very alarming set of diagnostic messages on the log file, and if that log file agrees with a master copy, then what you have is pretty sure to be TeX. If the log files don't agree, watch out.

Similarly, all DVI interpreters ought to produce the same basic results at the same output resolution. The DVITYPE program which is included in any distribution of TeX sets the standard. If anything seems odd, it should be possible to try out various sizes of rule and space and see whether the results produced by DVITYPE match the results you see on your laser-printer. The operation of high-resolution typesetters is rather less easy to check, but here it ought to be possible to check against the ideal measurements in the TeX source file. When you ask for a one-pica rule from a phototypesetter, you ought to get exactly that. When you try to do that on a low resolution printer, you can expect some fairly gross adjustments owing to rounding.

There is a fair amount of work still to be done with fonts, and perhaps the most significant news at this time is that METAFONT is being converted

from a SAIL program running in a very limited environment to a Pascal program which ought to run wherever TeX can run. For the present, we are still dependent on the old METAFONT and some very interesting problems have surfaced from the need to generate low-resolution fonts in several closely related pixel densities. The general lesson is that in font design you have to take a great many factors into account, not the least of which is the interaction of ink and paper on the specific device you are designing for. It will never be enough just to take a high-resolution design and cut it down mechanically to low-resolution densities.

At the end of February, there was a lively exchange over the mail networks about the possibility of establishing some standards for the inclusion of graphics in TeX. Several sites have already worked out protocols using the TeX special, and there was a widespread feeling that before we go too far in separate directions it would be a good idea to arrange for a department in TUGboat where the systematic use of special sequences could be worked out. We can probably wait to set this up until August, but meanwhile, if you are doing anything interesting with graphics and TeX, send it on through one of the networks, or to Barbara Beeton at the AMS, or to me. And remember that we will need a volunteer to coordinate the new department, so, if you are feeling public-spirited, we need you.

Unfortunately, we are losing one of the most active and public-spirited members of TUG for now, though we hope very much that it is only for a short time. Lynne Price, who has been one of our chief guides through the early years of TUG, has written that she is moving to a place where, for the moment, she has no access to TeX. The whole organization will miss her, and the Steering Committee will miss her especially. We will need someone else to take up her position as macro coordinator for TUGboat. Once again, do we have a volunteer?

To Lynne, for the present, we say "The very best of luck, and hurry back."

TeX INCUNABULA

by Donald E. Knuth

Several people have asked me for a list of the "first" books ever typeset by TeX. Bibliophiles might some day enjoy tracing the early history of this particular method of book production; I have therefore tried to record the publications known to me, before my memory of those exciting moments fades

LaTeX

The trace package\*

Frank Mittelbach

Introduction

When writing new macros one often finds that they do not work as expected (at least I do :-). This happens and one can't immediately figure out why there is a problem one has to start doing some serious debugging. TeX offers a lot of bells and whistles to control what is being traced but often enough I find myself applying the crude command \tracingall which essentially means "give me whatever tracing information is available".

In fact I normally use e-TeX in such a case, since that TeX extension offers me a number of additional tracing possibilities which I find extremely helpful. The most important ones are \tracingassign, which will show you changes to register values and changes to control sequences when they happen, and \tracinggroup, which will tell you what groups are entered or left (very useful if your grouping got out of sync). So what I really write is

```
\tracingassign=\tracinggroups+\tracingall
```

That in itself is already a misname (since it is a mouthful) but there is a worse catch: when using \tracingall you do get a awful lot of information and some of it is really useless.

For example, if LaTeX has to load a new font it enters some internal routines of NFSS which scan font definition tables etc. And 99.9% of the time you are not at all interested in that part of the processing but in the two lines before and the five lines after. However, you have to scan through a few hundred lines of output to find the lines you need.

Another example is the calc package. A simple statement like \setlength

```
\linewidth{1cm} inside your macro will result in
\setlength <-> \protect \setlength
<void>
\setlength <-> \calcassign@skip
\calcassign@skip <-> \calc@assign@pg@numeric \calc@skip \calc@skip
\calc@assign@numeric #123456->list \calc@list #11c \calc@list #21parameter \calc
Open \setlength@font #4{global \calc@list \calc@list \endgroup #3\calc@list
#1c-\calc@skip
#2c-\calc@skip
#3c-\linewidth
#4c-list
(list)
(list)
\endparameter)
\endparameter)
\calc@open <-> \begin@group \aftergroup \calc@init@list \begin@group \aftergroup \calc
\end@list \calc@open@font
\begin@group)
\aftergroup)
\begin@group)
\end@group)
\end@group)
```

\*This file has version number 1.0 in trace LaTeX code, last revised 2000/02/16.

Figure 27: TUGboat 5:2, with a new font for the section headings

Figure 26: TUGboat 5:1 — starry section headings

Figure 28: TUGboat 22:1 — the subject heading covers only the text of a doc-style article

Subject arrangement

Within the basic format, the content was divided into logical subject areas (General Delivery, etc.), introduced by distinctive headings. I no longer remember whose ideas were used in creating the style of these headings, but, like the covers, they have undergone some major changes through the years.

- For the first few years, the subject heads were set in a rather spindly sans-serif, centered, with rows of asterisks strung out above and below (Fig. 26). Similar rows of asterisks were used to separate articles within subject areas.
- Beginning with issue 5:2, a much nicer demi-bold sans was adopted, with the subject text centered in a column-wide box (Fig. 27).
- For items like the calendar that fill a dedicated page (Fig. 21), the subject head can span the full page, and if an article using the L<sup>A</sup>T<sub>E</sub>X doc style starts a section, the box is set to the width of the text (Fig. 28).
- With the change to boxed subject heads, articles in a subject area were separated only by vertical space. Owing to confusion in identifying the end of one article and the beginning of another, in 1989 a rule was added above the title of each succeeding article (Fig. 29).
- Subject areas managed by an associate editor sometimes have a more distinctive subject head. This has been particularly true for early installments of the Font Forum (Fig. 30) and for the Treasure Chest since 1998 (Fig. 31).
- For the guest-edited issue, the arrangement was entirely different. Short items were run together on pages of three columns, and articles of a page or more each began on a new page. This is best appreciated in context: Go to the *TUGboat* web site to examine this issue.

While we're on the subject of subject headings, we originally tried to follow the nautical theme implied by the name *TUGboat*, hence "General Delivery" and "Dreamboat" (wishes for the future), but we clearly ran out of inspiration. So the names of most subjects are far more prosaic.

The transition from plain T<sub>E</sub>X to L<sup>A</sup>T<sub>E</sub>X

The first issues of *TUGboat* — the parts that were actually prepared using the *TUGboat* style — were constructed in T<sub>E</sub>X 78. This language was rather different from the T<sub>E</sub>X we know now: only 32 fonts could be used at once, the syntax for boxes and alignments was different, etc., etc. In other words, a file created for T<sub>E</sub>X 78 probably won't run under

**General Delivery**

**Donald E. Knuth Scholarship**

Larry Sharlow was honored at the 1988 Annual Meeting, McGill University, Montréal, as the 1988 Scholarship Winner. He has volunteered to serve on the 1989 selection committee.

We are pleased to announce the Fourth Annual "Donald E. Knuth Scholarship" competition. This year two Scholarships will be awarded. The awards consist of an all-expense-paid trip to TUG's 1989 Annual Meeting and the Short Course offered immediately following the meeting. The competition is open to all 1989 TUG members holding support positions that are secretarial, clerical or editorial in nature.

To enter the competition, applicants should submit to the Scholarship Committee by May 12, 1989, the input file and final T<sub>E</sub>X output of a project that displays originality, knowledge of T<sub>E</sub>X, and good L<sup>A</sup>T<sub>E</sub>X style. The project may make use of a macro package, either a public one such as L<sup>A</sup>T<sub>E</sub>X or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge. Along with the T<sub>E</sub>X files, each applicant should submit a letter stating his/her job title, with a brief description of duties and responsibilities, and affirming that he/she will be able to attend the Annual Meeting and Short Course at Stanford University, Stanford, California, August 21-25, 1989.

Selection of the scholarship recipient will be based on the T<sub>E</sub>X sample. Judging will take place May 13 - June 12, and the winner will be notified by mail after June 12.

All applications should be submitted to the Scholarship Committee at the following address:

Larry Sharlow  
10 Tottee #3  
Flagstaff, AZ 86001

**From the President**  
Bart Childs

I am looking forward to the celebration of 10 years of TUG at Stanford. The call for papers has already appeared. I hope you can and will participate.

Below is an announcement for a "dingbat" competition. I have had a lot of positive response from the preliminary versions. I hope the competition will be a good step in expanding our METAFONT horizons, which I feel have been neglected.

---

**Announcing  
A TUG Dingbat Competition**

TUG announces a METAFONT competition for the creation of the best dingbat characters. Each entry will consist of one specific character (such as a logo) or a dingbat family, the METAFONT source, annotation of the source for pedagogical use, and samples of the use of the character(s). These characters can be in the spirit of the Zapf dingbats in PostScript, symbols, icons, logos, or of some other useful or entertaining nature.

A dingbat family could be:

- a character in different orientations (such as a hand or flag).
- a character in different presentations (such as outline, solid, black on white, white on black, gray, ...).
- a set of characters for doing border designs (TUGboat, Vol. 5, no. 2).
- a set of characters for a particular use (TUG could use an anchor, a dinghy, a printing press, ...).
- any reasonably useful, entertaining, or interesting character.

TUG is holding this competition to encourage the use of complete T<sub>E</sub>X systems and to complement the initial system that was created by Don Knuth and given to us all. It is hoped that this competition will contribute to excellence in fonts, graphics, and documents in general.

Figure 29: *TUGboat* 10:1 — now we make the top of every article stand out



Figure 30: *TUGboat* 10:1 — some section heads are more decorative than others

### The Treasure Chest

A package tour from CTAN—*soul.sty*

When the Treasure Chest is CTAN, there's so much to choose from. But even more... there are so many packages that keep being added! And how to even find out about them, if you don't keep up with notices posted to the newsgroups? This column is one way to try to bring some of these treasures to TUGboat readers, with a quick introduction to the package and some examples of what it can do.

This is the first such column; let me know what aspects are most useful and which ones less so, what additional facets should be examined, what packages cover some of the same issues; which packages do you prefer.

#### 1 Quick tour

**Package:** *soul.sty*  
This is version 1.2, dated 11 Jan. 1999. Upon processing, the file *changes.tex* is generated, and describes the differences (the file is also inside the *.dtx* file).

**Explanation of the name:** "it is only a combination of the two macro names *LaTeX* (*space out*) and *LaTeX* (*underline*)—nothing poetic at all."

**Keywords:** spacing out, letterspacing, underlining, striking out.

**Purpose:** *soul.sty* provides hyphenate-able letterspacing, underlining, and some variations on each. All features are based upon a common mechanism for typesetting text syllable-by-syllable, using TeX's excellent hyphenation algorithm to find the proper hyphenation points. As well, two examples are presented to show how to use the interface provided to address such issues as "an-a-lyzing syl-la-bles". Although the package is optimized for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, it works under plain TeX and L<sup>A</sup>T<sub>E</sub>X 2.09, and is compatible with other packages, too.

**Author:** Melchior Franz  
s@03555net.univie.ac.at

<sup>1</sup> Documented source (*.dtx*) files are a combination of macros and documentation, an evolution of Frank Mittelbach's original Docstrip utility. There are usually two steps: run L<sup>A</sup>T<sub>E</sub>X over the *.dtx* file to get the documentation, and run L<sup>A</sup>T<sub>E</sub>X over the *.dtx* file to get the macros. The *.ins* file is itself generated by the first step, which means you only have to pick up the *.dtx* file—even more compact packaging.

**Compatible with:** plain L<sup>A</sup>T<sub>E</sub>X (old and new). Note: the documentation describes some restrictions when the *soul* package is not used with L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>.

**Location on CTAN:** */macros/latex/contrib/supported/soul*

**Files to fetch:** *soul.dtx* and *example.cfg*

**How to install:** Put files with your other class and style files on your system. Read the top portion of *soul.dtx* (or the file *soul.txt*) for instructions on processing the files (you will need L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>). Notice that the *soul.sty* package is not actually on CTAN; it uses the *.dtx* method of documentation, a wonderful feature in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. If you're unfamiliar with how this works, see footnote 1 for a general overview.

**Files generated:** *soul.ins*, *soul.dvi* (documentation), *soul.toc*, *soul.sty*, *changes.tex*, (as well as the usual *soul.aux* and *.log* files).

**Documentation**  
The documentation is an extensive (26 pages long), with explanations, examples of basic use and variations, that little needs to be said here!

The opening pages are a pleasant introduction to the general notions of emphasis, however it is achieved, and the various opinions which exist on the suitability of their use. There is a pragmatic expressed here, offering the user the choice of options, leaving the reasons for such choices to the user.

The user portion of the documentation provides extensive examples and explanations for creating the various effects (underlining, overstriking, letterspacing).

Chapter 7 (pp. 14-25) provides a detailed explanation of the macros themselves, along with some additional points and tips, so do glance through it.

One nice addition from the author (in collaboration with Stefan Ulrich) is a sample configuration file, *example.cfg*, which shows how to select specific spacing values for different fonts automatically, and store them for local use. As well, the local file (call it *soul.cfg*) and looks easy to read it in automatically via *soul.sty* can be used to store other changes to the package default settings, thus avoiding making changes in either the style file or inserting the customizations into individual source files.

#### 2.1 Table of Contents

1. Introduction

<sup>2</sup> Note: CTAN also has the file *soul.txt* (description of package processing) and *soul.ins* (which can either be fetched, or generated by processing the *.dtx* file).

to Timbuktu, so Addison-Wesley rushed out a small second printing. No corrections were made to that printing.

Two new document-style options have been added: *bestier* for drawing curves and *ifthen* with conditional evaluation and looping commands. A document style that will format text for the ACM transactions' journals is in preparation, and I will be negotiating with the ACM to allow authors to submit either camera-ready copy or L<sup>A</sup>T<sub>E</sub>X input files.

I suspect that many sites have installed L<sup>A</sup>T<sub>E</sub>X without installing the appropriate human system for maintaining it. There should be a site coordinator who is responsible for installing L<sup>A</sup>T<sub>E</sub>X (with any necessary site-specific changes), creating and maintaining the Local Guide, fielding questions from users, and obtaining the latest versions of L<sup>A</sup>T<sub>E</sub>X files.

Leslie Lamport  
Digital Equipment Corporation

TeX is now truly multi-lingual. The restriction on the *triv-up* size has been removed. It is now possible to accommodate up to 65000 languages—although L<sup>A</sup>T<sub>E</sub>X currently has a consistency check that arbitrarily restricts it to 100.

Contrary to what was reported in the previous article, L<sup>A</sup>T<sub>E</sub>X cannot change hyphenation rules on a word by word basis. It is restricted to the language in force at the end of a paragraph. The reason for this is that the value of the *language* parameter is not carried along with the character in the same way as the font information. Most applications should be satisfied with paragraph by paragraph hyphenation. For those that are not, an extension involving an increase in the size of a char node is possible.

Michael J. Ferguson  
INRS-TÉLÉCOMMUNICATIONS

### Multilingual TeX Update

This note updates the extension to TeX that allows for multi-lingual hyphenation reported in TUGboat 6, no. 2 (July 1985): 57-58. A key feature of the extension is that it accommodates standard TeX fonts, including words with accented letters. For details of the features the reader should refer to the TUGboat report. The changes and retractions are as follows:

Figure 31: TUGboat 19:4—and some try to be self-explanatory

### Chapter Mottos and Optional Semi-Parameters in General and for L<sup>A</sup>T<sub>E</sub>X

Reinhard Wonneberger  
Hamburg<sup>1</sup>

#### Abstract

Motto texts will cause some logical and practical difficulties, when they are to be prefixed to chapter headings. To solve them, the motto text should be specified after the chapter heading. To allow for a variety of contained constructs such as footnotes or verbatim, this text must not be read as an ordinary parameter. These antagonistic goals are reconciled in a construct which looks like a parameter but is treated as input text. This concept is a modified version of the technique used for PLAIN footnotes. We give all macros necessary to implement this concept as an extension to L<sup>A</sup>T<sub>E</sub>X.

#### 1 On Mottos

To provide the reader with a glimpse of what is waiting for him, a book or its chapters are sometimes prefaced with mottos. The basic idea of mottos, going right into the heart of a text in one short sentence, can be traced back to the times of ancient Babylonia, the myth of *Atrahasis* starting with such a motto:<sup>2</sup>

*inēma dū aušum*  
When the gods were (also still) men ...

<sup>1</sup> The macros presented here were developed at DESY, Notkestraße 85, D-2000 Hamburg 94. FRG Comments should be sent to R. W. Wonneberger, D-2000 Hamburg 94 or through *Bayer, Chemie* to B03555net.univie.ac.at

<sup>2</sup> Wulfen von Soden: *Mythos von Atrahasis*, Berlin und New York: Walter de Gruyter, 1985, p. 206 from p. 206-212.

From a linguistic point of view, mottos are somewhat similar in function to particles, being both part of the text and a comment on it. So they are better understood in terms of a metatext.<sup>3</sup>

There is a wide range of possible motto texts, reaching from witty to enigmatic, from aphoristic to devotional, from past to present. Normally motto verbatim, this text must not be read as an ordinary parameter. These antagonistic goals are reconciled in a construct which looks like a parameter but is treated as input text.

As far as typesetting is concerned, graphic arrangement of mottos should meet several requirements. The special kind of text will be made clear through emphasis or even a different family of character type, e.g. some serif. A scope, i.e. the range of text the motto applies to, will be expressed by prefixing the motto to an already established unit like a chapter. And finally, aesthetic concepts should be taken into account. So the motto will normally be broken into smaller lines, which may be right-adjusted to stress the frame of the page in connection with a heading. Professional book designers should be consulted on a specific concept for formatting.

Since mottos are normally taken from some source, one might also wish to put names into the index or give some bibliographic information in a footnote. Through these seems to be nothing peculiar about these requirements, implementation of mottos needs

<sup>3</sup> It is fascinating to watch the gradual emergence of such metatexts, which are also used to give direct access to texts, a history still waiting to be written. For some remarks, cf. R. W. Wonneberger and Norbert Henning: *Neue Wege bei der Darstellung altorientalischer Texte*, *Zeitschrift für Sprachwissenschaft* 3 (1984) 303-313, also R. W. Wonneberger: *Bibliographische Bibliographie*, Göttingen: Vandenhoeck & Ruprecht 1984, chapters 3-5.

Figure 33: TUGboat 7:1—the coming of L<sup>A</sup>T<sub>E</sub>X had been foretold

#### Editorial and Production Notes

These Proceedings were prepared with TeX on various Unix workstations and L<sup>A</sup>T<sub>E</sub>X in Geneva. PostScript files for a Linotronic typesetter at 1270 dpi resolution were generated with Tom Rokicki's *dvips* program. From these files Philip Taylor produced the bromides on the Linotronic of the Computing Centre of the University of London. The color pages were completely done in the United States.

The present Proceedings are typeset in the Lucida Bright typeface designed by Bigelow & Holmes. For L<sup>A</sup>T<sub>E</sub>X the Lucid package (coming with L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> in the FMS5 system) for defining the fonts was used and a scaling factor of .94 has been applied to make the pages come out at an information density close to that of Computer Modern at 10pt. The complete set of fonts used is LucidaBright for text, LucidaSans for sans serif, LucidaTypeWriter for typewrite, and LucidaNewMath for the maths.

The authors sent their source files electronically via electronic mail or deposited them with *ftp* on a CERNA machine. Most referees were also able to use *ftp* to obtain a PostScript copy of the paper they had to review, and I got their comments, if practical, via email, which made communication relatively straightforward and fast. I would like to thank the authors for their collaboration in keeping (mostly) to the original production schedule. I also want to express my gratitude to the various referees, who kindly agreed to review the paper assigned to them. I am convinced that their comments and suggestions for improvements or clarifications have made the papers clearer and more informative.

Eight of the contributed papers were in plain L<sup>A</sup>T<sub>E</sub>X while the others used L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. All files associated to a given paper reside in a separate subdirectory in our *tug94/papers* directory, and each of the papers is typeset as a separate job. A makefile residing in our *tug94/papers/tug94* directory takes care that each paper is picked up from its directory and is processed with the right parameters. Information about the page numbers for the given job is written to the aux file using the `\vcenter` command for L<sup>A</sup>T<sub>E</sub>X and by redefining the `\enddocument` command for plain L<sup>A</sup>T<sub>E</sub>X. A sed script then collects this information and writes it into a master file. This master file is read in a subsequent run by using the `\AtBeginDocument` command for L<sup>A</sup>T<sub>E</sub>X and by redefining the `\vcenter` command for plain L<sup>A</sup>T<sub>E</sub>X.

All L<sup>A</sup>T<sub>E</sub>X files were run in native L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> mode (if they were not already coded in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>—about half of the L<sup>A</sup>T<sub>E</sub>X papers were—it was in most cases sufficient to replace `\begin{document}` by `\begin{document} class`). At CERNA we run L<sup>A</sup>T<sub>E</sub>X version 3.1415, based on Karl Berry's *web2t-ε-1* directory structure. This system could be used for most papers without problems, but Haralambous' *Ω*

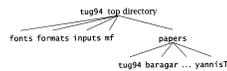


Figure 1: The directory structure for preparing the TUG94 Proceedings

(yanniso), and Phil Taylor's *NyrS* paper, needed the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> extensions, which have not yet been ported to that latest version of *web2t*. Therefore we had to build two special formats (one for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, and one for plain with the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> mods and the older *TeX* 3.14.1 *web2t-ε-1*). The fonts used in Haralambous' *Tiqvah* paper needed 60 instead of the standard 50 font files, so we also had to recompile METAFONT. When the *dvi* files were translated into PostScript with *dvips*, METAFONT would generate the font bitmap *pk* files on the fly, as they were needed, with the desired mode *def*. In total 334 supplementary METAFONT source files were received for running the various papers in the Proceedings.

Although most pictures were available as Encapsulated PostScript files, for two articles (the one by Sotka, and the *Big2Font* paper by Sowa) they could not be printed. Therefore we pasted originals obtained from the respective authors into the relevant places in the text.

#### Acknowledgements

These Proceedings would never have been ready in time were it not for the help of Sebastian Rahtz during the final stages of the production cycle. Building upon his experience gained last year when editing the TUG93 Proceedings, he developed a vastly improved production system for the generation of this year's Proceedings. Together we translated the old TUGboat styles into L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> classes, and used these classes to produce the Proceedings. With the help of Owen Patashnik and Joachim Schrod we also developed a first version of a Chicago-like TUGboat BibTeX bibliography style and introduced the corresponding necessary changes into the class files.

I also want to thank Barbara Beeton, Mimi Burbanck, Pierre Mackay, and Christina Thiele who, together, have read the preprint versions of all papers. They have pointed out several remaining typos and provided me and the authors with many useful comments and suggestions for improvement. Last but not least I want to acknowledge the competence and dedication of Phil Taylor (BIBRC, University of London) during the final production stage of going to film.

Michel Goossens

Figure 32: TUGboat 7:3, the first article produced with L<sup>A</sup>T<sub>E</sub>X (2.09)

Figure 34: TUGboat 15:3—the 1994 Proceedings issue was produced at CERNA

T<sub>E</sub>X 82 without a *lot* of work, regardless of how cleverly one recodes the underlying macros. Because *authors always want to do things their own way*, and *TUGboat* authors were trying to show what they could do with this wonderful new tool.

In any event, by 1984, the *TUGboat* transition was made to T<sub>E</sub>X 82, new fonts, new everything, with no more fuss than accompanied the simultaneous transition of AMS projects. (The one often performed as a test bed for the other.) But a growing number of authors wanted to use L<sup>A</sup>T<sub>E</sub>X, which is a very different beast. The first article written with L<sup>A</sup>T<sub>E</sub>X was published in 7:3 (Fig. 32), although L<sup>A</sup>T<sub>E</sub>X had certainly been mentioned earlier (Fig. 33); note Leslie Lamport’s comment regarding the *Local Guide*, an item always honored more in theory than in practice.

By 1991, the volume of L<sup>A</sup>T<sub>E</sub>X material had increased to the point where the production notes for 12:2 reported a nearly 50/50 split between plain T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. This made grouping of articles in subject areas more difficult for a couple of reasons:

- “Plain” articles can usually be processed in a single run, using a driver file, unless the complement of articles contains a lot of mutually incompatible author macros — a not infrequent occurrence.
- L<sup>A</sup>T<sub>E</sub>X requires that packages be loaded only in the preamble; this is true for both 2.09 and L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>; nearly any package use by an author precludes combining files in a single run.
- In *TUGboat*, if an article ends with more than a half-column empty, the next article may be started on that page; other than using physical paste-up or post-processing, the only way to achieve the desired continuity is to process both articles in the same T<sub>E</sub>X run.

Needless to say, all known methods of “splicing” disparate items have been used to get camera copy ready for the printer.

The next big leap toward L<sup>A</sup>T<sub>E</sub>X occurred with the proceedings of the Santa Barbara meeting in 1994. Michel Goossens, then TUG’s vice president, co-edited the proceedings with Sebastian Rahtz. Both were ardent supporters of L<sup>A</sup>T<sub>E</sub>X, and eager to take advantage of the new features of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. Together they created the first *TUGboat* document class file, and handled all the production as well as the editorial duties at CERN (Fig. 34). (Maintenance of the *TUGboat* document class is now in the care of Robin Fairbairns, to whom many thanks.)

At the same meeting, there was a report on Ω, a new approach to the composition requirements



Figure 35: *TUGboat* 18:1 — the first article produced with Ω

of highly-accented material and non-Western scripts (that is, scripts other than Latin, Greek and Cyrillic). The first article actually produced with Ω (Fig. 35) was set by the author to specs provided by the *TUGboat* production crew. Ω has unfortunately not proved sufficiently stable to be included permanently in the *TUGboat* toolbox, but work continues.

At the 1998 annual meeting, Hàn Thế Thành introduced pdfT<sub>E</sub>X (Fig. 36). This extension to T<sub>E</sub>X permits the use of existing L<sup>A</sup>T<sub>E</sub>X or plain T<sub>E</sub>X input, along with direct output to PDF. Thành’s dissertation (Fig. 37) was published in *TUGboat* several years later.

For a totally different approach to composition, ConT<sub>E</sub>Xt is directed largely toward creating attractive presentations on-line as well as in print, and requires pdfT<sub>E</sub>X. ConT<sub>E</sub>Xt made its appearance in several talks by Hans Hagen at the 1998 annual meeting. One of the resulting articles in the proceedings describes an interactive calculator (Fig. 38); sadly, the on-line version of this article is not interactive, but the figures are very colorful. Hans has created a ConT<sub>E</sub>Xt style for *TUGboat* which has been used for several other articles, but so far always with his assistance.

Improving TeX's Typeset Layout

Hàn Thế Thành
Faculty of Informatics
Masaryk University
Brno, Czech Republic

Abstract

This paper describes an attempt to improve TeX's typeset layout in PDFTeX, based on the adjustment of interword spacing after the paragraphs have been broken into lines. Instead of changing only the interword spacing in order to justify text lines, we also slightly expand the fonts on the line as well in order to minimise excessive stretching of the interword spaces. This font expansion is implemented using horizontal scaling in PDF. When such expansion is used conservatively, and by employing appropriate settings for TeX's line-breaking and spacing parameters, this method can improve the appearance of TeX's typeset layout.

Motivation

There exist many techniques which can be used to produce high quality typeset layout. Most of these are already implemented in TeX, such as ligatures, kerning, automatic hyphenation, and very importantly the algorithm for breaking paragraphs into lines in an optimal way, generally known as "optimum fit".

However, it is still a very difficult task to obtain a uniform level of grayness of the typeset layout, even with the help of these techniques. The primary reason is that it is not possible to ensure that all the interword spaces in different lines are the same. The "optimum fit" algorithm can break the paragraph into lines in the best way, but the amount of interword space depends strongly on many other parameters, such as the paragraph width, the tolerance of glue stretching/shrinking, the amount of interword glue, etc. Considerable effort is often required in order to adjust these parameters to achieve the appropriate break points and to reduce the contrast between the interword spaces in lines. The purpose of our experiment is an attempt to perform this task better by stretching or shrinking the fonts used in each line within reasonable limits. The idea is not really new, as it represents a quite common technique using electronic font scaling in order to expand text lines that do not fit the paragraph width. However this technique is also often regarded as a bad thing, since it is frequently abused in order to rescue "impossible" cases, which often leads to overdoing the scaling and produces really ugly results. In our approach, we try to use this technique

in a rather different way: instead of using font scaling to improve only some "really bad" lines, we try instead to produce a "relatively good" paragraph, which does not contain any lines where the interword spacing is too bad. Then we apply font scaling to each line to reduce the difference between the interword spaces in lines. The limit of font scaling must, of course, be strictly controlled: in fact, the sum of the spaces between the words on a line is often very small in comparison to the sum of the character widths on the same line, so very slightly expanding the fonts may help considerably in improving the interword spacing.

This idea can easily be integrated with TeX because of the biggest strength of TeX - the "optimum fit" algorithm which is implemented in a very flexible manner, in order to handle restrictions on many various parameters in an optimal way. In particular, we perform the implementation in PDFTeX, where the font expansion is currently carried out by horizontal scaling in PDF as a first attempt. Other approaches may be attempted in the future as time allows.

Implementation

PDFTeX is based on the original source of TeX, and employs the changefile mechanism which allows easy access to TeX's internal data structures and simple modification of the relevant program code. Generating PDF output directly from TeX is also an advantage for our task, as we can control the spacing much better than would have been the case had we attempted it via DVI. The process of adjusting interword spacing is as follows:

The Calculator Demo

Hans Hagen
pragma@pi.net

Abstract

Due to its open character, TeX can act as an authoring tool. This article demonstrates that by integrating TeX, METAPOST, JavaScript and PDF, one can build pretty advanced documents. More and more documents will get the characteristics of programs, and TeX will be our main tool for producing them. The example described here can be produced with PDFTeX as well as traditional TeX.

Introduction

When Acrobat Forms were discussed at the PDFTEX mailing list, Phillip Taylor confessed: "... they're one of the nicest features of PDF". Sebastian Ratz told us that he was "... convinced that people are waiting for forms". A few mails later he reported: "I just found I can embed JavaScript in forms, I can see the world is my oyster" after which in a personal mail he challenged me to pick up the Acrobat Forms plugin and wishing me "Happy JavaScripting".



Figure 1: The calculator demo.

At the moment that these opinions were shared, I already had form support ready in CONTEXT, so picking up the challenge was a sort of natural behaviour. In this article I'll describe some of the experiences I had when building a demo document that shows how forms and JavaScript can be used from within TeX. I also take the opportunity to introduce some of the potentials of PDFTeX, so let's start with introducing this extension to TeX.

Where do we stand

While e-TeX extends TeX's programming and typographic capabilities, PDFTeX primarily acts at the back end of the TeX processor. Traditionally, TeX

was (and is) used in the production chain:

ASCII -> TeX -> DVI -> whatever

The most versatile process probably is:

ASCII -> TeX -> DVI -> POSTSCRIPT

or even:

ASCII -> TeX -> DVI -> POSTSCRIPT -> PDF

All functionality that TeX lacks, is to be taken care of by the DVI postprocessing program, and that's why TeX can do color and graphic inclusion. Especially when producing huge files or files with huge graphics, the POSTSCRIPT -> PDF steps can become a nuisance, if only in terms of time and disk space.

With PDF becoming more and more popular, it will be no surprise that Han The Thanh's PDFTeX becomes more and more popular too among the TeX users. With PDFTeX we can reduce the chain to:

ASCII -> TeX -> PDF

The lack of the postprocessing stage, forces PDFTeX (i.e. TeX) to take care of font inclusion, graphic inserts, color and more. One can imagine that this leads to lively discussions on the PDFTEX mailing list and thereby puts an extra burden on the developer(s). Take only the fact that PDFTeX is already used in real life situations while PDF is not stable yet.

To those who know PDF, it will be no surprise that PDFTeX also supports all kind of hyper referencing. The version 1 used when writing this article supports:

- 1. link annotations
2. Currently I'm using beta-version 1.12g.

Figure 36: TUGboat 19:3—the first article with pdfTeX

Micro-typographic extensions to the TeX typesetting system

Hàn Thế Thành
Dissertation

Masaryk University Brno
Faculty of Informatics
October 2000

Figure 37: TUGboat 21:4, Thành's pdfTeX dissertation

Figure 38: TUGboat 19:3—the first article with ConTeXt

XqTeX, the Multilingual Lion: TeX meets Unicode and smart font technologies

Jonathan Kew
SIL International
Horsley Green
High Wycombe HP14 3XL
England
jonathan\_kew@sil.org

Abstract

Professor Donald Knuth's TeX is a typesetting system with a wide user community, and a range of supporting packages and enhancements available for many types of publishing work. However, it dates back to the 1980s and is tightly wedded to 8-bit character data and custom-encoded fonts, making it difficult to configure TeX for many complex script languages.

This paper will introduce XqTeX, a system that extends TeX with direct support for modern OpenType and AAT (Apple Advanced Typography) fonts and the Unicode character set. This makes it possible to typeset almost any script and language with the same power and flexibility as TeX has traditionally offered in the 8-bit, single-script world of European languages. XqTeX is currently available on Mac OS X, but possibly on other platforms in the future) integrates the TeX formatting engine with technologies from both the host operating system (Apple Type Services, CoreGraphics, QuickTime) and auxiliary libraries (CU, TEBOL), to provide a simple yet powerful system for multilingual and multi-script typesetting.

The most significant extensions XqTeX provides are its native support for the Unicode character set, replacing the myriad of 8-bit encodings traditionally used in TeX with a single standard for both input text encoding and font access; and an extended font command that provides direct access by name to all the fonts installed in the user's computer. It also provides a mechanism to access many of the advanced layout features of modern fonts.

Additional features that will also be discussed include built-in support for a wide variety of graphic file formats, and an extended line-breaking mechanism that supports Asian languages such as Chinese or Thai that are written without word spaces.

Finally, we look briefly at some user-contributed packages that help integrate the features of XqTeX with the established PDFX system. Will Robertson's fontspec, sty provides a simple, consistent user interface in PDFX for loading both AAT and OpenType fonts, and accessing virtually all of the advanced features those fonts offer. Ross Moore's outwrite, sty is a package that allows larger PDFX documents to be typeset using native Mac OS X fonts without converting the input text entirely to Unicode, by supporting traditional TeX input conventions for accents and other "special" (non-ASCII) characters.

Editor's note: This article is typeset in Adobe Garamond, with Adobe Minion for the code examples, and processed on the author's Mac OS X machine with XqTeX, so Unicode support was needed in several places.

What is XqTeX?

XqTeX is an extension of the TeX processor, designed to integrate TeX's "typesetting language" and document formatting capabilities with the Unicode/ISO 10646 universal character encoding for all the world's scripts, and with the font technologies available on today's computer systems, including fonts that support complex non-Latin writing systems.

XqTeX is in fact based on e-TeX, and therefore includes a number of well-established extensions to TeX. These include additional registers (\count, \setminus, \box, etc.) beyond the 256 of each that TeX provides; various new conditional commands, tracing features, etc.; and of particular significance for multilingual work, the TeX-Xq extension for bidirectional layout.

The TeX extensions inherited from e-TeX are not discussed further here, as they are already described in the e-TeX documentation<sup>1</sup>, except to note that for right-to-left scripts in XqTeX, it is necessary to set \TXXstate=1 and make proper use of the direction-changing commands \begl, \endgl, \setgl, etc. Without these, there will be some right-to-left behavior due to the inherent directionality defined by the Unicode standard for characters belonging to Hebrew, Arabic and similar scripts, but overall layout will not be correct.

XqTeX was created in order to typeset materials—literacy and educational books, translated Scriptures, linguistic studies, dictionaries, etc.—in a wide range of languages and scripts, including lesser-known ones that are not adequately supported in most existing products. It inherits ideas, and even some code, from an earlier system called PDFX that integrated TeX with the QuickDraw GX graphics system on older Macintosh operating systems.

The name XqTeX was inspired by the idea of a Mac OS X extension (hence the "X" prefix) to TeX, and as one of its intended uses is for bidirectional scripts such as Hebrew and Arabic, the name was designed to be memorable. The second letter should ideally be U (for UNICODE CAPITAL LETTER BEVERSED E), but as few current fonts support this character, it is normal to use a softened "q" glyph. The name is pronounced as if it were written "Xq".

<sup>1</sup> E.g., The e-TeX User Reference Manual, http://www.ctaf.fr/~ut/ut12/ut12/ut12ref.pdf, sec. 1.1.

Figure 39: TUGboat 26:2—XqTeX

12 TUGboat, Volume 2, No. 1

On an IBM 370/3033 with Pascal/VS at Stanford CIT (Eagle Bera).  
 On a VAX (VMS) at Oregon Software (Barry Smith).  
 On an IBM 370/3022 (VM-CMS) with SLAC-Pascal at the University of Pisa (Gianfranco Pinzi). They printed the DVI files on a Versatec.  
 On a Univac 1100/82 at the University of Wisconsin (Ralph Stromquist). Output is to a Compugraphic 8600. (See report, p. 51.)

From the information sent to Stanford, we gather that the Pascal compilers being employed in the installations of TeX are:  
 IBM 370: Pascal-VS, SLAC-Pascal, Pascal-8000  
 UNIVAC: U. of Wisconsin Pascal, Pascal-8000  
 PDP-10: Hamburg Pascal

VAX: (See report by Janet Incepi, p. 49.)

Note: Charles Lawson (Jet Propulsion Lab.—Caltech) has produced two short reports that can help in reprogramming the SYSDEP module of TeX-Pascal. (Both are reprinted in this issue, pp. 20 and 32.)

\*\*\*\*\*

**TeX FONT METRIC FILES**

— or —

What happens when you say `\font=CMR10`  
 David Fuchs

When you tell TeX that you will be using a particular font, it has to find out information about that font. It gets this information from what are known as TFM files. For instance, when you say `\font=CMR10` to TeX (`\font=CMR10` in the old TeX lingo), TeX looks around for a file called `CMR10.TFM`, and reads it in. If `CMR10.TFM` is not to be found, TeX will give you the error message `!lookup failed on file CMR10.TFM`, and you will be out of luck as far as using `CMR10` is concerned.

What does TeX want with the TFM file? Generally speaking, a font's TFM file contains information about the height, width and depth of all the characters in the font, plus kerning and ligature information. So, `CMR10.TFM` might say that the lower-case "d" in `CMR10` is 5.55 points wide, 6.94 points high, etc. This is the information that TeX uses to make its lowest-level boxes—those around characters. See the TeX manual (p. 41) for information about what TeX does with these boxes. Note that TFM files do NOT contain any device-dependent description of the font (such as the raster description of the characters at a certain resolution). Remember that the program TeX does not deal with

pixels. Only device-drivers that read TeX's DVI output files use that sort of information.

Where do TFM files come from? The best way to get a TFM file is with METAFONT. Post designers should include the METAFONT instructions that specify the width, height, etc. of each character they design. The METAFONT manual contains details and examples of how to do this—see the index entries for `charwd`, `charht`, `chardp`, etc. If this is done, then when METAFONT is run on `CMR10`, it produces `CMR10.TFM`. (Depending on what "mode" it is run in, it also makes `CMR10.FNT`, `CMR10.ANT`, `CMR10.VNT`, or `CMR10.OC`. These are all different formats for files containing the raster description of the font. Drivers for various devices require one or another of these files.)

Whatever happened to the TFX format that the TeX and METAFONT manuals actually refer to? In this just a misprint: For TFM! No—TFM files take the place of TFX files. The differences are conceptually small; they both contain more or less the same information. The main reason for changing TeX and METAFONT from using/making TFX files to TFM files is that TFX files were based on 36-bit words. This proved to be a real problem for people running Pascal TeX, especially on 32-bit machines. The format of TFM files assumes 8-bit bytes, packed four to a 32- or 36-bit word. They are readily adapted for use on 16-bit machines, too. While the format was being changed, a few new bits of information were added, too.

What if I have fonts that I want TeX to know about that were not made with METAFONT? Don't despair—we have two programs, `TFTOPL` and `PLTTF`, that convert TFM files to readable, editable format, and back again. For instance, if we run `TFTOPL` on `CMR10.TFM`, it makes `CMR10.PL`, an excerpt of which follows ("P" means a floating-point number is coming up [all dimensions given in this form are in terms of the `DESIGNSIZE`]; "C" means that a character is next; "W" means an octal value for a character that isn't an ASCII printing character is next):

```
(FAMILY CMR)
(DESIGNSIZE R 10.0000000)
(CODINGSCHEME TEX TEX)
(TEXINFO
(SPACE R .3838380)
(STRIKE R .1686070)
(STRIKE R .1111107)
(CHEIGHT R .4444447)
(WID R .0000000)
(EXTRAPAGE R .1111107)
)
```

\*\*\*\*\*

Figure 40: TUGboat 2:1 — 200dpi output is pretty grainy, even reduced from 130%

The most recent addition to the TeX zoo is XeTeX, by Jonathan Kew (Fig. 39). This Unicode-based extension of TeX can use system fonts directly. Jonathan produced the camera copy for this article on his Mac, but he is diligently working on implementations for Unix and Windows that can be included on the next edition of TeX Live.

### Production and distribution

Early issues of TUGboat were produced from a miscellany of sources and output devices. Material prepared "in-house" at AMS was processed using TeX on a DECsystem-20. For the first two issues, this output was magnified to 130% on a 200 dpi Benson-Varian electrostatic printer, and photographically reduced for the press (Fig. 40). Quite a bit of material was submitted as camera-ready copy prepared on a variety of other output devices, with running heads and page numbers pasted on.

The quality of copy prepared "in-house" improved radically with issue 2:2, when AMS installed an Alphatype CRS. This machine had the astounding resolution of 5333dpi, with output on large sheets of resin-coated photographic paper. A great deal of material was still arriving as camera-ready copy, however, and a statement of editorial policy (Fig. 41) encouraged authors to pay attention to the

TUGboat, Volume 3, No. 1

**REPORT ON THE  
TUG STEERING COMMITTEE MEETING**

The Steering Committee meeting in Cincinnati took place in several sessions. At the first, on January 11, the role of the American Mathematical Society in future production of TUGboat was discussed, and other items were suggested for discussion at the second session, an open meeting on January 12.

The following actions were taken, either by the Steering Committee alone or at the open meeting:

- a. Membership for 1981 will be available retroactively through April 30, at \$10.00; thereafter TUGboat Volume 3 will be available at the price of \$10.00 per back issue.
- b. Ordinary subscriptions will be accepted for TUGboat at the same price as individual membership; this is intended primarily for the convenience of libraries.
- c. Effective with the first 1982 issue of TUGboat, the American Mathematical Society can no longer provide free editorial and production services; these services will be charged to TUG at the same rates incurred by internal Society users of similar services. Other actions will be taken to streamline production while maintaining satisfactory quality; see the Statement of Editorial Policy by Barbara Beeton (page 3) for details.
- d. A rough budget was drawn up and presented to the membership, showing the expected cost of various TUG functions for 1982. A redrafted version appears on page 45.
- e. Steering Committee members will be permitted to attend TUG workshops at no charge if they are unable to obtain support from their institutions.
- f. The Finance Committee was requested to investigate the sale of mailing lists and advertisements in TUGboat, after soliciting opinions on the legal and tax consequences of such sales. They were also requested to obtain opinions on the legal and tax consequences of receiving fees for membership, subscriptions and royalties.
- g. The price of Don Knuth's manual for TeX82 will be increased by \$1.00, which will be paid as a royalty to TUG.
- h. A Bylaws Committee was appointed, consisting of Bob Morris, Susan Plass, Lance Carnes, Dave Kellerman, and Craig Platt. They will prepare a report for the next meeting.
- i. Institutional membership will be instituted when TeX82 is ready for distribution. Done of

issues, and she is now joined by other volunteers, whose names and addresses are listed inside the front cover. If you are writing an article in one of the areas listed, please submit paper copy to the appropriate editor; articles of general interest, or in areas not listed, should go to Editor-in-Chief Bob Weiland. Tapes are still welcome, and can be sent directly to me. (See Vol. 2: No. 1, page 53, and No. 3, page 23, and the form in the back of this issue for details on tape content and format.) It is not intended that all columns appear in all issues: if there is no traffic in a particular area, there will be no column. On the other hand, if traffic is exceptionally heavy in a particular area at any point, consideration will be given to publishing a "topical" issue.

It was suggested in Cincinnati that issues be published less frequently. In 1982, an issue will be published after every general meeting in order to report to the membership what happened. The deadline for manuscripts will be a month to six weeks after the end of the meeting. In between, any manuscripts received in Providence will be held until the next scheduled issue, unless it becomes obvious that enough material exists, or an associate editor volunteers to take charge of a special issue.

Copy is solicited in camera copy form, when possible. If copy has been prepared by TeX and is legible, it will be used as submitted, reduced photographically if necessary (which is advisable for copy prepared on an output device with 200 dots/inch or lower resolution), with running heads applied. The dimensions used in the TUGboat header files are: `Vsize 54pc` for one-column pages `Vsize 39pc`, and for two-column pages `Vsize 18.75pc` and `Vpagewr 39pc`. If the copy is to contain leaders which should not be covered up by the TUGboat running heads, `54pc` should be used as the length of the full page. The type used for ordinary text is `cmr10`, on `\baselinestretch 1.2pc`.

Deadlines will be firmly adhered to. Any material received in Providence later than the published deadline (in the announcements box of every issue) will be consigned to the back of the book, as "Late-Breaking News", or else held over for the next issue.

Since TUGboat is itself an advertisement for TeX, it is not our intention to lower quality, but to streamline production. Your attention to formatting of material submitted as camera copy and to the content and commands in material submitted on tape will assist greatly in reaching that goal.

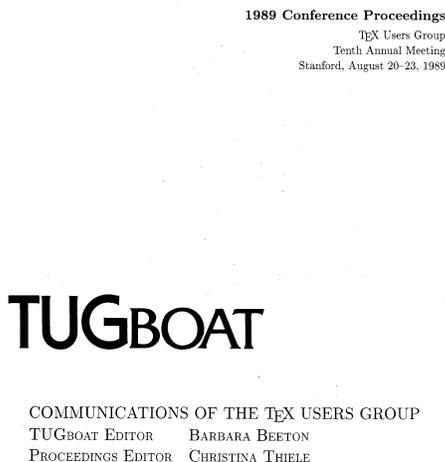
\*\*\*\*\*

Figure 41: TUGboat 3:1 — our first statement of editorial policy

guidelines. (Most authors did; some, I've learned, never read instructions.)

I didn't record when production of camera copy was shifted from the Alphatype to an Autologic APS-5, but 1984 sounds about right. That machine, with a resolution of 1200 dpi, used photographic paper in roll form, and was much less labor-intensive. Since TUGboat is printed on non-glossy paper, the difference in quality was not really noticeable, except perhaps for very tiny print.

In 1988, TUG applied for a second class postal permit, in an attempt to control expenses. One of the requirements for this permit is that at least four issues of the periodical must be published annually. Since the volume of material being submitted was sufficient for about three issues, the board decided that the proceedings of the annual meeting would become the fourth issue. The proceedings of the 1987 and 1988 meetings had already been published as issues of TeXniques, but this had only a limited distribution; inclusion in TUGboat would make the information available to all members. However, the time commitment was greater than I could handle, so the meeting program committee became responsible not only for the acceptance of papers for the meeting, but also for the editing of the proceedings. A member of the committee was designated to be



VOLUME 10, NUMBER 4      DECEMBER 1989  
 PROVIDENCE • RHODE ISLAND • U.S.A.

Figure 42: TUGboat 10:4—the first TUGboat proceedings issue

Lexicography with TeX

JÖRGEN L. PIND  
 Institute of Lexicography  
 University of Iceland  
 Reykjavík  
 Iceland  
 jorgens@lexis.hi.is

ABSTRACT

At the Institute of Lexicography at the University of Iceland, TeX is used for the typesetting of dictionaries. Currently we are in the process of bringing out a large etymological dictionary which is typeset in TeX with PostScript fonts. Details of this project are presented. The value of generic or logical coding over typographical coding is emphasized.

1. Background

In this paper I will discuss the use of TeX in the work carried out at the Institute of Lexicography of the University of Iceland. The Institute was founded in 1948 and has as its major aim the production of an historical dictionary of Icelandic from 1540 (when the first printed book appeared in Iceland) up to the present, a dictionary somewhat along the lines of the Oxford English Dictionary.

During the past forty years, a lot of material has been gathered for the dictionary. The main collection of the Institute comprises some 2.5 million dictionary slips; others include, for instance a collection of words from the spoken language. These other collections contain perhaps 300,000 slips in all. Near the end of 1982, it was decided to begin evaluating the collection with the aim of publishing an historical dictionary of the language. At the same time it was decided to embark on computerizing the Institute itself.

The first computational project involved registering the main collection so as to open more paths into the collection itself. A database of all the words contained in the collection was set up. The word class, date of oldest and newest citation, the oldest source, number of citations kept in the collection and the word type (whether the word is a compound, an affixed word or a simple word) were registered for each word. This database contains a total of just over 600,000 words. This is a surprisingly high figure but is explained in part by word-compounding, which is an active process in the Germanic languages, not the least in Icelandic.

In some respects, this database file can be viewed as a first approximation to a dictionary although a very primitive one, since it does not have any grammatical analysis to speak of. Yet, because the material is stored in a database (as opposed to a linear alphabetical order), it does enable us to escape from the "tranny of the alphabet" and gives us multiple access paths to the collections of the Institute.

The editing of historical dictionaries has usually proceeded in alphabetical order, the work being brought out in installments over a period of decades. This is an approach which is in many respects less than ideal since the editor is forced to deal with words which do not form a coherent set under any reasonable linguistic criterion. We would therefore like to proceed in a different manner, dealing with individual word classes at a time. The availability of the computer makes this relatively easy to accomplish. It has now been decided by the governing board of the Institute that the editing work will concentrate on the verbs with the aim of producing an historical dictionary of verbs as the first volumes of what will hopefully later become a comprehensive historical dictionary of Icelandic.

This work was begun in 1985. The editorial strategy involves some novelties compared with traditional methods (e.g., Kahn 1982) in that each citation is furnished with a set of editorial descriptors detailing the grammatical and semantic features of the citation itself. This is done on-line with the

Figure 43: TUGboat 10:4, it hadn't fully sunk in that wide pages are hard to read

E-TeX: Guidelines for Future TeX Extensions

Frank Mittelbach  
 Electronic Das System (Deutschland) GmbH  
 Eisenstraße 56 (N 15), D-6090 Rüsselsheim, Federal Republic of Germany  
 Td. +49 6142 80247  
 Elnet: ysf@schlurwacke

Abstract

With the announcement of TeX 3.0, Don Knuth acknowledged the need of the (ever growing) TeX community for an even better system. But at the same time, he made it clear, that he will not get involved in any further enhancements that would change *The TeXbook*.

TeX started out originally as a system designed to typeset its author's own publications. In the meantime it serves hundreds of thousands of users. Now it is time, after ten years' experience, to step back and consider whether or not TeX 3.0 is an adequate answer to the typesetting requirements of the nineties.

Output produced by TeX has higher standards than output generated automatically by most other typesetting systems. Therefore, in this paper we will focus on the quality standards set by typographers for hand-typeset documents and ask to what extent they are achieved by TeX. Limitations of TeX's algorithms are analyzed; and missing features as well as new concepts are outlined.

1 Introduction

Last year at Stanford we celebrated the tenth birthday of the TeX project. Up to now, TeX has served thousands of users well and we expect it will continue to do so in the future. The longevity of TeX lies in:

- the quality of its output
- its universal availability
- and its stability.

In the last few years, more and more users brought TeX from the universities into industry where it was challenged by new applications [2]. But time does not stand still, and what was at the top of its profession yesterday might prove to be obsolete tomorrow. TeX is still state of the art for the tasks it was designed to accomplish, but, with the growing understanding from several years' usage, we can now see where it will fail in high quality typesetting.

As a result of user pressure [27], Don Knuth announced a new version of TeX at Stanford, acknowledging the fact, that indeed the need for 8-bit input [19]. At the same time, he made it clear, that he had decided to retire from this project and return to his long delayed topic "The Art of Computer Programming".

So TeX is finally frozen, and any further development will result in a different system no longer maintained by Knuth. The main purpose, therefore, of this paper is to give an overview of high quality typesetting requirements (covered and not covered by TeX 3.0) thereby, we hope, channeling future developments so that we do not end up with several incompatible "TeX-based systems", but rather with one system that will provide the same characteristics (i.e., quality, portability, and availability) as the current program.

TeX was designed as a low-level formatter, a stable kernel, of a typesetting system where extensions at both ends would be possible to take into account developments in printing technology (back end) and in user interfaces (front end) [14]. Thus, complaints about user unfriendliness of TeX are unfounded, for since such requirements can be handled by front ends either written in the TeX language itself like LaTeX and, therefore, fully portable, or in an external language like ArborText's Publisher, or VAX Document, etc. These systems use TeX or a TeX-based system as the ultimate formatter but provide a user-friendly interface [32].

When we discuss missing features, we must distinguish carefully between things which can and should be handled by a front end system and things

Figure 44: TUGboat 11:3, but we learned before the next proceedings issue

Typesetting the Byzantine Cappelli

Philip TAYLOR  
 The Computer Centre, Royal Holloway,  
 University of London, TW20 0EX,  
 United Kingdom  
 email: P.Taylor@rhul.ac.uk

Abstract

An overview of the author's rôle in the preparation of the forthcoming *Lexicon of Abbreviations & Ligatures in Greek Minuscule Hands*, with particular emphasis on two challenges: sorting TeX markup for polytonic Greek using multiple comment keys, and deriving statistical data which could be used to provide input to the book design.

Introduction

One of the greatest pleasures that I get from my position as Webmaster at Royal Holloway, University of London, is the only-too-rare opportunity to work with truly gifted and dedicated scholars. For the last few years, I have been truly privileged to be able to work with Miss Julian Chrysostomides, Director of our Hellenic Institute, and with Dr Charalambos Dendrinos, a Research Fellow within the same Institute. These two extraordinary scholars have both devoted a considerable portion of their lives to the collection, collation and preparation of material for a *Lexicon of Abbreviations & Ligatures in Greek Minuscule Hands* which is intended to do for Byzantine scholarship what Adriano Cappelli's *Dizionario di Abbreviature latine ed italiane* has been doing for Latin scholarship for the past 100 years.

deciphering these, although for Latin scholars Cappelli's *Dizionario* provides an invaluable tool.

Only too aware of the difficulties that their students were experiencing in attempting to decipher Byzantine manuscripts, Julian & Charalambos decided to compile a Byzantine dictionary that would provide their students, and future scholars, with a key to those scribal notations which were most likely to cause problems in interpretation. For over five years, these two scholars have been painstakingly researching and deciphering hundreds if not thousands of individual manuscripts and recording the results of their work, initially using fairly primitive technology such as Windows 3.1's *Cardfile* and Eberhard Mattes' *emTeX* but more recently using spreadsheet technology (Microsoft's *Excel*) and the TeX Live Windows implementation of Bas Tjebk's *Thick's Pdf(LA)TeX* by Fabrice Popineau.

The Work of the Scholars

Although locating and obtaining copies of the manuscripts requires a not-insignificant amount of time, I will concentrate here on the tasks which the scholars undertake once the copies have been received. Each scribal notation that is potentially of interest is identified and scanned, and any artifacts that might serve to confuse are eliminated using a light pen and suitable software (ASC's *Paintshop PRO*). The resulting "clean" image is then stored as a PDF file using a fixed naming convention, and a corresponding entry made in an *Excel* spreadsheet; this entry contains the filename, the transcription, an explanation (if the notation is an abbreviation or similar) and the provenance (typically the date, but occasionally a more detailed provenance where this is felt to be important). Last this creates the impression that the rôle of the scholars is trivial, let me

Δϕ	(ϕ) alfa	xv m.	Δ	(ϕ) alfoi, -alfoi xvi
Δϕ	(A) Augustus, - Augustinus	xv	Δϕ	(ϕ) alfa xv m.
Δϕ	(ϕ) alfoi	xv t.	Δϕ*	(ϕ*) alfoim xv m.
Δϕ	(ϕ) alfoi	xv t.	Δϕ	(ϕ*) alfoi xv t.

Figure 1: A fragment from Cappelli's *Dizionario*

For both Latin and Byzantine scholars, the task of deciphering manuscripts which may be more than a thousand years old is not simply one of reading a long-dead scribe's handwriting; far more difficult is the task of identifying and correctly interpreting the various abbreviations, ligatures and other scribal shorthand notations that he or she may have used. Even a skilled palaeographer may have difficulty in

Figure 45: TUGboat 26:2, and we try to keep improving

**Production Notes**

Barbara Beeton

**A new approach to TUGboat production**

Owing to various circumstances beyond the Editor's control, time available for TUGboat production has diminished to the point where it is no longer possible for the regular issues of TUGboat to remain a one-person operation.

As is quite obvious, this issue is embarrassingly late. But rather than trying to explain why it is late, I would like to describe what has been done to try to avoid such delays in the future.

Mimi Burbank and the system management at SCRI—the Supercomputer Computation Research Institute at Florida State University—have kindly made available copious disk space, login access, and a group identity for a core team of volunteers: Mimi, Robin Fairbairns, Michel Goussens, Sebastian Rahtz, Christina Thiele, and myself. Every member of this team has previous experience in editing or producing TUGboat, proceedings issues, or similar TEX publications, so they have been able to “hit the ground running.”

In the space allotted, we have set up a full, isolated (B)TEX system and TUGboat work areas. Remaining in a management position, I have collected the tree with the material collected for issues 15(4), 16(1), et seq., identified which ones are encoded using plain or L<sup>A</sup>T<sub>E</sub>X conventions, and encouraged the team members to work first on items that match their interests and expertise. Articles are returned to me as PostScript files to be printed and given a final reading. I have edited the input files directly, where practical, and provided comments by e-mail to the “handlers” regarding adjustments in format. The final version is again delivered in PostScript form for printing and inclusion in a growing pile of printer-ready copy. No item has been slighted, with the result that 16(1) is nearly ready to put together, and should be sent to the printer—and thence to members—in no more than a month from 15(4). As I will be out of town for much of this interim, Mimi Burbank has agreed to be the manager for 16(1).

The plan for issue 16(2) is a bit different. For some time, the Publications Committee has been discussing the idea of theme issues—issues devoted to a single topic of narrower or wider scope—under the direction of a guest editor. 16(2) will be the first of such issues, containing articles related to electronic documents, in particular SGML, HTML, hypertext, Acrobat, . . . , with Malcolm Clark in

charge. Topics for future theme issues will be announced as plans become more firm; one theme issue per year is currently foreseen. Suggestions are welcome for both topics and prospective guest editors.

**Input and input processing**

Electronic input for articles in this issue was received by e-mail, and was also retrieved from remote sites by anonymous ftp.

In addition to text and various files processable directly by TEX, the input to this issue includes METAFONT source code and many encapsulated PostScript files. More than 200 files were required to generate the final copy; over 100 more contain earlier versions of articles, auxiliary information, and records of correspondence with authors and referees. These numbers represent input files only; .dvi files, device-specific translations, and fonts (.t<sub>1</sub>m files and rasters) are excluded from the total.

Most articles as received were fully tagged for TUGboat, using either the usual plain-based or L<sup>A</sup>T<sub>E</sub>X conventions.

By number, 47% of the articles, and 63% of the pages in this issue are in L<sup>A</sup>T<sub>E</sub>X. (For ease of production, three mostly-text items which were originally prepared using L<sup>A</sup>T<sub>E</sub>X were converted to plain, and one, from plain to L<sup>A</sup>T<sub>E</sub>X.) L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub> was the version used, thanks to some major systems work by Robin Fairbairns and Sebastian Rahtz.

Font work was required for the Indica article by Haralambous, for MacKay's recycle logo, and for the Chinese fragment in the EuroTEX'94 report.

Articles were processed individually by members of the team according to their own preferred methods, and the final inputs and output (PostScript) files delivered to the Editor for compilation into an issue. The Editor created the table of contents, the cover and front matter, printed out all the files, checked the copy and corrected it to the printer.

**Output**

The bulk of this issue was prepared at SCRI on an IBM RS6000 running AIX, using the WeatC implementation of TEX. The remainder was run at the American Mathematical Society from files installed on a VAX 6320 (VMS) and TEX'ed on a server running under UNIX on a Solbourne workstation. Output was printed at AMS at 600 dpi on an HP LaserJet 4M plus; this was used rather than a typesetter for reasons of both cost and speed.

How to Create a T<sub>E</sub>X Journal: A Personal Journey**Late-Breaking News****Production Notes**

Mimi Burbank

It is April, and the trees and flowers are blooming, and thoughts turn to new life—and the new T<sub>E</sub>X Live CD, which is included in this issue. Many thoughts fly through my feeble mind as I think about production of this issue—the first being that I am using the T<sub>E</sub>X Live 4 setup for production. I wanted to use each set of binaries available on the CD plus some extras, with the exception of the win32. I even managed to build a set of binaries on my own! The following binaries were used for production of this issue:

```
alpha-osf3.2      mips-sgi-irix5.3
alpha6-osf4.0d   rs6000-aix4.1.4.0
hppa1-1-ppa10.10 rs6000-aix4.2.1.0
i686-linux-gnulib1 sparc-solaris2.5.1
mips-irix6.2      sparc-solaris2.7
```

I tend to “play” (more proficient users would call this “beta testing”) with the binaries until they are complete. I like to think I represent the “novice” when it comes to setting up a T<sub>E</sub>X system—and always tell the TLA team members that, “If I can do it, anyone can!” I encourage all of you to read the T<sub>E</sub>X Live documentation, and become familiar with just what is on your system. There is a world of documentation available with all of the packages on the CD—all ready for your use and enjoyment.

**Output.** The final camera copy was prepared at SCRI on multiple workstations, using the T<sub>E</sub>X Live 4 setup, which is based on the WeatC L<sup>A</sup>T<sub>E</sub>X implementation version 7.3 by Karl Berry and Olof Weber. PostScript output, using outline fonts, was produced using Radical Eye Software's *dvipk*(k) 5.85, and printed on an HP LaserJet 4000 TN printer at 1200dpi.

This issue also marks a new trend in the production of TUGboat—we will also be transferring several PostScript files to the printer.

**TUGboat Web Pages**

I have been working to update and complete the TUGboat web pages for inclusion in the TLA CD. During this process, I was overcome by guilt feelings—because we promised to begin this process some time in 1996—and somewhere toward the end of March, I decided that we really needed to put up all of the contents files for every sin-

gle issue of TUGboat! The contents are archived in special files which may be run to produce the Contents pages for each volume (or more information see <http://ctan.tug.org/tex-archive/users/gp/tug/tugboat/t-od-cv/>). During the process, several comments were made about attempting to standardise the “naming” of the TUGboat directories on the TUG web server, so I also had to go in and rename every directory, and then edit each contents file, globally substituting one string for another. This took approximately three full days, and though the contents files are not as “nice” for the older issues, they at least are present and may be viewed at <http://www.tug.org/TUGboat/contents.html>. Any errors are my own; please send comments to [sebastian@tug.org](mailto:sebastian@tug.org) and I'll make changes as quickly as I have time to do so.

My primary reason for mentioning the above is to encourage you to look at the early contents files. I have been a member of TUG since 1986, and having attended nine annual meetings between 1986 and 1996, I must admit to taking a nostalgic trip down “memory lane” as I opened and chosed each of the contents files. For those who are longstanding members of TUG, there is quite a bit of history simply in the contents pages, and I enjoyed meeting “old friends” there. I only hope that we will not have to wait too much longer to be able to actually convert some of the earlier .tex files to a readable format on the web. This of course will require more volunteers who have time to convert some of the older files into a format which can be processed either using the original software or recoding files to run with the current version of L<sup>A</sup>T<sub>E</sub>X.

This will also require permission of every author who ever published in TUGboat! And not only for the TUG web pages. Discussion has begun with regard to making a CD of TUGboat articles. More information on this will appear on the web pages at <http://www.tug.org/TUGboat>, as well as future issues of TUGboat.

**Coming In Future Issues.** The next issue will contain an article entitled “A short introduction to font characteristics” by Maarten Gelderman (reprinted with permission from the editors of the Dutch MAPS). Also scheduled for publication in the next issue is an article describing the latest T<sub>E</sub>X system.

© Mimi Burbank  
SCRI, Florida State University,  
Tallahassee, FL 32306-4130  
mim@scri.fsu.edu

**Figure 46:** TUGboat 15:4—the transition to production at SCRI

the responsible Proceedings Editor. The first person to take on this challenge was Christina Thiele (Fig. 42). Many of the decisions on the style of the proceedings issues grew out of Christina's ideas and opinions, and Christina remains to this day a valuable member of the TUGboat editorial team.

Articles in the first TUGboat proceedings issue were presented as a single, wide column (Fig. 43). This validated the original contention that text of such great width was difficult to read, and a modified two-column format was introduced for the 1990 proceedings (Fig. 44). With minor modifications—the abstract is now wider, though still less than 5 inches—this format is still in use today (Fig. 45).

Not only was the editorial job getting to be more than I could handle and keep TUGboat on schedule, but the workload at the AMS print shop was growing, and it was necessary to look for another printer. With the help and encouragement of the group that had done such a fine job with the 1994 proceedings, a production team was established. Mimi Burbank, with the support of her employer, the Florida State University Supercomputer Computations Research Institute (SCRI), provided a new production home (Fig. 46). With remote logins at SCRI, everyone involved in the production ef-

**Figure 47:** TUGboat 20:1—TUGboat goes electronic, both delivering copy to the printer, and posting on-line

fort could work as effectively across the Internet as they could “at home”. Printing and distribution was contracted to Cadmus, a long-established printer of technical journals on the Eastern Shore of Maryland.

At first, physical camera copy was sent to Cadmus, but when they offered the capability of receiving copy in the form of PostScript files, we tried it out, and found that it worked (Fig. 47). TUG had a stable web site by this time, and TUGboat tables of contents had been posted regularly upon publication of each issue. With the routine processing of files to PostScript, and the ability to convert these to a form readable with a browser, it was decided to try to post the entire TUGboat archive on the web site. Since some decisions regarding copyright meant that TUG didn't have clear title to the material, this in turn meant that permission would have to be obtained from every author who had ever published in TUGboat.

Unfortunately, TUGboat suffered a drought of submissions, and that, along with delays in receiving files from meeting presentations, snowballed into a serious production delay. The mailing permit was terminated after the 2002 volume, allowing a cut-back to three issues per year.

By 2003, PostScript files sent to the printer had been supplanted by PDF files; PDF files were already



ceased when it became practicable to obtain the errata files via a network connection. (Remember—the Internet didn’t exist when T<sub>E</sub>X was launched.) Of course, major upgrades to the software and CM fonts have always been announced in *TUGboat*, and Don Knuth has been a significant contributor of other material as well, if only in the form of transcribed question and answer sessions (Fig. 49).

The topic of output devices was very “hot” in the early days of T<sub>E</sub>X, and a recurring section contained reports on the devices that users had managed to implement, and examples of output from the devices. (One of my favorites was a Diablo daisy-wheel printer, with a driver cobbled together by Timothy Murphy, although no sample was published.) Beginning with issue 3:2 (Fig. 50) a chart appeared in most issues; the run ended in 11:4, when device drivers and laser printers were no longer a novelty.

A decision that didn’t have a visual effect on *TUGboat* but did have an impact on the quality of the content was the implementation of a peer review process for all technical submissions. This practice was initiated in 1990 with particularly strong encouragement from Nelson Beebe. The goal of this review is not to reject material—there has never been a real problem of over-supply—but to ensure that what is published is complete, accurate, and can be understood by a reader with the requisite background. (We do *not* wish to be a closed society of “great experts talking to other great experts”.) Some really fine tutorials have emerged from this practice. However, it has been a real disappointment for me that many T<sub>E</sub>X practitioners who might be best able to write cogently for novices have either pled lack of time to do so, or have directed their efforts solely to writing books, presumably yielding to the profit motive.

Fonts and language support have figured prominently in *TUGboat*’s pages. Although this may be an area of specialized interest (at least one suggestion was received that it should be less prominent), T<sub>E</sub>X may provide scholars working with obscure languages their only practical means of producing documents with fonts of good quality. Just a few of the language- and font-specific topics covered are Coptic, Arabic math (Fig. 51), Byzantine music, classical Greek, Bengali, Thai (Fig. 52), Hebrew, Deseret, the list goes on . . .

**The future . . .**

Maybe it’s time to think about handing this job off to someone else. Karl has been especially supportive and helpful, nagging authors and doing yeoman

The command `\ab{expr}` or `_{expr}` gives `expr` as an index. The index `expr` should be given within braces unless it consists of a single token. The command `_` does not change the direction of `expr`. It can therefore be used for a single token.

$$\$(\{1\}\{\text{ab}\{177\}\}+5 ; \$(\{\text{ab}\{2\}\}+\$(\{1\}\{\text{ab}\{3\}\})+\$(\{4\}\{\text{ab}\{5\}\})+\$(\{6\}\{\text{ab}\{7\}\})+\$(\{8\}\{\text{ab}\{9\}\})+\$(\{10\}\{\text{ab}\{11\}\})+\$(\{12\}\{\text{ab}\{13\}\})+\$(\{14\}\{\text{ab}\{15\}\})+\$(\{16\}\{\text{ab}\{17\}\})+\$(\{18\}\{\text{ab}\{19\}\})+\$(\{20\}\{\text{ab}\{21\}\})+\$(\{22\}\{\text{ab}\{23\}\})+\$(\{24\}\{\text{ab}\{25\}\})+\$(\{26\}\{\text{ab}\{27\}\})+\$(\{28\}\{\text{ab}\{29\}\})+\$(\{30\}\{\text{ab}\{31\}\})+\$(\{32\}\{\text{ab}\{33\}\})+\$(\{34\}\{\text{ab}\{35\}\})+\$(\{36\}\{\text{ab}\{37\}\})+\$(\{38\}\{\text{ab}\{39\}\})+\$(\{40\}\{\text{ab}\{41\}\})+\$(\{42\}\{\text{ab}\{43\}\})+\$(\{44\}\{\text{ab}\{45\}\})+\$(\{46\}\{\text{ab}\{47\}\})+\$(\{48\}\{\text{ab}\{49\}\})+\$(\{50\}\{\text{ab}\{51\}\})+\$(\{52\}\{\text{ab}\{53\}\})+\$(\{54\}\{\text{ab}\{55\}\})+\$(\{56\}\{\text{ab}\{57\}\})+\$(\{58\}\{\text{ab}\{59\}\})+\$(\{60\}\{\text{ab}\{61\}\})+\$(\{62\}\{\text{ab}\{63\}\})+\$(\{64\}\{\text{ab}\{65\}\})+\$(\{66\}\{\text{ab}\{67\}\})+\$(\{68\}\{\text{ab}\{69\}\})+\$(\{70\}\{\text{ab}\{71\}\})+\$(\{72\}\{\text{ab}\{73\}\})+\$(\{74\}\{\text{ab}\{75\}\})+\$(\{76\}\{\text{ab}\{77\}\})+\$(\{78\}\{\text{ab}\{79\}\})+\$(\{80\}\{\text{ab}\{81\}\})+\$(\{82\}\{\text{ab}\{83\}\})+\$(\{84\}\{\text{ab}\{85\}\})+\$(\{86\}\{\text{ab}\{87\}\})+\$(\{88\}\{\text{ab}\{89\}\})+\$(\{90\}\{\text{ab}\{91\}\})+\$(\{92\}\{\text{ab}\{93\}\})+\$(\{94\}\{\text{ab}\{95\}\})+\$(\{96\}\{\text{ab}\{97\}\})+\$(\{98\}\{\text{ab}\{99\}\})+\$(\{100\}\{\text{ab}\{101\}\})+\$(\{102\}\{\text{ab}\{103\}\})+\$(\{104\}\{\text{ab}\{105\}\})+\$(\{106\}\{\text{ab}\{107\}\})+\$(\{108\}\{\text{ab}\{109\}\})+\$(\{110\}\{\text{ab}\{111\}\})+\$(\{112\}\{\text{ab}\{113\}\})+\$(\{114\}\{\text{ab}\{115\}\})+\$(\{116\}\{\text{ab}\{117\}\})+\$(\{118\}\{\text{ab}\{119\}\})+\$(\{120\}\{\text{ab}\{121\}\})+\$(\{122\}\{\text{ab}\{123\}\})+\$(\{124\}\{\text{ab}\{125\}\})+\$(\{126\}\{\text{ab}\{127\}\})+\$(\{128\}\{\text{ab}\{129\}\})+\$(\{130\}\{\text{ab}\{131\}\})+\$(\{132\}\{\text{ab}\{133\}\})+\$(\{134\}\{\text{ab}\{135\}\})+\$(\{136\}\{\text{ab}\{137\}\})+\$(\{138\}\{\text{ab}\{139\}\})+\$(\{140\}\{\text{ab}\{141\}\})+\$(\{142\}\{\text{ab}\{143\}\})+\$(\{144\}\{\text{ab}\{145\}\})+\$(\{146\}\{\text{ab}\{147\}\})+\$(\{148\}\{\text{ab}\{149\}\})+\$(\{150\}\{\text{ab}\{151\}\})+\$(\{152\}\{\text{ab}\{153\}\})+\$(\{154\}\{\text{ab}\{155\}\})+\$(\{156\}\{\text{ab}\{157\}\})+\$(\{158\}\{\text{ab}\{159\}\})+\$(\{160\}\{\text{ab}\{161\}\})+\$(\{162\}\{\text{ab}\{163\}\})+\$(\{164\}\{\text{ab}\{165\}\})+\$(\{166\}\{\text{ab}\{167\}\})+\$(\{168\}\{\text{ab}\{169\}\})+\$(\{170\}\{\text{ab}\{171\}\})+\$(\{172\}\{\text{ab}\{173\}\})+\$(\{174\}\{\text{ab}\{175\}\})+\$(\{176\}\{\text{ab}\{177\}\})+\$(\{178\}\{\text{ab}\{179\}\})+\$(\{180\}\{\text{ab}\{181\}\})+\$(\{182\}\{\text{ab}\{183\}\})+\$(\{184\}\{\text{ab}\{185\}\})+\$(\{186\}\{\text{ab}\{187\}\})+\$(\{188\}\{\text{ab}\{189\}\})+\$(\{190\}\{\text{ab}\{191\}\})+\$(\{192\}\{\text{ab}\{193\}\})+\$(\{194\}\{\text{ab}\{195\}\})+\$(\{196\}\{\text{ab}\{197\}\})+\$(\{198\}\{\text{ab}\{199\}\})+\$(\{200\}\{\text{ab}\{201\}\})+\$(\{202\}\{\text{ab}\{203\}\})+\$(\{204\}\{\text{ab}\{205\}\})+\$(\{206\}\{\text{ab}\{207\}\})+\$(\{208\}\{\text{ab}\{209\}\})+\$(\{210\}\{\text{ab}\{211\}\})+\$(\{212\}\{\text{ab}\{213\}\})+\$(\{214\}\{\text{ab}\{215\}\})+\$(\{216\}\{\text{ab}\{217\}\})+\$(\{218\}\{\text{ab}\{219\}\})+\$(\{220\}\{\text{ab}\{221\}\})+\$(\{222\}\{\text{ab}\{223\}\})+\$(\{224\}\{\text{ab}\{225\}\})+\$(\{226\}\{\text{ab}\{227\}\})+\$(\{228\}\{\text{ab}\{229\}\})+\$(\{230\}\{\text{ab}\{231\}\})+\$(\{232\}\{\text{ab}\{233\}\})+\$(\{234\}\{\text{ab}\{235\}\})+\$(\{236\}\{\text{ab}\{237\}\})+\$(\{238\}\{\text{ab}\{239\}\})+\$(\{240\}\{\text{ab}\{241\}\})+\$(\{242\}\{\text{ab}\{243\}\})+\$(\{244\}\{\text{ab}\{245\}\})+\$(\{246\}\{\text{ab}\{247\}\})+\$(\{248\}\{\text{ab}\{249\}\})+\$(\{250\}\{\text{ab}\{251\}\})+\$(\{252\}\{\text{ab}\{253\}\})+\$(\{254\}\{\text{ab}\{255\}\})+\$(\{256\}\{\text{ab}\{257\}\})+\$(\{258\}\{\text{ab}\{259\}\})+\$(\{260\}\{\text{ab}\{261\}\})+\$(\{262\}\{\text{ab}\{263\}\})+\$(\{264\}\{\text{ab}\{265\}\})+\$(\{266\}\{\text{ab}\{267\}\})+\$(\{268\}\{\text{ab}\{269\}\})+\$(\{270\}\{\text{ab}\{271\}\})+\$(\{272\}\{\text{ab}\{273\}\})+\$(\{274\}\{\text{ab}\{275\}\})+\$(\{276\}\{\text{ab}\{277\}\})+\$(\{278\}\{\text{ab}\{279\}\})+\$(\{280\}\{\text{ab}\{281\}\})+\$(\{282\}\{\text{ab}\{283\}\})+\$(\{284\}\{\text{ab}\{285\}\})+\$(\{286\}\{\text{ab}\{287\}\})+\$(\{288\}\{\text{ab}\{289\}\})+\$(\{290\}\{\text{ab}\{291\}\})+\$(\{292\}\{\text{ab}\{293\}\})+\$(\{294\}\{\text{ab}\{295\}\})+\$(\{296\}\{\text{ab}\{297\}\})+\$(\{298\}\{\text{ab}\{299\}\})+\$(\{300\}\{\text{ab}\{301\}\})+\$(\{302\}\{\text{ab}\{303\}\})+\$(\{304\}\{\text{ab}\{305\}\})+\$(\{306\}\{\text{ab}\{307\}\})+\$(\{308\}\{\text{ab}\{309\}\})+\$(\{310\}\{\text{ab}\{311\}\})+\$(\{312\}\{\text{ab}\{313\}\})+\$(\{314\}\{\text{ab}\{315\}\})+\$(\{316\}\{\text{ab}\{317\}\})+\$(\{318\}\{\text{ab}\{319\}\})+\$(\{320\}\{\text{ab}\{321\}\})+\$(\{322\}\{\text{ab}\{323\}\})+\$(\{324\}\{\text{ab}\{325\}\})+\$(\{326\}\{\text{ab}\{327\}\})+\$(\{328\}\{\text{ab}\{329\}\})+\$(\{330\}\{\text{ab}\{331\}\})+\$(\{332\}\{\text{ab}\{333\}\})+\$(\{334\}\{\text{ab}\{335\}\})+\$(\{336\}\{\text{ab}\{337\}\})+\$(\{338\}\{\text{ab}\{339\}\})+\$(\{340\}\{\text{ab}\{341\}\})+\$(\{342\}\{\text{ab}\{343\}\})+\$(\{344\}\{\text{ab}\{345\}\})+\$(\{346\}\{\text{ab}\{347\}\})+\$(\{348\}\{\text{ab}\{349\}\})+\$(\{350\}\{\text{ab}\{351\}\})+\$(\{352\}\{\text{ab}\{353\}\})+\$(\{354\}\{\text{ab}\{355\}\})+\$(\{356\}\{\text{ab}\{357\}\})+\$(\{358\}\{\text{ab}\{359\}\})+\$(\{360\}\{\text{ab}\{361\}\})+\$(\{362\}\{\text{ab}\{363\}\})+\$(\{364\}\{\text{ab}\{365\}\})+\$(\{366\}\{\text{ab}\{367\}\})+\$(\{368\}\{\text{ab}\{369\}\})+\$(\{370\}\{\text{ab}\{371\}\})+\$(\{372\}\{\text{ab}\{373\}\})+\$(\{374\}\{\text{ab}\{375\}\})+\$(\{376\}\{\text{ab}\{377\}\})+\$(\{378\}\{\text{ab}\{379\}\})+\$(\{380\}\{\text{ab}\{381\}\})+\$(\{382\}\{\text{ab}\{383\}\})+\$(\{384\}\{\text{ab}\{385\}\})+\$(\{386\}\{\text{ab}\{387\}\})+\$(\{388\}\{\text{ab}\{389\}\})+\$(\{390\}\{\text{ab}\{391\}\})+\$(\{392\}\{\text{ab}\{393\}\})+\$(\{394\}\{\text{ab}\{395\}\})+\$(\{396\}\{\text{ab}\{397\}\})+\$(\{398\}\{\text{ab}\{399\}\})+\$(\{400\}\{\text{ab}\{401\}\})+\$(\{402\}\{\text{ab}\{403\}\})+\$(\{404\}\{\text{ab}\{405\}\})+\$(\{406\}\{\text{ab}\{407\}\})+\$(\{408\}\{\text{ab}\{409\}\})+\$(\{410\}\{\text{ab}\{411\}\})+\$(\{412\}\{\text{ab}\{413\}\})+\$(\{414\}\{\text{ab}\{415\}\})+\$(\{416\}\{\text{ab}\{417\}\})+\$(\{418\}\{\text{ab}\{419\}\})+\$(\{420\}\{\text{ab}\{421\}\})+\$(\{422\}\{\text{ab}\{423\}\})+\$(\{424\}\{\text{ab}\{425\}\})+\$(\{426\}\{\text{ab}\{427\}\})+\$(\{428\}\{\text{ab}\{429\}\})+\$(\{430\}\{\text{ab}\{431\}\})+\$(\{432\}\{\text{ab}\{433\}\})+\$(\{434\}\{\text{ab}\{435\}\})+\$(\{436\}\{\text{ab}\{437\}\})+\$(\{438\}\{\text{ab}\{439\}\})+\$(\{440\}\{\text{ab}\{441\}\})+\$(\{442\}\{\text{ab}\{443\}\})+\$(\{444\}\{\text{ab}\{445\}\})+\$(\{446\}\{\text{ab}\{447\}\})+\$(\{448\}\{\text{ab}\{449\}\})+\$(\{450\}\{\text{ab}\{451\}\})+\$(\{452\}\{\text{ab}\{453\}\})+\$(\{454\}\{\text{ab}\{455\}\})+\$(\{456\}\{\text{ab}\{457\}\})+\$(\{458\}\{\text{ab}\{459\}\})+\$(\{460\}\{\text{ab}\{461\}\})+\$(\{462\}\{\text{ab}\{463\}\})+\$(\{464\}\{\text{ab}\{465\}\})+\$(\{466\}\{\text{ab}\{467\}\})+\$(\{468\}\{\text{ab}\{469\}\})+\$(\{470\}\{\text{ab}\{471\}\})+\$(\{472\}\{\text{ab}\{473\}\})+\$(\{474\}\{\text{ab}\{475\}\})+\$(\{476\}\{\text{ab}\{477\}\})+\$(\{478\}\{\text{ab}\{479\}\})+\$(\{480\}\{\text{ab}\{481\}\})+\$(\{482\}\{\text{ab}\{483\}\})+\$(\{484\}\{\text{ab}\{485\}\})+\$(\{486\}\{\text{ab}\{487\}\})+\$(\{488\}\{\text{ab}\{489\}\})+\$(\{490\}\{\text{ab}\{491\}\})+\$(\{492\}\{\text{ab}\{493\}\})+\$(\{494\}\{\text{ab}\{495\}\})+\$(\{496\}\{\text{ab}\{497\}\})+\$(\{498\}\{\text{ab}\{499\}\})+\$(\{500\}\{\text{ab}\{501\}\})+\$(\{502\}\{\text{ab}\{503\}\})+\$(\{504\}\{\text{ab}\{505\}\})+\$(\{506\}\{\text{ab}\{507\}\})+\$(\{508\}\{\text{ab}\{509\}\})+\$(\{510\}\{\text{ab}\{511\}\})+\$(\{512\}\{\text{ab}\{513\}\})+\$(\{514\}\{\text{ab}\{515\}\})+\$(\{516\}\{\text{ab}\{517\}\})+\$(\{518\}\{\text{ab}\{519\}\})+\$(\{520\}\{\text{ab}\{521\}\})+\$(\{522\}\{\text{ab}\{523\}\})+\$(\{524\}\{\text{ab}\{525\}\})+\$(\{526\}\{\text{ab}\{527\}\})+\$(\{528\}\{\text{ab}\{529\}\})+\$(\{530\}\{\text{ab}\{531\}\})+\$(\{532\}\{\text{ab}\{533\}\})+\$(\{534\}\{\text{ab}\{535\}\})+\$(\{536\}\{\text{ab}\{537\}\})+\$(\{538\}\{\text{ab}\{539\}\})+\$(\{540\}\{\text{ab}\{541\}\})+\$(\{542\}\{\text{ab}\{543\}\})+\$(\{544\}\{\text{ab}\{545\}\})+\$(\{546\}\{\text{ab}\{547\}\})+\$(\{548\}\{\text{ab}\{549\}\})+\$(\{550\}\{\text{ab}\{551\}\})+\$(\{552\}\{\text{ab}\{553\}\})+\$(\{554\}\{\text{ab}\{555\}\})+\$(\{556\}\{\text{ab}\{557\}\})+\$(\{558\}\{\text{ab}\{559\}\})+\$(\{560\}\{\text{ab}\{561\}\})+\$(\{562\}\{\text{ab}\{563\}\})+\$(\{564\}\{\text{ab}\{565\}\})+\$(\{566\}\{\text{ab}\{567\}\})+\$(\{568\}\{\text{ab}\{569\}\})+\$(\{570\}\{\text{ab}\{571\}\})+\$(\{572\}\{\text{ab}\{573\}\})+\$(\{574\}\{\text{ab}\{575\}\})+\$(\{576\}\{\text{ab}\{577\}\})+\$(\{578\}\{\text{ab}\{579\}\})+\$(\{580\}\{\text{ab}\{581\}\})+\$(\{582\}\{\text{ab}\{583\}\})+\$(\{584\}\{\text{ab}\{585\}\})+\$(\{586\}\{\text{ab}\{587\}\})+\$(\{588\}\{\text{ab}\{589\}\})+\$(\{590\}\{\text{ab}\{591\}\})+\$(\{592\}\{\text{ab}\{593\}\})+\$(\{594\}\{\text{ab}\{595\}\})+\$(\{596\}\{\text{ab}\{597\}\})+\$(\{598\}\{\text{ab}\{599\}\})+\$(\{600\}\{\text{ab}\{601\}\})+\$(\{602\}\{\text{ab}\{603\}\})+\$(\{604\}\{\text{ab}\{605\}\})+\$(\{606\}\{\text{ab}\{607\}\})+\$(\{608\}\{\text{ab}\{609\}\})+\$(\{610\}\{\text{ab}\{611\}\})+\$(\{612\}\{\text{ab}\{613\}\})+\$(\{614\}\{\text{ab}\{615\}\})+\$(\{616\}\{\text{ab}\{617\}\})+\$(\{618\}\{\text{ab}\{619\}\})+\$(\{620\}\{\text{ab}\{621\}\})+\$(\{622\}\{\text{ab}\{623\}\})+\$(\{624\}\{\text{ab}\{625\}\})+\$(\{626\}\{\text{ab}\{627\}\})+\$(\{628\}\{\text{ab}\{629\}\})+\$(\{630\}\{\text{ab}\{631\}\})+\$(\{632\}\{\text{ab}\{633\}\})+\$(\{634\}\{\text{ab}\{635\}\})+\$(\{636\}\{\text{ab}\{637\}\})+\$(\{638\}\{\text{ab}\{639\}\})+\$(\{640\}\{\text{ab}\{641\}\})+\$(\{642\}\{\text{ab}\{643\}\})+\$(\{644\}\{\text{ab}\{645\}\})+\$(\{646\}\{\text{ab}\{647\}\})+\$(\{648\}\{\text{ab}\{649\}\})+\$(\{650\}\{\text{ab}\{651\}\})+\$(\{652\}\{\text{ab}\{653\}\})+\$(\{654\}\{\text{ab}\{655\}\})+\$(\{656\}\{\text{ab}\{657\}\})+\$(\{658\}\{\text{ab}\{659\}\})+\$(\{660\}\{\text{ab}\{661\}\})+\$(\{662\}\{\text{ab}\{663\}\})+\$(\{664\}\{\text{ab}\{665\}\})+\$(\{666\}\{\text{ab}\{667\}\})+\$(\{668\}\{\text{ab}\{669\}\})+\$(\{670\}\{\text{ab}\{671\}\})+\$(\{672\}\{\text{ab}\{673\}\})+\$(\{674\}\{\text{ab}\{675\}\})+\$(\{676\}\{\text{ab}\{677\}\})+\$(\{678\}\{\text{ab}\{679\}\})+\$(\{680\}\{\text{ab}\{681\}\})+\$(\{682\}\{\text{ab}\{683\}\})+\$(\{684\}\{\text{ab}\{685\}\})+\$(\{686\}\{\text{ab}\{687\}\})+\$(\{688\}\{\text{ab}\{689\}\})+\$(\{690\}\{\text{ab}\{691\}\})+\$(\{692\}\{\text{ab}\{693\}\})+\$(\{694\}\{\text{ab}\{695\}\})+\$(\{696\}\{\text{ab}\{697\}\})+\$(\{698\}\{\text{ab}\{699\}\})+\$(\{700\}\{\text{ab}\{701\}\})+\$(\{702\}\{\text{ab}\{703\}\})+\$(\{704\}\{\text{ab}\{705\}\})+\$(\{706\}\{\text{ab}\{707\}\})+\$(\{708\}\{\text{ab}\{709\}\})+\$(\{710\}\{\text{ab}\{711\}\})+\$(\{712\}\{\text{ab}\{713\}\})+\$(\{714\}\{\text{ab}\{715\}\})+\$(\{716\}\{\text{ab}\{717\}\})+\$(\{718\}\{\text{ab}\{719\}\})+\$(\{720\}\{\text{ab}\{721\}\})+\$(\{722\}\{\text{ab}\{723\}\})+\$(\{724\}\{\text{ab}\{725\}\})+\$(\{726\}\{\text{ab}\{727\}\})+\$(\{728\}\{\text{ab}\{729\}\})+\$(\{730\}\{\text{ab}\{731\}\})+\$(\{732\}\{\text{ab}\{733\}\})+\$(\{734\}\{\text{ab}\{735\}\})+\$(\{736\}\{\text{ab}\{737\}\})+\$(\{738\}\{\text{ab}\{739\}\})+\$(\{740\}\{\text{ab}\{741\}\})+\$(\{742\}\{\text{ab}\{743\}\})+\$(\{744\}\{\text{ab}\{745\}\})+\$(\{746\}\{\text{ab}\{747\}\})+\$(\{748\}\{\text{ab}\{749\}\})+\$(\{750\}\{\text{ab}\{751\}\})+\$(\{752\}\{\text{ab}\{753\}\})+\$(\{754\}\{\text{ab}\{755\}\})+\$(\{756\}\{\text{ab}\{757\}\})+\$(\{758\}\{\text{ab}\{759\}\})+\$(\{760\}\{\text{ab}\{761\}\})+\$(\{762\}\{\text{ab}\{763\}\})+\$(\{764\}\{\text{ab}\{765\}\})+\$(\{766\}\{\text{ab}\{767\}\})+\$(\{768\}\{\text{ab}\{769\}\})+\$(\{770\}\{\text{ab}\{771\}\})+\$(\{772\}\{\text{ab}\{773\}\})+\$(\{774\}\{\text{ab}\{775\}\})+\$(\{776\}\{\text{ab}\{777\}\})+\$(\{778\}\{\text{ab}\{779\}\})+\$(\{780\}\{\text{ab}\{781\}\})+\$(\{782\}\{\text{ab}\{783\}\})+\$(\{784\}\{\text{ab}\{785\}\})+\$(\{786\}\{\text{ab}\{787\}\})+\$(\{788\}\{\text{ab}\{789\}\})+\$(\{790\}\{\text{ab}\{791\}\})+\$(\{792\}\{\text{ab}\{793\}\})+\$(\{794\}\{\text{ab}\{795\}\})+\$(\{796\}\{\text{ab}\{797\}\})+\$(\{798\}\{\text{ab}\{799\}\})+\$(\{800\}\{\text{ab}\{801\}\})+\$(\{802\}\{\text{ab}\{803\}\})+\$(\{804\}\{\text{ab}\{805\}\})+\$(\{806\}\{\text{ab}\{807\}\})+\$(\{808\}\{\text{ab}\{809\}\})+\$(\{810\}\{\text{ab}\{811\}\})+\$(\{812\}\{\text{ab}\{813\}\})+\$(\{814\}\{\text{ab}\{815\}\})+\$(\{816\}\{\text{ab}\{817\}\})+\$(\{818\}\{\text{ab}\{819\}\})+\$(\{820\}\{\text{ab}\{821\}\})+\$(\{822\}\{\text{ab}\{823\}\})+\$(\{824\}\{\text{ab}\{825\}\})+\$(\{826\}\{\text{ab}\{827\}\})+\$(\{828\}\{\text{ab}\{829\}\})+\$(\{830\}\{\text{ab}\{831\}\})+\$(\{832\}\{\text{ab}\{833\}\})+\$(\{834\}\{\text{ab}\{835\}\})+\$(\{836\}\{\text{ab}\{837\}\})+\$(\{838\}\{\text{ab}\{839\}\})+\$(\{840\}\{\text{ab}\{841\}\})+\$(\{842\}\{\text{ab}\{843\}\})+\$(\{844\}\{\text{ab}\{845\}\})+\$(\{846\}\{\text{ab}\{847\}\})+\$(\{848\}\{\text{ab}\{849\}\})+\$(\{850\}\{\text{ab}\{851\}\})+\$(\{852\}\{\text{ab}\{853\}\})+\$(\{854\}\{\text{ab}\{855\}\})+\$(\{856\}\{\text{ab}\{857\}\})+\$(\{858\}\{\text{ab}\{859\}\})+\$(\{860\}\{\text{ab}\{861\}\})+\$(\{862\}\{\text{ab}\{863\}\})+\$(\{864\}\{\text{ab}\{865\}\})+\$(\{866\}\{\text{ab}\{867\}\})+\$(\{868\}\{\text{ab}\{869\}\})+\$(\{870\}\{\text{ab}\{871\}\})+\$(\{872\}\{\text{ab}\{873\}\})+\$(\{874\}\{\text{ab}\{875\}\})+\$(\{876\}\{\text{ab}\{877\}\})+\$(\{878\}\{\text{ab}\{879\}\})+\$(\{880\}\{\text{ab}\{881\}\})+\$(\{882\}\{\text{ab}\{883\}\})+\$(\{884\}\{\text{ab}\{885\}\})+\$(\{886\}\{\text{ab}\{887\}\})+\$(\{888\}\{\text{ab}\{889\}\})+\$(\{890\}\{\text{ab}\{891\}\})+\$(\{892\}\{\text{ab}\{893\}\})+\$(\{894\}\{\text{ab}\{895\}\})+\$(\{896\}\{\text{ab}\{897\}\})+\$(\{898\}\{\text{ab}\{899\}\})+\$(\{900\}\{\text{ab}\{901\}\})+\$(\{902\}\{\text{ab}\{903\}\})+\$(\{904\}\{\text{ab}\{905\}\})+\$(\{906\}\{\text{ab}\{907\}\})+\$(\{908\}\{\text{ab}\{909\}\})+\$(\{910\}\{\text{ab}\{911\}\})+\$(\{912\}\{\text{ab}\{913\}\})+\$(\{914\}\{\text{ab}\{915\}\})+\$(\{916\}\{\text{ab}\{917\}\})+\$(\{918\}\{\text{ab}\{919\}\})+\$(\{920\}\{\text{ab}\{921\}\})+\$(\{922\}\{\text{ab}\{923\}\})+\$(\{924\}\{\text{ab}\{925\}\})+\$(\{926\}\{\text{ab}\{927\}\})+\$(\{928\}\{\text{ab}\{929\}\})+\$(\{930\}\{\text{ab}\{931\}\})+\$(\{932\}\{\text{ab}\{933\}\})+\$(\{934\}\{\text{ab}\{935\}\})+\$(\{936\}\{\text{ab}\{937\}\})+\$(\{938\}\{\text{ab}\{939\}\})+\$(\{940\}\{\text{ab}\{941\}\})+\$(\{942\}\{\text{ab}\{943\}\})+\$(\{944\}\{\text{ab}\{945\}\})+\$(\{946\}\{\text{ab}\{947\}\})+\$(\{948\}\{\text{ab}\{949\}\})+\$(\{950\}\{\text{ab}\{951\}\})+\$(\{952\}\{\text{ab}\{953\}\})+\$(\{954\}\{\text{ab}\{955\}\})+\$(\{956\}\{\text{ab}\{957\}\})+\$(\{958\}\{\text{ab}\{959\}\})+\$(\{960\}\{\text{ab}\{961\}\})+\$(\{962\}\{\text{ab}\{963\}\})+\$(\{964\}\{\text{ab}\{965\}\})+\$(\{966\}\{\text{ab}\{967\}\})+\$(\{968\}\{\text{ab}\{969\}\})+\$(\{970\}\{\text{ab}\{971\}\})+\$(\{972\}\{\text{ab}\{973\}\})+\$(\{974\}\{\text{ab}\{975\}\})+\$(\{976\}\{\text{ab}\{977\}\})+\$(\{978\}\{\text{ab}\{979\}\})+\$(\{980\}\{\text{ab}\{981\}\})+\$(\{982\}\{\text{ab}\{983\}\})+\$(\{984\}\{\text{ab}\{985\}\})+\$(\{986\}\{\text{ab}\{987\}\})+\$(\{988\}\{\text{ab}\{989\}\})+\$(\{990\}\{\text{ab}\{991\}\})+\$(\{992\}\{\text{ab}\{993\}\})+\$(\{994\}\{\text{ab}\{995\}\})+\$(\{996\}\{\text{ab}\{997\}\})+\$(\{998\}\{\text{ab}\{999\}\})+\$(\{1000\}\{\text{ab}\{1001\}\})+\$(\{1002\}\{\text{ab}\{1003\}\})+\$(\{1004\}\{\text{ab}\{1005\}\})+\$(\{1006\}\{\text{ab}\{1007\}\})+\$(\{1008\}\{\text{ab}\{1009\}\})+\$(\{1010\}\{\text{ab}\{1011\}\})+\$(\{1012\}\{\text{ab}\{1013\}\})+\$(\{1014\}\{\text{ab}\{1015\}\})+\$(\{1016\}\{\text{ab}\{1017\}\})+\$(\{1018\}\{\text{ab}\{1019\}\})+\$(\{1020\}\{\text{ab}\{1021\}\})+\$(\{1022\}\{\text{ab}\{1023\}\})+\$(\{1024\}\{\text{ab}\{1025\}\})+\$(\{1026\}\{\text{ab}\{1027\}\})+\$(\{1028\}\{\text{ab}\{1029\}\})+\$(\{1030\}\{\text{ab}\{1031\}\})+\$(\{1032\}\{\text{ab}\{1033\}\})+\$(\{1034\}\{\text{ab}\{1035\}\})+\$(\{1036\}\{\text{ab}\{1037\}\})+\$(\{1038\}\{\text{ab}\{1039\}\})+\$(\{1040\}\{\text{ab}\{1041\}\})+\$(\{1042\}\{\text{ab}\{1043\}\})+\$(\{1044\}\{\text{ab}\{1045\}\})+\$(\{1046\}\{\text{ab}\{1047\}\})+\$(\{1048\}\{\text{ab}\{1049\}\})+\$(\{1050\}\{\text{ab}\{1051\}\})+\$(\{1052\}\{\text{ab}\{1053\}\})+\$(\{1054\}\{\text{ab}\{1055\}\})+\$(\{1056\}\{\text{ab}\{1057\}\})+\$(\{1058\}\{\text{ab}\{1059\}\})+\$(\{1060\}\{\text{ab}\{1061\}\})+\$(\{1062\}\{\text{ab}\{1063\}\})+\$(\{1064\}\{\text{ab}\{1065\}\})+\$(\{1066\}\{\text{ab}\{1067\}\})+\$(\{1068\}\{\text{ab}\{1069\}\})+\$(\{1070\}\{\text{ab}\{1071\}\})+\$(\{1072\}\{\text{ab}\{1073\}\})+\$(\{1074\}\{\text{ab}\{1075\}\})+\$(\{1076\}\{\text{ab}\{1077\}\})+\$(\{107$$

**TUGboat wish list**

These are some of the topics on which the editor is looking for authors. Add your own suggestions or volunteers! Send e-mail to [TUGboat@math.ams.org](mailto:TUGboat@math.ams.org) with details.

- interviews with people who have influenced  $\TeX$  and TUG
- real product reviews of both commercial and PD  $\TeX$  implementations and other software, also macro packages like *petricks*, etc.
- surveys of  $\TeX$  implementations for particular hardware/operating system combinations, with comparisons of features
- “road map” to the CTAN  $\TeX$  areas
- more tutorials and expository material, particularly for new users and users who aren’t intending to become  $\TeX$  wizards; one possibility — answers to the “top ten” questions sent to `comp.text.tex` by people writing dissertations
- “how to” articles — how to build your own style based on, say, `article.sty`, how to include an abstract and other stuff in the full-width block at the top of a two-column article, etc.
- comparative analyses of style files that address the same problem, e.g., crop marks
- crossword puzzles for the whole  $\TeX$  community

columnists of the latter variety may be promoted to associate editor (see the list on the reverse of the title page of this issue). If you are interested in either track, a message to TUGboat would be welcomed.

- **Production assistance.** This is a more problematic area, as the successful production of an issue of TUGboat requires that every file and every font be available to and compatible with the equipment on which the camera copy is generated. However, sometimes it helps to have someone to call on to generate fonts, vet macro files (I always assume that if the author doesn’t specify otherwise, the current version on CTAN will work properly, an assumption that isn’t always warranted), and help fight other fires. If you’re an experienced (B) $\TeX$  user and are interested in this sort of challenge, send a message to the TUGboat address with the details of the system you’re working on — computer, operating system, implementation and version of  $\TeX$  and METAFONT, output device(s) available. Previous production experience is a big plus, and a direct Internet connection is necessary.

By now, you’ve seen Christina’s solicitations for a new  $\TeX$  and TUG News editor. The editor of TUGboat has been having similar thoughts off and on for several years, but hasn’t done anything so

far about it. After the nearly disastrous failures to meet the publication schedule this past year, it’s imperative that I do start looking toward the future. I know that TUGboat edited by someone else wouldn’t be quite the same, but there are many valid conceptions of what such a journal should be. The criteria that I’d value in a possible successor include, in no particular order:

- broad and thorough knowledge of  $\TeX$  and its relations;
  - fascination with the typographic art and a desire always to learn more;
  - literacy;
  - a good (native) command of English and some ability to understand other human languages;
  - tact;
  - a comfortable familiarity with the electronic networks;
  - the ability to bend a computer to one’s will;
    - a well-developed sense of responsibility.
- If you think you might be such a person, or know of someone else who is, please contact me directly: [bb@math.ams.org](mailto:bb@math.ams.org).

◊ Barbara Beeton  
 American Mathematical Society  
 P.O. Box 6248  
 Providence, RI 02940 USA  
[bb@math.ams.org](mailto:bb@math.ams.org)

**Figure 53:** TUGboat 14:4 — my wish list, and my list of qualifications for a future TUGboat editor

work with editing and production, especially since Mimi’s retirement. He’d make a fine editor, though perhaps he’d rather “have a life” outside of TUG. I’d like to see TUGboat continue as a publication for *all*  $\TeX$  users, and indeed for anyone interested in high-quality typography and composition.

That brings up a matter that has bothered me for a while. The bulk of TUGboat is still produced with  $\LaTeX$ , and much of the content is also biased in that direction. One effect is the downplaying of plain  $\TeX$ , which still has its devoted users; a sad consequence is that at least one member of long standing has resigned, citing the  $\LaTeX$  bias as the reason. Remember — *all*  $\TeX$  users. Let’s not neglect our old friends, or take them entirely for granted.

A very long time ago, I published a “wish list” (Fig. 53). Rereading it now, I wouldn’t change much, nor would I change the list of qualifications I thought would be good in my successor as editor.

What will I do when I retire? Well, I hope not

#### Topics in a TUGboat Index

Barbara Beeton and Ron Whitney

Index-construction is a notoriously difficult task. One can approach an index from within (looking at the existing text and classifying what one sees) or from without (imagining what words and concepts users of the index will naturally want to use for queries). Of course, the whole process will comprise both these methods with additional cross-references and consistency checks.

One of the authors (BB) has spent considerable time examining TUGboat issues 7–10, marking items for a cumulative TUGboat index. The

following list contains the headings generated from these volumes for a *subject* index. Items listed within square brackets (as *Notes*) are for production purposes only and will not appear in the final copy. We would value any help you care to give! Please add to the list any subject areas you might naturally query in a TUGboat Subject Index which are not already present. Annotations and other comments are also quite welcome. The white-space you see is there for your benefit. Please use it!

Accents, *see* Diacritics

$\LaTeX$ - $\TeX$

Announcements, miscellaneous

Applications, *see* Back-end formatter; Book production; Database applications; Electronic publishing; General applications; Journal production;

Archives of  $\TeX$  material

ASCII output from  $\TeX$  or  $\LaTeX$

Back-end formatter

Beginner’s topics (*see also* Training; Tutorials)

Bibliographic tools

Book production

Books on  $\TeX$ , *see* Publications on  $\TeX$

Bridge

Budgets, TUG, *see* Financial reports; Treasurer’s reports

Calendar of events [Note: regular feature]

Chemical notation

Chess

Commercial use of  $\TeX$ , *see* Production use of  $\TeX$

preliminary draft, 13 Jun 1990 18:15

preliminary draft, 13 Jun 1990 18:15

**Figure 54:** The bare bones of a potential TUGboat index

to lose touch with either TUG or  $\TeX$ . If I just hang around the house, it will simply drive my usually patient husband to distraction. A project I started long ago might be revived: a TUGboat index. I’ve already accumulated the data for volumes 1–10. Organizing this needs a method of sorting (and printing) locations that handles volume and issue as well as page number. A basic outline for cross-references already exists — a draft was circulated for comments at the 1990 meeting at Texas A & M in College Station (Fig. 54). No promises, but this seems a worthy project, and at least it would keep me off the streets.

## Thanks

And finally, I’d like to thank all TUG members and  $\TeX$  users, many of whom have become good friends through the years, for their support and encouragement. The Math Society has been a good place to work and be involved in this  $\TeX$  enterprise. And most of all, Don Knuth, who started it all.

# A lifetime as an amateur compositor

David Walden

*The first section of this paper briefly relates my experience writing and printing documents until I began to use T<sub>E</sub>X. The second section summarizes why I now use T<sub>E</sub>X and gives examples of its benefits, particularly writing books. Section 3 touches on the advantage of being able to use a separate powerful text editor, since T<sub>E</sub>X does not require use of a built in editor. Go straight to section 2 if you want skip my reminiscences that are not directly related to T<sub>E</sub>X.*

## 1 First fifty years

### 1.1 Pre-computers

For some reason, I have always been interested in putting marks on paper — as with many people, my first work was with crayons, finger paints, and 1 inch, ruled “primary paper” and thick “primary pencils.”<sup>1</sup> But it was not long before I moved on to more publication-like processes. Our church had a mimeograph machine and my parents were involved with producing the Sunday programs, and my parents both taught in the public schools<sup>2</sup> where they prepared handouts to students using ditto machines. I was a little involved with reproducing such materials at least throughout my teen years.

My father had an Underwood manual typewriter upon which he typed and on which I banged with a few fingers as a child. Later, he obtained a Royal manual typewriter on which I typed with ten fingers from the time I took a typing class two hours a week for one term during my sophomore year of high school. Ever since taking that typing class, I have been typing, often for reproduction, more or less constantly: so much so that when my son was a child and people asked him what his father did, my son’s answer was, “He is a typist.” After I went away to college, I moved to a Smith Corona electric portable typewriter; and when I entered the work world, we used IBM Selectric typewriters with their changeable type balls.

However, I wasn’t a mistake-free typist, and I had much use for the tools of typewriter correction using carbon paper and other typewriter-base media:

<sup>1</sup> The oral presentation version of this paper given at the PracT<sub>E</sub>X’06 conference included a number of photographs that are not included here because I did not seek permission to use them.

<sup>2</sup> Supported by the town government.

erasers, stuff to patch a mimeograph stencil, a razor blade to scrape the ink off of a ditto master, and KO-REC-TYPE paper and Snopake correction fluid to paint over typing so characters could be retyped correctly on pages that would be reproduced on Xerox copiers.

At <http://www.tpub-products.com/>, I found a document for sale that describes the duties of a military “religious program (RP) specialist” (an assistant or secretary to a chaplain), and it includes instructions for using Ditto masters; I quote it below. This description represents about the mid-level of complexity of pre-computer desktop “publishing” — more complicated than carbon paper (but not much) and slightly less complicated than a stencil machine.

Before proceeding to an explanation of stencil preparation, the Ditto master will be discussed. The white Ditto master (overlay) is attached to a sheet of paper which is thickly coated with a carbon substance. Typing and hand- stylus impressions are made on the overlay and cause the carbon substance to be imprinted on the reverse side of the master. When the overlay is attached to the Ditto machine, the carbon-coated sheet is detached. The carbon impressions of the Ditto master are moistened by the duplicating fluid as the drum is rotated, which in turn transfers the carbon dye to the paper being fed into the machine. This transfer yields an exact reproduction of the master.

Preparing a neat and accurate Ditto master stencil is one of the more important secretarial tasks that the RP will perform. Command Religious Program announcements are often distributed to command personnel through the use of Ditto copies. Just as the appearance of the office of the chaplain makes an instant and lasting impression, an information “flyer” or announcement will also leave lasting impressions. If the announcement is neatly prepared with concise and accurate information, it will probably give people the impression that the office of the chaplain is an efficient and caring organization. Therefore, it is important that the RP prepare each Ditto master with these thoughts in mind. The following helpful hints should aid the RP in preparing Ditto masters:

- The “flimsy” sheet of paper that is inserted between the Ditto overlay and the carbon attachment MUST be removed before it is possible to

have impressions transferred to the back of the overlay. NOTE: If there is some art-work involved, the “flimsy” may be left between the overlay and carbon attachment while the art-work is penciled lightly onto the overlay. The artwork can then be retraced with a stylus when the “flimsy” is removed. If an electric typewriter is being used, a test line should be typed on a Ditto master at each typing pressure setting. A copy should then be run and the RP can select the pressure that will provide the best copy. For manual typewriters, the typing pressure lever should be set to a medium or light position for best results.

- A Ditto master should be left in the typewriter when errors are corrected. The typewriter platen should be turned until there is enough room to separate the perforated overlay from the carbon backing. A razor blade or other sharp-edged instrument should then be used to lightly scrape the carbon deposit of the incorrect characters from the back of the overlay. Next, a clean piece of Ditto carbon should be placed between the overlay and the original carbon. Then the typewriter should be returned to its original position and the correct letters typed. After the correction has been made, the temporary carbon that was used for this correction MUST be removed before proceeding.

- Ditto masters may be reused at a later date if they are properly stored after the initial use. The masters should be placed in large envelopes and separated by flimsy sheets. It is imperative that they be stored in a flat position to keep them from becoming wrinkled

The point I am trying to make with the above discussion about the pre-computer era is that it took many (fussy, touchy, tedious) steps of careful work to produce good output, just as it does today in the world of fancy computer-based systems. Added problems were that the “desktop” (versus professional printing) approaches to reproducible typesetting in the pre-computer era didn’t produce high quality printing, and there were limits on the number of copies you could make before the masters wore out.

## 1.2 Early computer use

I first came in contact with computers when I was in my junior year in college. While I still continued to use a typewriter for the next decade or so, I was also gradually changing over to using computers for typing documents, especially those that would be reproduced. I started with punch cards and an IBM 025 key punch machine, moved to rolls of punched paper tape with editing using Dan Murphy’s TECO (tape editor and corrector), continued using TECO (modified to work with computer files

rather than paper tape) via a Model 33 Teletype and then a TI Silent 700 as I moved into the world of computer time sharing (where the computer terminal was in my own office for the first time), used Jeremy Salter’s RUNOFF (the first word processing program) on MIT’s CTSS system, MRUNOFF (a version of RUNOFF for the TENEX operating system), and briefly touched troff/nroff in the early years of Unix. This computer-based world allow editing (e.g., with TECO) and reprinting of the actual raw text of a document or, eventually, inclusion of typesetting commands that would be interpreted by RUNOFF, MRUNOFF, and troff/nroff to produce the final document which could then be reproduced.

In the mid- to late-1970s, I first used a personal computer — an Apple II — but only to run VisiCalc. I was still doing word processing using MRUNOFF on TENEX. In 1981, IBM announced its PC and I got one for the following Christmas, I believe. My wife began using WordStar, and I helped her because I was familiar with command-based word processing from MRUNOFF which my friend Rob Barnaby (developer of WordStar) had also used.

## 1.3 Word and WYSIWYG

I don’t remember when, but before too long (perhaps on the first PC AT) I began using the WYSIWYG PC-Word for DOS (based on the ideas of Charles Simonyi). Then I converted to using the Mac and MacWord which seemed to be where the forefront of Word development was taking place. MacWord was somewhat incompatible with PC-Word, but my PC-Word files converted over to the MacWord pretty well, although my memory is that the very straightforward style sheets of Word for DOS were no longer quite so straightforward with MacWord and I couldn’t find some other features I had been using with PC-Word. I used MacWord for about eight years. In the early 1990s I decided to convert back to using an IBM PC using a Windows-based DOS operating system and then Win 3.1, but I discovered that my original Word files for the early PC that had been converted to MacWord did not convert back to the later versions of PC-Word very well. This was quite distressing to me. Moreover, after each of these changes I could not find various capabilities I was used to using — they were perhaps still there but apparently had moved or how they were executed had changed.

As time went by and I continued to use Word as part of Microsoft’s Office Suite, I became increasingly annoyed at Word. Bigger, more complicated releases kept coming out, and in time there was pressure from people with whom I exchanged Word files to upgrade to the latest version because earlier versions couldn’t

easily handle files from later versions without the person using the later version explicitly saving the files in the format of the earlier version, something many Word users didn't even know how to do. Also, each new release tended to again change how one called for various capabilities to be executed, while in time Microsoft stopped shipping hard copy manuals with Word from which one could learn such things (Microsoft increasingly forced users to depend on on-line documentation which doesn't work so well when you don't know how to ask for what you want to know about). Also, each new release tended to try to do more things automatically for me, and it took more and more work to turn off all the "help" it was trying to provide to me—help that in many cases actually made things harder for me (while it didn't help me by providing powerful editing functions, e.g., using regular expressions).

#### 1.4 Breaking with Word; choosing TeX

My level of annoyance and frustration grew and grew, and eventually I made the decision to stop using Word for significant writing projects and to seek an alternative. Before I go on about my alternate approach, I must emphasize that I still use Word regularly for short, one-off projects (e.g., a short letter that I will not need to access on-line at a significantly later time) and when I work with someone who uses Word for his or her document preparation.

I chose to use L<sup>A</sup>T<sub>E</sub>X as my alternative to using Word for document preparation for several reasons:

- It had a visible, non-proprietary, documented markup with a simple, plain text syntax that I was confident would allow me to reuse text in different documents over the years.<sup>3</sup>
- I was already familiar with command-based word processing and (as a computer programmer at heart) liked what I know about TeX's programmability. I also welcomed the prospect of being able to use a powerful text editor again as part of my document editing process (more about this in section 3).
- I had been involved with religious arguments about which of PageMaker, FrameMaker, or Interleaf was the best tool in various situations; and, from what I knew then, they also had some of the same problems as Word in terms of

hidden markup and pressure on users to adopt new releases that obsoleted prior releases. I also was definitely looking for something that did not involve a graphical user interface (GUI)—something that required less mouse clicks. So, I didn't seriously investigate the just mentioned systems.

- I am a great admirer of Donald Knuth and thought it would be nice to try the system he developed.

Part of my preference for L<sup>A</sup>T<sub>E</sub>X over Word comes from the fact that all of the markup is in a file where I can see it and change it rather than it having to be accessed by various menus and being largely unseen (except in its effect) in the document. To take a simple example, suppose I wanted to make the word "brown" in the phrase "quick brown fox" be bold. In Word I would select "brown" with the mouse, pull down the Format menu, click the Font item on the menu, click Bold in the Font Style column, click OK, and then the text would appear in the document in bold when displayed or printed (alternatively I could type control-B after selecting the word "brown"). To do the same thing in L<sup>A</sup>T<sub>E</sub>X, I would change the text "quick brown fox" to "quick `\textbf{brown}` fox" with my text editor, and "brown" would display in bold when my L<sup>A</sup>T<sub>E</sub>X file was compiled.

No doubt there are ways in Word to do many if not all of the things I now do with L<sup>A</sup>T<sub>E</sub>X, but I find them mostly easier to find and do in L<sup>A</sup>T<sub>E</sub>X.

As an aside, another aspect of Word that annoys me is that it is forever guessing what I want. For instance, if I type an explicit new-line (Return key), Word may decide to capitalize the first word of the next line, which may or may not be right. When I select some text with the mouse in Word, it often chooses different text than I touch with the mouse, for instance an extra space. Much or all of this can probably be turned off and I turn off as much as I can, but I never seem to be able to turn off everything; and, while Word's "help" sometimes does result in what I want, it seems more often to choose what I don't want. L<sup>A</sup>T<sub>E</sub>X never seems to cause me this problem, which is not to say there are not other problems with L<sup>A</sup>T<sub>E</sub>X.

I don't remember what TeX distribution—I downloaded something from the Internet—I tried first using NotePad on the PC for my editing. I do remember buying *The TeXbook*, and then quickly discovering L<sup>A</sup>T<sub>E</sub>X which I experimented with a little bit. Then, I bought a copy of PCT<sub>E</sub>X on the theory that it would be nicely packaged, and I used it for a while but grew dissatisfied with the power of its editor. Then I found and downloaded WinEdt and MiK<sub>T</sub>E<sub>X</sub>.

<sup>3</sup> Word's hidden markup and WYSIWYG editing means that it is often hard to tell how something got to be the way it is. Also, since many Word users don't use style sheets, formatting (for instance, of a subsection title) might be done one way for one subsection and another way for another subsection, increasing the probability of inconsistencies in the output.

Later I bought and tried the Y&Y distribution, but I could never get it to work well; I did buy and make good use of the V $\TeX$  distribution for one particular project, but again I didn't like its editor. I ended up using WinEdt (and occasionally EMACS for things that seemed harder to do in WinEdt than in EMACS) and Mi $\TeX$  for a number of years, most recently obtained as part of TUG's Pro $\TeX$ t distribution.

## 2 Why I use L $\TeX$ , particularly for writing books

Two reasons typically given for using L $\TeX$  are for its math support and for very nice looking typesetting. Neither of these is particularly important to me: I rarely have any math in my writing (but it is nice to be able to handle it easily in those rare cases where I do have it); I have a pretty undiscerning eye when it comes to typesetting, and what L $\TeX$  produces is more than good enough for me.

Here's what matters most to me about L $\TeX$ :

1. its programmability and modularity;
2. that I get to use a powerful editor with it;
3. that the mark-up is clearly visible to me and can be changed directly with a text editor;
4. its capabilities for explicitly specifying cross-references, maintaining bibliographies, and automatically numbering chapters, sections, figures, tables, footnotes, etc., which permit easy reorganization of text within documents and reuse in other documents
5. its relatively slow pace of change and great concern among the developers for backwards compatibility.

In other words, my use of L $\TeX$  is primarily about productivity. (Of course, there are certain limitations on this productivity such as when I finish writing a book using L $\TeX$  and the publisher tells me I must convert the text to Word and the figures to PowerPoint slides for input into the compositor's typesetting system.)

Much of my work using L $\TeX$  is on book length documents. For these I have compiled a more or less standard set of techniques that I feel help me be more efficient. I don't claim that the techniques I use are the techniques of a master; in fact, I view myself as an intermediate user of L $\TeX$ —I know enough to make L $\TeX$  jump through a few simple hoops, but not enough to know if my approaches are recommended or if they include some bad habits.

In my experience, publishers don't think much about the design of a book until they have the completed manuscript in hand. Since I use L $\TeX$  to develop the original manuscript, I have to make lots

of temporary design decisions, and I want to be able to change these decisions with a minimum of work when the publisher does begin to deal with the design. Also, I am currently working on a book that I will be self-publishing, and settling on the design for this book is an iterative, experimental process where it is even more important to be able to make changes throughout the book (for instance, to the style of figure captions) with minimal work. My experience, however, should not prevent you from checking if the publisher of your document already has a standard style and perhaps even a L $\TeX$  class file that you can use from the outset of your writing. In any case, my emphasis here is *not* on the methods of representing preferences for appearance; my emphasis is on methods for easily and repeatedly *changing* the overall document appearance as well as on other methods for working efficiently on large documents.

Some of what I am about to describe for working efficiently on books or other long documents is probably already well known to many readers; perhaps you can make suggestions for how I might do things better.

(At several points in the following, I have included in parentheses discussions of basic  $\TeX$  and L $\TeX$  issues that reviewers and others who have read drafts of this paper have asked me about that are not actually on the subject of book-writing productivity. Perhaps these parenthetical notes should have been footnotes, but I was too lazy to deal with the need for alternatives to `\verb` in footnotes.)

### 2.1 Include files

Suppose I am working on a book entitled *Breakthrough Management*, as I have been recently. I created a top level file named `bt.tex` with the following contents:

```
\documentclass{btbook}
\begin{document}
\include{titlepages}
\include{preface}
\include{surviving}    % a chapter
\include{rapid}       % another chapter
. . .                  % more chapters
\include{acknowledgements}
\include{bibliography}
\include{bio}
\include{index}
\end{document}
```

The text from included files appears to L $\TeX$  as if it were in the file `bt.tex` in place of the `\include` commands. In this way, I contain the text related to each chapter and other parts of the book in its own file. I let L $\TeX$  take care of numbering the chapters

and figures (or whatever) within chapters. If I later decided to change the order of chapters, I just change the order of the `\include` commands in the `bt.tex` files, and L<sup>A</sup>T<sub>E</sub>X automatically renumbers everything.

To work on one chapter at a time, my file `bt.tex` evolved to include many `\includeonly` commands, e.g.,

```
\documentclass{btbook}
%\includeonly{preface}
%\includeonly{surviving}
\includeonly{rapid}
%\includeonly{surviving,rapid}
. . .
\begin{document}
\include{titlepages}
\include{preface}
\include{surviving} % a chapter
\include{rapid}    % another chapter
. . .              % more chapters
\include{acknowledgements}
\include{bibliography}
\include{bio}
\include{index}
\end{document}
```

In the above example, only the file `rapid.tex` gets compiled when I run L<sup>A</sup>T<sub>E</sub>X on the file `bt.tex`. In this 10-chapter book I had a couple of dozen `\includeonly` commands in the `bt.tex` file that I could comment in and out to work on each chapter individually and with various combinations of related chapters.

(Since the `\include` commands result in text being typeset, they must follow the `\begin{document}` command. The `\includeonly` commands must go in the preamble or else L<sup>A</sup>T<sub>E</sub>X complains.)

## 2.2 Custom class file

I have created a file `btbook.cls` which is my own personal class file for this particular book. This file is processed when L<sup>A</sup>T<sub>E</sub>X sees, at the beginning of the file `bt.tex`, the command `\documentclass{btbook}`. The first three lines of the file

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesClass{btbook}[2006/01/21 BTbookclass]
\LoadClass{book}
```

define the class for this book to be named ‘`btbook`’ and to be an augmentation of the L<sup>A</sup>T<sub>E</sub>X book class.

The rest of the lines of the file are read and executed when L<sup>A</sup>T<sub>E</sub>X is run as if they were lines of text immediately following the `\documentclass` command in the `bt.tex` file.

(If your publisher already provides a L<sup>A</sup>T<sub>E</sub>X class file, you can still collect all of the sorts of things I describe below in their own file and `\input` that file in the preamble rather than just putting all these things

directly in your preamble. I prefer not to have much in my preamble beyond the `\includeonly{...}` commands that I am constantly commenting in and out.)

## 2.3 Packages

Next in the class file comes the list of packages I use for writing this book.

```
% Palatino is basic roman font
\RequirePackage{mathpazo}
% Helvetica is sans serif font
\RequirePackage[scaled=.95]{helvet}
% Courier is typewriter font
\RequirePackage{courier}
% for including images
\RequirePackage{graphicx}
% for formatting URLs
\RequirePackage{url}
%to be able to rotate figures
\RequirePackage[figuresright]{rotating}
% for dropped caps
\RequirePackage{lettrine}
% for tighter list spacing
\RequirePackage{paralist}
\setlength{\pltopsep}{.05in}
% for comment environment
\RequirePackage{comment}
% for endnotes with reformatted numbers
\RequirePackage{dw-endnotes}
\RequirePackage{setspace} %\doublespacing
```

When I find I need to use another package, I add another `\RequirePackage` line to this list. (As I understand it, `\RequirePackage` does the same job as `\usepackage` except it doesn’t allow the same package to be loaded twice which apparently might cause problems in some cases.)

Notice that the package name in one case includes the characters `dw-`. This is my convention for noting a package that I have modified. In such cases, the file of the modified package is in the same directory with the rest of the files for this book or in the local changes part of my `texmf` data structure. I seldom understand a package I am modifying; I typically use a hit and miss approach to change stuff until I get the results I want.

Copy editors who edit on hard copy like double spacing, and I can provide that with a one character change—uncommenting the `\doublespacing` command on the last line above that loads the `setspace` package.

## 2.4 Miscellaneous useful macros

The following macros provide a few capabilities I use relatively frequently.

```

% space around em-dashes
\newcommand{\Dash}{\thinspace---\thinspace}
% mark text that needs checking
\newcommand{\CK}[1]{\textbf{CK #1}}
% marginal note to myself
\newcommand{\manote}[1]{%
  \marginpar{\scriptsize To do:\#1}}
% cut-in title
\newcommand{\partitle}[1]{%
  \medskip\noindent\textbf{#1}}
%change function of \url command
\let\Originalurl=\url
\def\url#1{\fontsize{10.7pt}{13.05pt}
  \Originalurl{#1}}

```

For some documents I have worked on, I have had many more such miscellaneous useful macros.

Anyone trying to improve productivity using L<sup>A</sup>T<sub>E</sub>X who doesn't already define his or her own macros should learn to do so. User-defined macros allow significant improvements in efficiency. For instance, the first macro above defines the command `\Dash{}` to be an abbreviation for the character string `\thinspace---\thinspace` which results in an em-dash being typeset with a *little* bit of space on each side of it, as in `aaa—bbb`. It is less characters and probably more reliable to type `\Dash{}` many times in a book than it is to type the characters `\thinspace---\thinspace{}` many times. In my view, however, the greater benefit of defining the `\Dash` command comes when my publisher tells me that its style is closed-form em-dashes (no space on either side, i.e., `aaa—bbb`) or a more open form (`aaa — bbb`). To implement either of these changes *throughout* the book, I merely redefine `\Dash`, e.g.,

```

% no spaces around em-dashes
\newcommand{\Dash}{---}
or
% full spaces around em-dashes
\newcommand{\Dash}{ --- }

```

and recompile my document. Containing such style conventions within a few lines of a large document and being able to change the style throughout the document with only a few key strokes is an enormous advantage. (I'll give a more complex example of such containment when I discuss macros for figures and tables below.)

To redefine a command that already exists in L<sup>A</sup>T<sub>E</sub>X or has been defined by a package that has already been loaded, for instance to define a variation on `\url` as I do in the last two lines of my group of miscellaneous useful macros, I have to use the `\renewcommand` command. The `\renewcommand` works just like `\newcommand` except that L<sup>A</sup>T<sub>E</sub>X does not complain with `\renewcommand` if I try to give

a definition to a command that already exists— a good thing to be warned about when one uses `\newcommand`. (In the next subsection I give another redefinition example— redefining `\footnote`.)

## 2.5 Footnotes and endnotes

In the case of `\RequirePackage{dw-endnotes}`, I am using the `endnotes` package, modified slightly to change the format of the note numbers.

Typically, I put footnotes on the bottom of text pages where they are referenced, at least while I am drafting chapters and want to be able to see the notes without having to turn a bunch of pages. However, publishers tend not to like having footnotes— it makes a book look too academic to be popular, in their view. Thus, before actual publication, I often find myself converting all my footnotes to endnotes. The next commands in my class file do this.

```

%comment out to not have end notes
\renewcommand{\footnote}{\endnote}

\newcommand{\dumpendnotes}{%
  \medskip
  \begingroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{1ex}%
    \renewcommand{\enotesize}{\normalsize}%
    \theendnotes
  \endgroup
  \setcounter{endnote}{0}}

```

First, the `\footnote` command is redefined to be the `\endnote` command; this avoids my having to replace every instance of `\footnote` with `\endnote`. Then the class file defines a command (`\dumpendnotes`) that can go at the end of each chapter to dump the chapter's endnotes, formatted as I want them to be. If the command `\dumpendnotes` was already defined in L<sup>A</sup>T<sub>E</sub>X or some other package, L<sup>A</sup>T<sub>E</sub>X would warn me because I didn't do the definition with `\renewcommand`.

## 2.6 Formatting figures and tables

The next set of commands in the class file have to do with changing the format of figure and table captions without actually modifying a L<sup>A</sup>T<sub>E</sub>X or package file. The L<sup>A</sup>T<sub>E</sub>X default does not use bold face for captions and uses a period rather than a hyphen between the chapter number and figure number within a chapter. The following changes patch L<sup>A</sup>T<sub>E</sub>X to follow my preference for bold face and hyphens.

```

\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{\textbf{#1}. \textbf{#2}}%
  \ifdim \wd\@tempboxa >\hsiz
    {\textbf{#1}. \textbf{#2}\par}

```

```

\else
  \global \@minipagefalse
  \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
\fi
\vskip\belowcaptionskip}
\renewcommand \thefigure
{\ifnum \c@chapter>\z@ \mbox{\thechapter-}%
\fi\@arabic\c@figure}
\renewcommand \thetable
{\ifnum \c@chapter>\z@ \mbox{\thechapter-}%
\fi\@arabic\c@table}

```

(I do not have to bracket these lines, top and bottom, with `\makeatletter` and `makeatother` commands as I would have to if this patch was in the preamble of my document; the at-sign is a letter by default in class files and packages. Some readers may be back a step, at the question of, “What is it about at-signs anyway?”. The answer is that the files for basic  $\LaTeX$ , for class files, and for other packages are full of macros names that include an at-sign (@), e.g., a macro named `\@makecaption` is defined at the beginning of the above example. My understanding is that an at-sign is used in low level programming of  $\LaTeX$ , class files, and packages to create macro names that can’t accidentally conflict with names that may be defined by users not doing such  $\LaTeX$  “systems programming.” An at-sign is normally not a letter and thus cannot be part of a macro name. However, in the above example I want to patch low level  $\LaTeX$  code that includes at-signs in its macro names; if I was trying to make this patch in my preamble (as I used to do before I learned to make some patches in a personal class file), I would have to tell  $\LaTeX$  to temporarily turn at-signs into letters, make the patch, and then turn them back into non letters (other) so the rest of my program could use @ in the normal way where it is not a special character of any kind.)

Perhaps there is a caption package that would allow such changes without patching  $\LaTeX$ , but I was shown how to make this patch a few years ago and it works, so why bother trying to find and learn a new package?

Next in my class file comes a set of definitions for commands I use to include graphics. I seldom insert `\begin{figure}` and `\end{figure}` commands directly into my documents; I do, from time to time, insert the commands `\begin{table}` and `\end{table}`. It is inevitable that, before I am done with a big document, I will want to change the formatting relating all figure and tables — perhaps several times. Thus, I use macros for inserting almost all figures (or tables) such that I can make changes to

formatting relating to the figures by making changes to only a few lines in the relevant macros.

```

%switch argument among pdf, eps, etc.
\newcommand{\figfiletype}{pdf}
%tell LaTeX directory path to figures
\graphicspath{{figures/}}
%commands to display file name, or not
\newcommand{\DFN}[2]{%
  \texttt{\small[#1 #2]}}
%\newcommand{\DFN}[2]{

\newcommand{\snfig}[3]{%scaled numbered figure
  %drop htb and %s for single page figures
  \begin{figure}[htbp]
%\vbox to \vsize{%
  \hfil\scalebox{#3}{
    \includegraphics{#2.\figfiletype}}\hfil
  \caption{\label{fig:#2}#1 \DFN{#2}{#3}}
%\vfil
%}
  \end{figure}
}

\newcommand{\sntab}[3]{%scaled numbered tables
  \begin{table}[thbp]
%\vbox to \vsize{%
  \centering
  \caption{\label{tab:#2}#1 \DFN{#2}{#3}}
  \smallskip
  \scalebox{#3}{
    \includegraphics{#2.\figfiletype}}
  \label{tab:#2}
%\vfil
%
  \end{table}
}

\newcommand{\unfig}[2]{%scaled unnumbered fig.
  \begin{figure}[htbp]
  \hfil\scalebox{#2}{
    \includegraphics{#1.\figfiletype}}\hfil
  \label{fig:#1}\centerline{
    \DFN{#1}{#2}}
  \end{figure}
}

%sideways scaled numbered figure
\newcommand{\swsnfig}[3]{
  \begin{sidewaysfigure}
  \centering
  \scalebox{#3}{
    \includegraphics{#2.\figfiletype}}
  \caption{\label{fig:#2}#1 \DFN{#2}{#3}}
  \end{sidewaysfigure}
}

```

For instance, the macro `\snfig` above takes three arguments. The text for a figure caption, the

unique part of the file name for the graphic to be included, and a scale factor for the graphic, e.g.,<sup>4</sup>

```
\snfig{This is the caption}{figure3-31}{.8}
```

The full name of the file to be included is the concatenation of the part of the file name that came from the second argument of the macro call, the directory that is specified by the `\graphicspath` command (an option of the `graphicx` package) as the place  $\LaTeX$  searches for figures, and the `\figfiletype` definition as the file name extension. The latter is useful because sometimes all of my figures are `.eps` files and sometimes they are `.pdf` files, and sometimes I switch between these two formats at different times in the production of the book. (When using `.eps` format, I compile using  $\LaTeX$  and a `dvi-to-pdf` conversion; when using `.pdf` format, I use `pdf $\TeX$`  to compile. If the graphic format was changing from file to file within the document, I would instead specify the format as another argument to the `\snfig` command. [However, Will Robertson and others have recently pointed out to me that if I leave the extension off, `\includegraphics` will pick the appropriate extension: `.eps` for  $\LaTeX$  and `.pdf` for `pdf $\TeX$` .]

While I am drafting and revising a book manuscript, I want to be able to look at a figure in the printed output and know what file I need to modify to change the figure. Thus, my macros for including figures and tables causes the file name to be included in the printed output in small letters enclosed in small square brackets, using the macro `\DFN`. When it comes time to create the final manuscript, I swap to a definition of `\DFN` that produces nothing and recompile the book's  $\LaTeX$  files.

The definitions of `\snfig` and `\sntab` also include several lines that are commented out. Professional editors often like to see the manuscript with figures or tables each on its own page rather than in-line with the text. Commenting in these few lines puts the figures and tables of the whole book on their own pages.

The `\snfig`, `\sntab`, and `\swsnfig` macros also define labels for cross-referencing the figures with `\ref` or `\pageref` commands. A slight limitation of my implementation is that I cannot reuse the same figure or table file without confusing the labeling.

<sup>4</sup> Barbara Beeton pointed out to me at the `Prac $\TeX$ '06` conference that `\includegraphics` can take a scale factor as an optional argument, and thus I don't need a separate `\scalebox` in the above definitions. I believe the `\scalebox` commands remain from before I switched from using the `graphics` package to using the `graphicx` package.

Also, I began using the `\hfil` commands in the above definitions before I understood I could use `\centering` as in the last definition.

However, it is easy enough to create a duplicate figure or table with a different file name.

I typically create all figures and most tables outside of  $\TeX$  itself and include them from separate files. If I found myself inserting very many tables directly into my `.tex` files rather than including them from graphics files, I would define a `mytable` environment so that I could still contain and simply change the sort of formatting I have discussed.

## 2.7 Thought breaks

The next group of commands (mostly commented out) are various options for indicating what I call “thought breaks” — places where formatting indicates a change of topic big enough to highlight but not big enough to have its own section or subsection title.<sup>5</sup> (These commands are defined with `\def` because I know they will pick up the correct arguments this way, and I am not sure enough of the details of how `\newcommand` works. I understand the details of how  $\TeX$  defines a macro and then collects its arguments when the macros are called because Knuth explains it pretty completely in *The  $\TeX$ book*. In particular,  $\TeX$  allows macro calls where the arguments of the macro are not all embedded in pairs of braces. However, I have never stumbled across a rigorous explanation of how a macro defined with `\newcommand` collects its arguments and thus in what situations arguments not in braces will be recognized or to what extent  $\LaTeX$  defined macros can have both of what Knuth calls delimited and undelimited arguments — and I have not bothered to study the  $\LaTeX$  code to figure it out. Consequently, out of ignorance, I use `\def` to define macros which don't have their arguments delimited by braces.)

```
\begin{comment}
\def\newthoughtgroup#1{\bgroup
  \afterassignment\BigFirstLetter \let\next=}
\def\BigFirstLetter#1{
  \bigskip\noindex{\Large#1}}

  %adapted slightly from Victor Eijkhout on ctt
\def\newthoughtgroup#1{\BigFirstLetter#1$}
\def\BigFirstLetter#1#2${
  \bigskip\noindent{\Large #1}#2}
\end{comment}

\def\newthoughtgroup#1{%
  \bigskip\noindent {\large #1}}

\begin{comment}
\def\newthoughtgroup{%
```

<sup>5</sup> See my *Prac $\TeX$  Journal* 2005-4 “Travels in  $\TeX$  Land” column (<http://www.tug.org/pracjourn/2005-4/walden/>) for examples of thought breaks.

```

\bigskip\noindent }

%big bold dropped cap letter with rest
% of word small caps
\def\newthoughtgroup#1#2 {
  \bigskip\noindent\lettrine{#1}{#2}\ }

\def\thoughtbreak{\vskip2pt
  \centerline{$^{\vrule width2cm height 1pt}$}
  \vskip2pt\noindent}
\end{comment}

```

The version of `\mythoughtgroup` currently not commented out indicates the new thought by a vertical space and a slightly bigger capital letter at the beginning of a non-indented paragraph.

## 2.8 Chapter formatting

The final set of commands in my class file has to do with the beginnings and ends of chapters. At the beginning and ending of each chapter I insert some commands that I can change either by changing the commands themselves or changing macros in the class file.<sup>6</sup>

```

\RequirePackage{fancyhdr}\pagestyle{fancyplain}
\newcommand{\mypartname}{}
\newcommand{\mychaptername}{}
\lhead[\fancyplain]{
  \thepage}\fancyplain{}{}}
\chead[\fancyplain]{
  \mypartname}\fancyplain{}\mychaptername}
\cfoot[\fancyplain{}]{
  \fancyplain{\thepage}}
\rhead[\fancyplain{}]{
  \fancyplain{}\thepage}}
\newcommand{\EMPTYPAGE}{\clearpage
  \thispagestyle{empty}\cleardoublepage}

\newcommand{\ENDCHAPTER}{\dumpendnotes}
\newcommand{\fENDCHAPTER}{\vfil\dumpendnotes}

```

In the class file for the book from which I drew these illustrations, there are a couple of alternative macros that I can include at the end of each chapter to dump the endnotes, but the end-of-chapter macros could be defined to cause other actions and outputs. In this book (which has only 10 chapters) I do not combine everything in a single beginning-of-chapter macro (e.g., `\BEGINCHAPTER`), but I have done this with some books (e.g., the 20-chapter book I am also currently working on). The typical beginning-

<sup>6</sup> When I first started customizing my page headings a few years ago, I used the `fancyheadings` package; recently I learned that the package `fancyhdr` has replaced `fancyheadings`, but I have not yet bothered to rewrite all the heading commands to use the new forms that come with the `fancyhdr` package and don't use the `fancyplain` device.

of-chapter commands for the chapter with the file name `rapid.tex` (mentioned earlier) are

```

\EMPTYPAGE
\chapter{Rapid Change in a Global World}
\label{ch:rapid}
\renewcommand{\mychaptername}{Chapter %
  \thechapter: Rapid Change in a Global World}

```

## 2.9 Using a fully developed class

For some books I have included different or additional capabilities in my custom class file.

Obviously, I could also use a fully developed class such as `memoir` rather than making lots of modifications of my own to I would still use some of the ideas I have described above. the  $\LaTeX$ 's standard book class. However, I suspect

It is clear that  $\TeX$  and its derivatives with their explicit, visible markup provide a strong base for incrementally building a personal library of techniques that are easy to apply from one project to the next.

## 3 Possible benefits of a separate editor

Using a word processor such as MS Word that has invisible, undocumented, proprietary markup means you have to use its built-in, WYSIWYG editor that knows about that markup. This has two potential disadvantages: (1) GUI-based editing often takes a lot more key strokes to do simple things than an editor like WinEdt or EMACS (I gave an example of this in section 1 and provide additional examples below); (2) an editor like MS Word's does not seem to have a lot of useful features that an editor like WinEdt or EMACS has.

Many of the ideas in this section are probably relevant to any typesetting system that has editing capability (like those I describe below). For all I know, MS Word can do many of these things; but, as I said in the first section, a number of years ago I lost interest in struggling to find stuff buried in all Word's menus, dealing with its ever changing user interface, and its planned-obsolescence-and-forced-upgrades business strategy.<sup>7</sup>

### 3.1 Two ways to make a change throughout a document

In the last section I sketched the benefits of using macros for some sequences of commands (for instance, `\Dash{}` for ---) that enable the replacement sequence to be changed everywhere in a document by

<sup>7</sup> The content of this section also appeared in a slightly different version in issue 2006-4 of *The PracTeX Journal* (<http://www.tug.org/pracjourn/2006-4/walden/>). That paper had additional content by Yuri Robbers on a number of the editors that work well with  $\TeX$ .

just changing the definition of the macro in one place. Another option for making a change to the same sequence of characters throughout a document is to use a text editor’s Replace All command. For instance, suppose I hadn’t used a macro for em-dashes and instead had closed form instances of --- throughout my document, e.g., “this is the end—the end of the line.” And then suppose I decide to change the style to uses semi-open form em-dashes, e.g., “this is the end — the end of the line.” With my editor I can do a Replace All of --- by `\thinspace---\thinspace{}`. If the document is broken up into separate files for each chapter, it will be good if the text editor has the option for doing the Replace All over all documents open in the editor instead of only in the document where the cursor currently is.

Here is another example of a simple text replacement of the entire document. Suppose I decide (for some reason) to replace all en-dashes by hyphens. Then I can do the following sequence of three steps (the first and third steps are to avoid accidentally changing instances of --- into --):

```
Replace All --- with #X#X#
Replace All -- with -
Replace All #X#X# with ---
```

Now suppose I want to add a fourth argument to every instance of the macro call `\snfig{ }{ }{ }` (see definition and discussion of this macro in subsection 2.6), that is, change the macro call formats to `\snfig{ }{ }{ }{ }`. Of course, one approach is to search for each instance of `\snfig{`, then move the cursor to after the third pair of braces, and then type the fourth pair of braces. However, if your text editor has a capability for dealing with regular expressions, you can make this change more easily (the last book I wrote had a couple of hundred instances of `\snfig`).

While I won’t get into the specific format of any particular editor’s representation of regular expressions, they would do something like the following in our example case.

```
Replace All \snfig{(.*)}{(.*)}{(.*)}
with \snfig{$1}{$2}{$3}{}
```

Everything in the Replace All command that is in a typewriter format is literal characters to be replaced. The characters in italics in the first part of the command are special characters that match any number of characters between balanced braces. For instance, in the command

```
\snfig{Caption title.}{file-name}{.5}
```

the first instance of `(.*)` would match the first argument `Caption title`. The second instance of

`(.*)` would match `file-name`, and the third instance would match `.5`. Better than that, the editor stores each matched set of characters in its own place for later reuse, as in the second half of the above command. The part of the command after the word “with” says to replace the `\snfig` command that was matched with all the same literal characters for the command names and braces, but to put the first match text in place of `$1`, the second match text in place of `$2`, and the third matched text in place of `$3`, and to add an extra pair of literal braces at the end of the replacement. Thus,

```
\snfig{Caption title.}{file-name}{.5}
is turned into
```

```
\snfig{Caption title.}{file-name}{.5}{}
and the same is done for every other instance of
\snfig, in each case properly maintaining the argu-
ment text through the replacement step.
```

The above example may need some tweaking if the instances of the command being changed sometimes span line boundaries, but typically this also can be handled, as can be much more complex instances of detecting what should be replaced and what should not be in various instances. In fact, depending on the editor’s particular regular expression capability, the earlier example of replacing en-dashes by hyphens perhaps could have been done with one Replace All using a regular expression to search of `-- not` followed by a third `-`.

In section two I recommended that anyone not already using  $\LaTeX$  macros should learn to use them. I recommend the same thing about using regular expressions if your editor supports them. They won’t be needed as often as macros, but when they are needed they are a major productivity increaser.

### 3.2 Other editor features

All of the serious text editors I have used allowed me to mark the cursor position with a couple of key strokes (e.g., Alt-F11 in WinEdt), move the cursor somewhere else (for instance to select some distant text and cut it, and then jump back to the first cursor position (e.g., Cntl-F11), where I might paste the text cut from elsewhere in the document.<sup>8</sup>

Text editors such as I have in mind also typically have provision to have multiple text buffers rather

---

<sup>8</sup> I don’t claim that none of these features are available in various editors that are packaged with commercial versions of  $\TeX$ ; I hope they are. I am only suggesting that you find an editor that supports such capabilities. What I do know is that with each new release of MS Word I find it harder and harder to find such features, if they exist at all, while they are easy to find in the two text editors I currently use regularly (WinEdt and EMACS).

than just one cut-and-paste clipboard.<sup>9</sup> I gave an example of this in the regular expression example in subsection 3.1, where three bits of text were simultaneously saved from the replaced string of characters for placement in the replacement string. With multiple places to save text, it is also possible, for example, to search-for-and-cut one bit of text, search-for-and-cut another bit of text, search-for-and-cut a third bit of text, and then paste together at some other point in the file, saving (in this example) several moves of the cursor in comparison with an editor with a single clipboard.

To give another example of the usefulness of multiple text buffers, I often find something on a web page and want to copy something from the page as a quote in a document I am writing *and* copy the URL as the source of the quote. Without multiple text buffers this requires the following sequence: (1) select text to be copied, (2) copy to clipboard, (3) switch window to other document, (4) position cursor, paste contents of clipboard, (5) switch window for first document, (6) select URL text, (7) copy to clipboard, (8) switch window to other document, (9) position cursor, (10) paste contents of clipboard.<sup>10</sup> With multiple text buffers it requires: (1) select text to be copied, (2) copy to buffer A, (3) select URL text, (4) copy to buffer B, (4) switch window to other document, (5) position cursor, (6) paste contents of buffer A, (7) reposition cursor, (8) paste contents of buffer B. The latter method is not necessarily less key strokes (the macros I have for WinEdt take 12 steps), but it is somehow easier for me not to switch windows and have to re-find my place as often.

The fact that L<sup>A</sup>T<sub>E</sub>X is not locked to a particular editor also means that each participant in a collaborative project can use the editor with which he or she is most familiar. (Collaboration is also made easier because there is much more compatibility from release to release of T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X, even with multiple providers, than there is from release to release with many non-T<sub>E</sub>X commercial products. For instance, MS Word seems to go out of its way to enforce inter-release incompatibility in a way apparently aimed at forcing all collaborators to all upgrade to the same release.)

Using a text editor in conjunction with L<sup>A</sup>T<sub>E</sub>X with its explicit markup also has advantages. For instance, it is easy to search for an italicized ver-

sion of a word (i.e., search for `\textit{word}`) as distinct from a non-italicized version of the same word. Similarly, it is possible to search for all headings of a certain level (for instance, all instances of `\subsection`); with a system with implicit markup (e.g., MS Word) one might have to search for the words of each subsection title.

In subsection 2.1 I showed the use of `\include` files. This works because L<sup>A</sup>T<sub>E</sub>X has the provision for specifying in one file a list of files to be included as if they were text in that first file. The editor I mostly use, WinEdt, also supports this; it knows enough about L<sup>A</sup>T<sub>E</sub>X to search the highest level file for instances of `\include` and gives me a list of visual tabs to the various files to be included; this makes it very easy for me to move among the various files in a longer document.

I could give an unlimited number of examples of what powerful text editors can do once one breaks free of the limitations of hidden, proprietary, undocumented markup and those built-in editors whose graphical user interfaces eliminate powerful editing capabilities as part of providing a point-and-click environment to the user. One of the best things is that I can use the same editor with which I have become facile from application to application. (Also, if several authors are collaborating, they can all use their own preferred editor, and — perhaps more importantly — they don't all have to have the same version of Word installed.)

## 4 Conclusion

To conclude, I give some additional opinions on a couple of thoughts I hinted at in the earlier sections and one additional opinion.<sup>11</sup>

### 4.1 Conservation of hassle

In my observations covering many decades, it has always taken many fussy steps to do anything involving typesetting for reproduction. The computer era has eliminated many physical steps, each of which required its own sort of skill and complexity. However, the computer era hasn't done anything to decrease the total number of steps — they are just done with a keyboard and mouse now and the skill is in knowing what commands can do what you need and where to find them. What computer-based approach is best is a matter of personal choice — they are all filled with hassle. My own choice has evolved to be a powerful, explicit typesetting language (L<sup>A</sup>T<sub>E</sub>X) combined with

<sup>9</sup> Alex Simonic, the developer of WinEdt, the editor I mostly use, showed me how to write macros to provide multiple text buffers in WinEdt.

<sup>10</sup> I know I could have two windows open at the same time, but sizing the windows so both can be seen takes too many steps unless I am going to do a lot of copying between two documents.

<sup>11</sup> For a related point of view to some of what I say here, see “L<sup>A</sup>T<sub>E</sub>X for Windows — A User's Perspective”, Proceedings of the 2001 Annual Meeting, *TUGboat*, Volume 22(2001), No. 3, pp. 140–145.

a powerful text editor. Some people argue that the WYSIWYG approach is easier. On average, however, I see the WYSIWYG approach as just as much work for the same task: A simple task in Word also tends to be simple in L<sup>A</sup>T<sub>E</sub>X; a more advanced task (e.g., inserting a cross-reference) seems more straightforward to do in L<sup>A</sup>T<sub>E</sub>X than in Word. As one uses more and more of L<sup>A</sup>T<sub>E</sub>X's power, the same task typically seems more and more difficult to do in Word as well, and my frustration level seems to grow faster with Word. Since most users use Word in only trivial ways, Word is pretty trivial to use at that level, but then so is L<sup>A</sup>T<sub>E</sub>X at that level.

#### 4.2 The assertion that L<sup>A</sup>T<sub>E</sub>X is hard to learn

I have no doubt that most people could learn to use L<sup>A</sup>T<sub>E</sub>X and a good text editor if they saw it as beneficial; L<sup>A</sup>T<sub>E</sub>X and a text editor at the intermediate level of sophistication at which I use them are no more complex than trying to use Word for the same task (and I mostly think they are less complex than Word). Think about all the other, fairly complex things people master in their lives — cooking, knitting, growing flowers, fly tying, and the rules of baseball. By comparison, there is nothing inherently too difficult about using L<sup>A</sup>T<sub>E</sub>X — it's only a question of learning enough to see its comparative benefits (and cheaper price). And anyone who can learn to use Word at a high level with all its particular weirdnesses (and changes to the user interface with each release) can surely also learn to use L<sup>A</sup>T<sub>E</sub>X at the same level.

#### 4.3 Using what everyone else is using

I believe the main argument against L<sup>A</sup>T<sub>E</sub>X and for Word is ubiquity of use. People use Word because everyone else does — their collaborators, their publishers, etc. — not because Word is better. If people were interested in a better word processor, they would use Word Perfect or perhaps one specialized to their area of writing such as Note Bene.

If the world of T<sub>E</sub>X is to have the best chance of snagging and keeping potential L<sup>A</sup>T<sub>E</sub>X users and users of other T<sub>E</sub>X-related system, we must continue to offer them a great, if small, community of fellow users. This is what volunteer-supported capabilities such as the T<sub>E</sub>X FAQ, CTAN, discussion groups such as `comp.text.tex` and `texhax`, TUG and the other volunteer-based user groups, the volunteer-created hardcopy and on-line journals, and meetings such as this PracT<sub>E</sub>X conference are about. It is a great pleasure for me to participate in this welcoming and informative community. Thank you.

#### Acknowledgements

I owe thanks to many sources for what I have learned about using L<sup>A</sup>T<sub>E</sub>X — books, the `comp.text.tex` list, the `texhax` list, and many individuals. Of course, none of them are responsible for lessons I have mislearned.

I can't remember and acknowledge everyone who directed me to techniques illustrated in this column; however, I can remember some of them. Karl Berry reviewed an early version of this paper and earlier told me about some of the methods I have described here. I also remember Peter Flynn, Steve Peter, and Steve Schwartz telling me about particular techniques. Peter Flom, Will Robertson, and anonymous reviewers provided many helpful suggestions for sections 2 and which appeared in earlier incarnations in issues 2006-2 and 2006-4 of *The PracT<sub>E</sub>X Journal*. Will Robertson also carefully reviewed the complete paper here and made many substantive suggestions for improvement as well as catching many minor errors.

#### Biographical note

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. More history is at [www.walden-family.com/dave](http://www.walden-family.com/dave).

# TeX and medicine

Elizabeth Dearborn

Buffalo, New York

elizabeth (at) medicalese dot org

<http://www.MeDiCaLeSe.org>

## 1 Introduction

Way back in 1985, when I worked at a struggling graphic design shop, I learned to set type on a Mergenthaler phototypesetting machine which was old even then. Remember width cards and photographic chemicals? In 1987, I went into medical transcription, which is an extremely knowledge-intensive line of work with very tight deadlines. I wanted to be good at what I did, so I maintained my own word list, which grew exponentially with time.

I was interested in computer programming and the Internet. I studied these things on my own, and in 2001 I started my website at <http://www.MeDiCaLeSe.org>. Using a JavaScript site search program I found on the Internet, I put my word list online in searchable form. Although the website and the database were available to everyone, my main objective at the time was to be able to search my own word list quickly. Whenever a person searched the database, my entire word list was loaded into a temporary file on the client's machine. This also required JavaScript to be enabled on the client side.

After a couple of years, the word list had gotten huge, the JavaScript search engine was taking a long time to load, and I was beginning to feel that the products of my research were worth money. I wanted to find a way to serve *only* the requested results and serve them up faster, and, if possible, I wanted to accomplish this without making my code visible to the end user.

## 2 The php programming language

I looked, but I could not find a free or inexpensive package that would do what I wanted. Hiring a programmer was out of the question, as was going back to school. I'm not the type who takes courses; everything I've ever wanted to know about, I've learned on my own, with some degree of success. I checked out the various programming languages which are used on the Internet, such as C++, Pascal, php, and Perl. Of these, php seemed best for my purposes. It is free, runs entirely on the server side, and virtually every web host offers it to its customers.

My approach to learning enough php to get the

job done was very similar to the way I later went about learning TeX. First, I defined exactly what I wanted to do. Next, I searched the official php website at <http://www.php.net> to see if a command existed that would accomplish this. If not, I would need to write my own. Sound familiar?

I used two books, both published by O'Reilly Media: *Programming PHP* by Rasmus Lerdorf and Kevin Tatroe, and *PHP Cookbook* by David Sklar and Adam Trachtenberg. Between these two wonderful books, the vast resources of the Internet, and my own determination, the improved site search was ready to go live in about two months' time, in the summer of 2003. The original version of my php site search had the Google-style highlighting as it does today, but did not then include the capability to exclude one string from the search results. I invite you to visit <http://www.MeDiCaLeSe.org> and try it out.

## 3 A database in plain text files

If you type the word "Jones" into the search box on the site, it will return 12 results, two of which are shown below:

Matches: jones: 12

**Orthopedics:** Jones fracture - fractured proximal fifth metatarsal.

**Diagnostics:** Jones silver stain.

If you view the page source, you won't see a listing of the php code that made the page, but you will be able to see the HTML code for the results. This is not very different from what the plain text entries in the database actually look like, which is:

```
<B>Orthopedics: </B>Jones fracture -  
fractured proximal fifth  
metatarsal.|06/04/04
```

```
...  
<B>Diagnostics: </B>Jones silver  
stain.|06/02/04
```

Short and sweet, and in no particular order. Anything following the vertical line does not show up on

the results page.

I do extensive research on medical terminology every single day, and I enter the new or changed data into the two text files which make up the database and upload them to the server every night. The php search program calculates the number of words, number of entries, and the last time the files were updated.

*MeDiCaLeSe*, the book and the website, concentrates on words that are new, ambiguous, or difficult to find, and includes words that are not found in the standard medical references, such as clinical trial acronyms, homeopathic and herbal remedies, products available only outside the U.S., discontinued drugs, and unapproved cancer therapies.

#### 4 Show me the money!

I worked hard for little compensation as a transcriptionist, and in the case of the website for no compensation at all. In 2004 someone suggested I turn my word database into a book. At first I resisted, because the daily updates and corrections were a large part of the value of the website, and I thought it was impossible to publish even a reasonably current book on such an enormous, constantly changing, and complicated subject, if typographic quality were to be a consideration.

I had seen a few quickly-printed books which were published in a hurry to cash in on current events, but their quality was atrocious — full of typographical errors and crookedly printed on brittle, yellowing newsprint.

I had guest-edited several medical terminology books for a large medical publisher in the mid-1990s. These books were very nicely produced, but as a transcriptionist I felt the information they offered was not quite enough. And, by now, I enjoyed having total control over my own website and was not about to relinquish that just to get a book published. If I went to the trouble to put a manuscript together, I felt that the big medical publishers would turn it away. I was good at Internet research, and I started looking into print-on-demand publishing.

I would have to typeset my own manuscript, but this didn't intimidate me, since I had worked in typesetting before. I had heard of T<sub>E</sub>X in the context of mathematical typesetting, and I started studying it on the Internet. I didn't have much money, and I appreciated the fact that T<sub>E</sub>X, besides being free, requires fairly minimal computer hardware. I continued to study T<sub>E</sub>X while organizing my database in manuscript form. I downloaded and experimented with T<sub>E</sub>X, joined TUG, and lurked on `comp.text.tex` and the T<sub>E</sub>Xhax list.

I write a little bit of short mystery fiction, and I knew several mystery writers who had either published with a subsidy house or gone out on a limb and published their own work. Subsidy publishing didn't interest me, as I needed to have the books available through the regular book-buying channels. Finally, a friend who started her own publishing company convinced me that I could do the same, and that I could have the books printed on demand by Lightning Source and distributed worldwide by Ingram. This was all I needed to hear! In April 2005 I obtained a business license and business checking account in the name of Blowtorch Press, and ordered a block of ISBN numbers. I named the book *MeDiCaLeSe 2005*, so that identifying subsequent revisions would be a no-brainer.

#### 5 Why T<sub>E</sub>X for medicine?

The large medical book publishers, such as Elsevier, usually have their own T<sub>E</sub>X style files available. Information on manuscript preparation for medical journals is available from the International Committee of Medical Journal Editors, <http://www.icmje.org>. Over 400 journals use the Vancouver style files, available at <http://www.ctan.org/tex-archive/biblio/bibtex/contrib/vancouver>.

For those who need the old apothecary symbols for minim, dram, scruple, etc., they are available in the PIXymbols font for Windows and Mac, sold at [http://www.vershen.com/psg\\_txtc.html](http://www.vershen.com/psg_txtc.html).

The Computer Modern font has almost all the diacritical marks and special characters needed in medicine. The `textcomp` package is needed only for the R symbol. I wanted the  $\mu$  symbol, the dot and umlaut in Åström, and the beta symbol in Dia $\beta$ eta to be properly typeset, even though these characters are not used in medical transcription; in that context, these words are written as micro, Astrom, and DiaBeta respectively. Medical transcription is usually done in a word processing program, and turnaround time is crucial.

By mid-2004 I had found the T<sub>E</sub>X editor I like best, which is L<sup>A</sup>T<sub>E</sub>X Editor by Shu Shen, a graduate student in Singapore. This software is free and very lightweight, which was an advantage since I didn't know much about T<sub>E</sub>X beyond what was needed to get the job done.

The php code does all the heavy lifting for the website. For a book, I would need to organize the data somehow. I put the entries into 39 chapters called Drugs, Abbreviations, Vocabulary, Cardiology, Neurology, and so on, alphabetizing the entries within the chapters, and then duplicating the entries into the different chapters as needed. I aimed for as

much redundancy as possible, because my primary intended audience of medical transcriptionists would not be willing to purchase the book unless they knew they could find what they were looking for quickly, and with enough information to know whether the word was the correct one for the situation.

Also because of the special requirements of the book's intended audience, I included this statement in the preface:

NO ADDED HYPHENS: We did not introduce new breaks into any of the words in this book. If you see a hyphenated word at the end of a line, it means the word should always be written with the hyphen. For the medical transcription community, we felt it was important to be clear on this, even at the expense of aesthetically undesirable line breaks. We have tried to make the book attractive and easily readable in spite of our self-imposed constraints on hyphenation.

To the best of my knowledge, *MeDiCaLeSe 2005* is the only medical transcription reference book with no added hyphenation.

I immediately saw that I would have no end of trouble without a bulletproof method of producing dictionary-style pages in double columns. I also needed the first and last words to appear at the top of the page on which they were defined. I began an intensive study of the `fancyhdr`, `geometry`, and `multicol` packages, and I tweaked the code until I was able to produce dictionary-style pages.

For double columns, all point size commands active at the end of each line must be the same. Otherwise, the lines of type do not match up from one column to the next.

Some of the main L<sup>A</sup>T<sub>E</sub>X file for the book appears below. I included my personal commands, which all begin with “en” plus one letter.

```
% c:\m05\medicalese.tex
\documentclass{book}
\def\enl{\filbreak\small\textbf}
\def\ene{\filbreak\normalsize\textbf}
\def\enn{\normalsize}
\def\eno{\enspace\scriptsize}
\def\enc{\filbreak\small\texttt}
\def\enb{\protect\raisebox{-2pt}}
\def\enu{\protect\raisebox{2pt}}
\usepackage{fancyhdr}
\usepackage[papersize={6.14in,9.21in},
margin={0.5in},top={1in},bottom={0.75in},
headheight={34.545pt},centering,
```

```
verbose=true]{geometry}
\usepackage[none]{hyphenat}
\usepackage{textcomp}
\usepackage{multicol}
\newcommand{\enk}[1]{#1\markboth
{#1}{#1}}
\begin{document}
% \frontmatter
\mainmatter
% COPY THIS CODE WHEN CHANGING HEADERS
% \end{multicols}
% \eject
\input drugs
\input notes
...
\input discont
\input notes
% COPY THIS CODE WHEN CHANGING HEADERS
% \backmatter
\ejct
\end{document}
\bye
```

Here is the beginning of one of the chapter files:

```
% c:\m05\labpath.tex
\pagestyle{fancy}\fancyhead{}
\fancyhead[LE,RO]{\textsf{\rightmark
\\leftmark}}
\fancyhead[C]{\textbf{Diagnostics}}
\fancyhead[RE,LO]{\textsl{\small includes
laboratory,\\pathology, and radiology}}
\renewcommand{\headrulewidth}{0.2pt}
\fancyfoot[LE,RO]{\textit{MeDiCaLeSe~2005}}
\fancyfoot[RE,LO]{\textsf
{www.medicalese.org}}
\renewcommand{\footrulewidth}{0.2pt}
\raggedright
\begin{multicols}{2}\\
% END OF CODE TO COPY WHEN CHANGING
% HEADERS ... diagnostics
\textbf{\enk{3-androstenedione}}\enn:
endocrine test ... \ene
{\enk{3D color Doppler}}\enn: developed ...
be measured by standard 2D imaging.\ene
```

And the end of the same file:

```
{\enk{Xplorer}}\enn~filmless radiographic
imaging system.\ene
{\enk{ZstatFlu}}\enn~throat swab, a
quick test for influenza A and B.\ene
\\
% COPY THIS CODE WHEN CHANGING HEADERS
\end{multicols}
\ejct
```

which produces:

**Xplorer** filmless radiographic imaging system.  
**ZStatFlu** throat swab, a quick test for influenza A and B.

## 6 Ugliness and badness

Because of the rule about not adding hyphens, the book contains some ugly paragraphs, like this:

**Avalide:** irbesartan/  
hydrochlorothiazide, combination  
angiotensin II receptor blocker/thiazide  
diuretic.

Also, the warning

```
Underfull \vbox (badness 10000) has  
occurred while \output is active
```

happened quite often. I learned to disregard these.

Here's one that doesn't look that bad, but  $\TeX$  complains that the line is just a tad too long (for my actual page width), and suggests an incorrect hyphenation:

**Markham-Meyerding** hemilaminectomy  
retractor.

```
Overfull \hbox (4.52963pt too wide) in  
paragraph at lines 689--689  
\OT1/cmr/bx/n/10 Markham-Meyerding  
\OT1/cmr/m/n/10 hemil-aminec-tomy
```

In a project such as this, the most insidious kind of badness is not seen until the `.pdf` file has been made and one is examining it page by page, since there is no way to know in advance where the page breaks will occur. I'm embarrassed to admit that I didn't catch this point size error in the guide words at the top of page 592 in time.

SaphLITE  
Songer

(Songer is incorrectly printed in `\small`, like `\LITE`.) When something like this happens, I go back to the input file and find the entry corresponding to the *first* guide word on the offending page. Just before the two closing brackets that define the first guide word, add the command to revert to normal size type.

## 7 Finished!

I spent a couple of hours a day for about six months typesetting the book. I gave up my transcription job in August to devote more time to the book, and finished the typesetting on September 14, 2005.

The book contained essentially the same information as the website of four days prior. Now I had to tackle the cover! At this point I started studying the `pstricks` package in earnest, and I purchased *The L<sup>A</sup>T<sub>E</sub>X Companion*, which is still the only  $\TeX$ -related book I've ever bought. Because of memory limitations, I wasn't able to use  $\TeX$  to make the front cover, but I did use it for the publishing company logo and for the text overlays on the spine and the back cover.

Altogether, the cover took me three weeks to make using Paint Shop Pro 7.04. This is not the printer's recommended software, but it was all I had. I made the cover in three pieces—front, spine, back—and pasted it together as a giant 300 dpi `.tif` file. Lightning Source had provided the bar code and I pasted this on the back cover. Then I burned the cover `.tif` file and the `.pdf` files which made up the interior to a CD and mailed it off to Lightning. Eleven days later, I received my proof copy by overnight delivery. I knew the `.pdf` files of the text wouldn't cause problems, but I wasn't at all sure how the cover would turn out. To my delight, it was absolutely beautiful.

I signed off electronically on the proof copy, and within days my book appeared for sale online at Barnes & Noble, Powell's, the university distributor eFollett.com, and Amazon, as well as other book-sellers all over the world. The book is selling within its niche market.

As publisher, I decide on suggested retail price, discount rate, and when/if a book goes out of print. *MeDiCaLeSe 2005* lists for \$41. I have no control over the actual selling price, and have seen the book advertised for sale at prices ranging from \$25 to \$80.

## 8 *MeDiCaLeSe 2006* and beyond

Today is October 22, 2006, and I'm in the final stages of preparing *MeDiCaLeSe 2006* for publication. The page count is 744 this time around. I discovered along the way that the upper limit of what one  $\TeX$  file can process on my machine is about 713 pages of size 6.14"  $\times$  9.21". As soon as I finish proof-reading the final printed `.pdf` output, I'll make the cover—this time with `pstricks`, specifying CMYK colors for high quality color reproduction. I now have 512 mb of RAM, which I hope will be enough.

When the total number of pages exceeds 828, I will either have to increase the page size or look for an alternative to print-on-demand publishing. The average age of the book's target audience of medical transcriptionists is late 50s or older, so it would not be practical to decrease the point size.

# L<sup>A</sup>T<sub>E</sub>X at a liberal arts college

Jon Breitenbucher

Department of Mathematics and Computer Science

308 E. University

The College of Wooster

Wooster, Ohio 44691 USA

`jbreitenbuch (at) wooster dot edu`

`http://jbreitenbuch.wooster.edu/~jonb/`

## Abstract

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.

## 1 Background

The College of Wooster is a small liberal arts college located in northeastern Ohio. Wooster's annual enrollment is around 1800 students and the Mathematics and Computer Science Department typically has 25–30 majors per year. One of the distinguishing features about Wooster is its Independent Study (IS) program founded by its seventh president Howard Lowry. The independent study program requires every senior to complete a year long independent research project. A typical faculty member will advise 3–4 such projects. One of the challenges I encountered in my first advising experience was getting students to write technical mathematics. This is what lead me to introduce L<sup>A</sup>T<sub>E</sub>X at Wooster.

The majority of seniors use Microsoft Word. This probably comes as no surprise to most of the readers. In fact, for the vast majority of seniors, Word is the appropriate tool. However, Word is not the best tool for everyone. Students writing in foreign languages that requires special fonts and the ability to have text go from right to left could benefit from using L<sup>A</sup>T<sub>E</sub>X in conjunction with X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X. Students in music might find MusiX<sub>T</sub>E<sub>X</sub> to be a better environment for preparing their theses. However, I think the science majors have the most to be gained from switching to L<sup>A</sup>T<sub>E</sub>X. Science majors typically have a large number of equations, figures, and tables. Having used L<sup>A</sup>T<sub>E</sub>X for my dissertation, I knew that it could do a much better job of formatting the theses of the science majors. I decided to introduce L<sup>A</sup>T<sub>E</sub>X into my department first and then to expand into other departments. In this article I would like to outline my approach, some of the issues that I encountered, and student reaction.

## 2 Why L<sup>A</sup>T<sub>E</sub>X?

As a student at Wooster I had struggled with Word version 5.5 on the Mac to produce a passable document. Some of you may remember that the earlier versions of Word were not too different from T<sub>E</sub>X. One would type in a command sequence to get a sum, product, or other symbols. It may even be the case that some of these command sequences survived into the present day, but people have long forgotten their existence.

When I returned to teach at Wooster I was shocked by the poor (typesetting) quality of the Independent Study theses. Current students were not able to produce a document that looked anywhere near the quality of my thesis from nine years earlier, and this was with more advanced versions of Word. I found that the students spent weeks trying to format their theses to make sure that section numbering, equation numbering, and figure numbering were correct (and most did not succeed). Almost none of the students knew how to have an automatic Table of Contents, Figures, etc. created by Word. It was at this point that I asked myself whether L<sup>A</sup>T<sub>E</sub>X could make the process of writing a thesis more about the writing and less about the formatting. Why L<sup>A</sup>T<sub>E</sub>X? Because I felt that it was strong in all of the areas with which students were having trouble. L<sup>A</sup>T<sub>E</sub>X would handle the numbering, formatting, front matter and back matter and the students could just worry about the content.

However, there is a down side which Neuwirth touches on in Neuwirth [1991]; none of the students know L<sup>A</sup>T<sub>E</sub>X. This means that someone has to be willing to teach them and answer their questions. However, this situation is different than that addressed by Neuwirth. Neuwirth was discussing the place of T<sub>E</sub>X in what would be considered middle

school and high school in the United States. And I agree that those students don't need the full power of  $\text{T}_{\text{E}}\text{X}$ , but I'm not sure that they couldn't benefit from an introduction to  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ . One of the questions that my experiences have raised is, "Where do our students learn how to use Word or other document preparation software?" I have been unable to find anyone that knows the answer to this question. Some of my colleagues assumed that our Writing Center was helping students learn how to write technical documents, but in talking to the Writing Center staff I found this was not the case. What we have found is that most of our math and science students begin college or university study with no idea of how to use Word or other tools to write a technical paper. However, they are expected to be able to produce a technical paper when they get to graduate school. This being the case, it is incumbent on us to teach them, so that is what I decided to do with the students at Wooster. There is no release time or other compensation for this; I do it because I love doing it.

### 3 The process

So how did I go about getting  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  into our program? The first step was to identify exactly what I wanted to accomplish. As mentioned above, I definitely wanted the students to let  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  handle all of the formatting. What does that mean? I decided it means that I don't want students to have to load packages, learn the intricacies of incorporating graphics, or have to try to force  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  to do something that Word can do. In turn, what this means is that I needed to construct a Wooster thesis class. (I leave the explanation of the difference between a class and package to a more knowledgeable  $\text{T}_{\text{E}}\text{X}$ er.)

Before beginning to construct a thesis class for Wooster, I examined a number of classes available at other institutions. During this process I discovered two things: none of the classes did exactly what I wanted and almost all of them were modifications of the standard book or article class. After realizing this, I decided to try to modify the book class myself using a couple of other thesis classes<sup>1</sup> as models. To meet the stated goal above, my class has to load all of the packages I think students will need or provide a class option which will load certain packages. At first I was only loading a few packages, but as students have used the class I have added more packages and options. I think the current mix<sup>2</sup> serves my students very well, and I don't envision them

<sup>1</sup> I used the *kthesis* and *osuthesis* classes as models.

<sup>2</sup> My class loads *ifpdf*, *amsthm*, *amssymb*, *amsmath*, *setspace*, *graphicx*, *eso-pic*, *natbib*, *float*, *caption*, *subfig*,

needing any more packages. This process was not easy and I wish I knew and had known more about writing a class file. I would recommend that a beginner or intermediate user find an expert  $\text{T}_{\text{E}}\text{X}$ nician to help them write or modify a class file. Doing so will save a lot of hair pulling and time spent in trial and error.

At this point I sought input from my colleagues, the Registrar, the Secretary of the College, and the Vice President for College Relations and Marketing to make sure that the format and images used were acceptable. Others might not need to include so many people, but since IS is such a major component of our curriculum, I needed to make sure everyone liked the design. I was told to change a few things and resubmit, at which point my design was approved. Others will probably find that they will have a similar experience. Now it was time to involve the students.

#### 3.1 Editors, platforms, and documentation, oh my!

There are a few things that I had to do before I could start showing students how to use  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  and my class file. The first is dealing with the platform issue. I am very committed to allowing users to choose the operating system they are comfortable with using. I have almost 20 years of experience on the Mac OS and as such I know almost all the  $\text{T}_{\text{E}}\text{X}$  editors available. On the other hand, I don't know much about Linux and have only seven or so years of experience with Windows. So my first task was to identify software packages for each of the three major OS variants.

If you find that you need to do this keep in mind that the school will probably not want to buy software, so free or low-cost shareware solutions are desirable. After some research I settled on the following: *TeXShop/gwTeX* for Mac OS X, *TeXnicCenter/MiKTeX* for Windows XP, and *Kile/teTeX* for GNU/Linux. Why these packages? I chose these packages because they all provide panels or menus for common  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  tasks, are free, and are as close to the point-and-click Word model as I could find. They also do not require nearly as much technical ability to install and use as something like Emacs. Remember a number of the students may not be technically savvy, so the more like Word the better. Emacs is great and would have made for a more uniform environment, but I was afraid the level of technical ability required to install and use Emacs would scare

color, and *hyperref* and has options for *pxfonts*, *floatflt*, and *listings*.

away the weaker students — the ones I most want to use L<sup>A</sup>T<sub>E</sub>X.

Once I settled on software packages I was comfortable supporting, it was time to document the thesis class and introduce L<sup>A</sup>T<sub>E</sub>X. I chose to document L<sup>A</sup>T<sub>E</sub>X and the thesis class by using the thesis class to write the documentation. In this way I am able to give students a zip archive containing all the files needed to produce the documentation. In addition the students can use the archive as a template for creating their theses; they just need to make a copy of the folder they get when they unzip the download and start putting their content into the files. This has worked very well as they can see the code that I used to achieve something and copy and paste or alter it to their needs.

My documentation<sup>3</sup> covers basic things such as starting new chapters and sections, creating lists, making things bold or italics, including graphics, inputting mathematics, and inputting computer code. It does not cover installation of a T<sub>E</sub>X system or the software (that is left to the authors of the software). It is really a summary of things found in [Kopka and Daly \[2003\]](#), [Mittelbach et al. \[2004\]](#) and [Flynn \[2003\]](#), except for the Typesetting Mathematical Formulae chapter which comes from [Oetiker et al. \[2003\]](#). The intent is for students to teach themselves how to use the few L<sup>A</sup>T<sub>E</sub>X commands that they need and to come to me if they have difficulty. Choosing editors with panels or menus for L<sup>A</sup>T<sub>E</sub>X input makes this possible. This is a much different approach than used by [Gray and Costanza \[2003\]](#) and [Childs \[1989\]](#) where there is an actual course where students learn T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X. Students have done reasonably well under my setup, but a course where some introductory L<sup>A</sup>T<sub>E</sub>X could be covered would be desirable. My department is considering trying to move technical writing issues into the proofs/introduction to higher level mathematics course, but it is hard to cut content in favor of this new material.

### 3.2 Involving others

In writing the class file I tried to make it as general as possible to allow other departments to use it. After a year of use in the Math and CS department, I introduced the class file and L<sup>A</sup>T<sub>E</sub>X to the physics students. The students picked up L<sup>A</sup>T<sub>E</sub>X very easily and liked the results. They particularly liked the fact that I had set everything up to use pdfT<sub>E</sub>X and produce a “live” document. However, some of the physics faculty did not like the design of the output and so they have modified the class to produce a

different-looking document. Others trying to introduce L<sup>A</sup>T<sub>E</sub>X may find this as well. Make sure everyone knows you are not responsible for modifying the class file or troubleshooting others’ changes; otherwise you will find yourself maintaining ten slightly different versions of the same class.

After introducing L<sup>A</sup>T<sub>E</sub>X to the physics students, I approached Chemistry and Biology. My plan was to move through the sciences and then approach Music, Classical Studies, Chinese, and Arabic. I discovered that neither the Chemistry nor the Biology department were interested in introducing this to their students, their main concern being that no one in their departments was familiar with L<sup>A</sup>T<sub>E</sub>X. I met with the same response from Classical Studies, but in this case no one had even heard of L<sup>A</sup>T<sub>E</sub>X. This is a real issue when trying to introduce L<sup>A</sup>T<sub>E</sub>X. In retrospect, I should have identified a few individuals in each department to learn L<sup>A</sup>T<sub>E</sub>X from me. Those people would then act as point people for their students and would use me as backup. A faculty workshop designed around the material of [Gray and Costanza \[2003\]](#) might be a way to accomplish this.

So, as it stands now, Math and CS and Physics are the only departments using L<sup>A</sup>T<sub>E</sub>X, which is not surprising when one browses through the various mailing lists and samples the common fields-of-study. Involving people from other departments from the start might have made a difference. I would suggest that if others try this, they develop a clear plan for implementation and have a timeline to measure progress.

### 4 Assessment

So how did I do? That’s hard to say because I didn’t have a formal assessment plan in place. My assessment has been in the form of an informal Pizza Party after all the seniors have completed their theses, and two questions on the departmental IS evaluation form. This is not what I would recommend for others. Unfortunately, I am beyond the stage for assessing the success of the introduction, and have lost that chance. What I am doing is developing materials to assess the process of learning L<sup>A</sup>T<sub>E</sub>X so that I can improve that process and make it easier for students.

There are a few things that I can communicate in an anecdotal manner. In general the students have felt that this model is working well. The first group of students suggested introducing L<sup>A</sup>T<sub>E</sub>X earlier in the curriculum. I took that recommendation and created a homework package and template file and encourage sophomores to use it and require ju-

<sup>3</sup> [http://jbreitenbuch.wooster.edu/pdf/latex/IS\\_guide.pdf](http://jbreitenbuch.wooster.edu/pdf/latex/IS_guide.pdf)

niors to do one assignment in  $\LaTeX$ . I have also started requiring all homework submissions to be typed in sophomore-level classes and above. Students also suggested the need for various capabilities for placing images and styling chapter headings. I incorporated the packages necessary to accomplish this in the class file. The result has been that no one had any suggestions at this year's pizza party.

The students also felt like they did focus more on the writing, but there are some formatting issues that really bother them. Image placement is a big source of frustration. The students are used to dragging an image into the document exactly where they want it and having it stay. It takes them some time to get used to letting the images float and to use references to refer to their images. The other frustration is learning commands. It takes them a few weeks to really get the hang of things. However all of them said these minor issues are more than compensated for by the auto-generation features of  $\LaTeX$ , and they are glad they took the time to learn  $\LaTeX$ .

Has this process improved the writing? This is difficult to answer. I used these questions to measure this on the IS evaluation:

- Based on your discussions with this IS student, the bibliography, and the final written document, which statement best describes the student's assimilation of the material?
  1. The student assimilated material from a wide variety of sources.
  2. The student used material from multiple sources and did some assimilation of that material.
  3. The student used material from multiple sources.
  4. The student primarily used material from one source, but did use some material from at least one other source.
  5. The student used one primary source from which all material is taken.
- Based on the final written document, which statement best describes this IS?
  1. The IS is written in a clear and well-organized manner, with excellent grammar, spelling, and typesetting. Moreover, it is written in the student's unique style and directed toward an audience of peers.
  2. The IS is very readable, with very few errors in spelling, grammar, or typesetting. The thesis is well-organized.
  3. The IS is readable, despite some errors in spelling, grammar, or typesetting. The thesis is well-organized.

4. A number of errors in spelling, grammar, or typesetting make this IS somewhat difficult to read. A better organization of ideas would have made it more clear.
5. The IS lacks organization, the grammar is poor, and it is difficult to read.

I chose these questions because my goal is to make the IS experience more about the writing and less about the formatting. If I am succeeding then students using  $\LaTeX$  should assimilate more and produce a better written document. Of course I cannot set up a control group and conduct a true study to control for all the confounding factors, but anecdotally I can say that, in general, students using  $\LaTeX$  have scored better on these questions than those who have not. My colleagues also agree that in their judgement  $\LaTeX$  has increased the overall quality of the IS produced by the students.

So I would say that my attempt has accomplished my goal. For anyone planning on doing something similar, an assessment plan for all phases is a must. I say this because more and more accreditation bodies want to see evidence showing the success or failure to meet stated goals. Also, I do not think that you have to have a senior thesis to try this. Programs with writing across the curriculum could also see an improvement in student performance, and might have an easier time of assessing  $\LaTeX$ 's impact.

## 5 The future

So now what do I do? There are a few things I hope to do in the next few years. One is to expand the use of  $\LaTeX$  into the foreign languages. The introduction of  $X\TeX$  and Mac OS X makes it extremely easy to typeset in foreign languages. I think that students studying Eastern languages would find great benefits to using the  $X\TeX$  system, and I hope to be able to talk to the faculty in those disciplines in the near future. Another goal is to increase the use of  $\LaTeX$  in lower level courses, which will require training my colleagues in the use of  $\LaTeX$  and will allow students to learn  $\LaTeX$  at a much slower pace.

## 6 Acknowledgments

I want to thank Karl Berry for encouraging me to write about my experience and providing several articles relating to the topic. I thank the reviewers for their insightful comments. And most of all I thank all of the Wooster students who have used  $\LaTeX$  and my class file for their ISs and provided valuable feedback; without them this project wouldn't exist.

**Bibliography**

- Bart Childs. Teaching T<sub>E</sub>X. *TUGboat*, 10(2): 156–163, 1989. URL <http://tug.org/TUGboat/Articles/tb10-2/tb24childs.pdf>.
- Peter Flynn. Formatting Information. *TUGboat*, 23(2):115–237, 2003. URL <http://www.ctan.org/tex-archive/info/beginlatex/beginlatex.letter.pdf>.
- Gary Gray and Francesco Costanza. Experiences and lessons learned teaching L<sup>A</sup>T<sub>E</sub>X to university students. *TUGboat*, 24(1):124–131, 2003. URL <http://tug.org/TUGboat/Articles/tb24-1/gray-class.pdf>.
- Helmut Kopka and Patrick W. Daly. *Guide to L<sup>A</sup>T<sub>E</sub>X*. Pearson Education, New York, 4th edition, 2003.
- Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison Wesley Professional, New York, 2nd edition, 2004.
- Konrad Neuwirth. T<sub>E</sub>X in Schools: Just Say No. *TUGboat*, 12(1):171–173, 1991. URL <http://tug.org/TUGboat/Articles/tb12-1/tb31kneuwirth.pdf>.
- Tobias Oetiker, Hurbert Partl, Irene Hyna, and Elisabeth Schlegl. The Not So Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. 2003. URL <http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf>.

# Design of presentations: Notes on principles and L<sup>A</sup>T<sub>E</sub>X implementation

Boris Veytsman

Computational Materials Science Center, MS 5A2

George Mason University

Fairfax, VA 22030

borisv@1k.net

## Abstract

There are many T<sub>E</sub>X packages available for creating presentations. Mostly they imitate the ubiquitous style of a certain tool, striving to produce PowerPoint-like slides, hopefully with better typographical execution.

In this talk the principles of good design for presentations are considered. We discuss the problems with the common design of presentations as well as the famous proposition by Tufte to avoid slides at all. We try to formulate the principles of good presentation design and discuss T<sub>E</sub>X implementations from this point of view.

The discussion is based on the author's experience in making slides for talks, lectures and training sessions.

## 1 Introduction

An often asked question in news groups like `comp.text.tex` and other T<sub>E</sub>X forums is this: “I want to create a nice presentation like my colleagues do, but I want to have beautiful math. Is it possible to make one using L<sup>A</sup>T<sub>E</sub>X?”. The indispensable T<sub>E</sub>X FAQ (UK T<sub>E</sub>X Users Group, 2006) has a number of answers (see <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=slidecls>). However, this is in my opinion the wrong question. It is akin to the question, “How can I create texts like my colleagues using word processors do?” Of course you can do this (and CTAN has a package for such task!), but why would you want to add an ugly text to so many ugly texts already existing?

The right question, in my opinion, should be structured differently. One should ask first, “What kind of presentations should I make, if any?” Only after this question is answered, one may think about the tools to create the right kind of presentation. Here, as in many other fields of Computer Science, the approach should be goal-oriented rather than tool-oriented.

This paper tries to answer the first question and describe the “right kind of presentation”. Then we outline the tools to create such presentations in the L<sup>A</sup>T<sub>E</sub>X document preparation paradigm.

It should be noted that this paper describes the personal experience and opinions of the author. They are necessarily subjective and reflect the author's idiosyncrasies and tastes. Therefore I recommend that the reader take my conclusions with the usual grain of salt. This paper is intended to be

a starting point for the reader to think about presentations rather than an exhaustive and balanced guide to presentation design and implementation.

## 2 The bane of slides

If we judge by the folklore of office dwellers, presentation slides are an instrument of torture comparable to the tools of the Inquisition. A Google search for “Death by PowerPoint” gives about 5.6 million links (Muir, 2006). The leading expert in graphical design, Edward Tufte, makes a convincing case to banish slides altogether (Tufte, 2003). He shows that the misuse of viewgraphs leads to *information obscuring* and shows how this hiding is at least partially to blame for the two tragic NASA catastrophes during *Challenger* and *Columbia* flights. The author of this paper has had his share of listening to presentations accompanied by slides, and can attest to the detrimental effect of the projector on the audience's attention and the level of discussion. Unfortunately the “culture of PowerPoint” exposed by Tufte is spreading. One of my students, a federal contractor, complained that after his company submitted the results of the engineering study ordered by an esteemed US agency, the customer asked him to supply “PowerPoint slides with the results of the study”. “Our white papers had all the data and conclusions in a form suitable for engineers”, lamented the student, “but they wanted a dumbed-down version on a dozen slides”.

Tufte argues that the cognitive style of PowerPoint presentations has the following characteristics: “foreshortening of evidence and thought, low

spatial resolution, a deeply hierarchical single-path structure as the model for organizing every type of content, breaking up narrative and data into slides and minimal fragments, rapid temporal sequencing of thin information rather than focused spatial analysis, conspicuous decoration and Phluff, a preoccupation with format not content, [and] an attitude of commercialism that turns everything into a sales pitch” (Tufte, 2003, p. 4). This leads to his advice to abolish slides and use printed handouts instead. He quotes a funny parody of the Gettysburg Address where the moving speech was turned to a series of boring items and meaningless charts (Norvig, 2000).

This is a drastic recommendation. It is tempting to declare slides completely broken and return to the good old days of great speeches. However we should remember that in those days Gettysburg Addresses were not the sole oral genre. There were university lectures, engineering reports, and other oral presentations that required some visual materials. We discuss this and the lessons learned in the next section.

### 3 The lessons of the blackboard

Almost any lecture in mathematical or physical sciences since ancient times was accompanied by chalk and a blackboard. Equations were derived on the blackboard. Important theorems were written down and often a frame was drawn around the most important ones. The professors teaching humanities used blackboards less often, but some of them wrote important points and conclusions on the blackboard. It was considered to be an important teacher’s skill to have a clear and logical blackboard at any time during the lecture. The author of this note, by the way, had a chance to study the art of teaching in the pre-slide days, and was often scolded by the professors because his own blackboard notes were not designed well enough. Obviously the importance of the visual accompaniment to a lecture was self-evident to the professors.

What lessons can we learn from classical blackboards?

First, the blackboard was always an auxiliary, a tool, but never the centerpiece of the lecture. It was unthinkable for a lecturer to start with the blackboard design, and then create a lecture around it. Even stranger would be the idea of writing the lecture on the blackboard, and then reading it from there. A lecturer worked in a well-lit room, and the attention of the audience was on him, not on his board. This is well captured by the famous photograph by Tarasevich (Tarasevich, 1977). What a contrast to a modern-day presenter, working in a

dark room, the screen with projected slides being the only bright spot, and the silhouette of the presenter himself almost lost in the shadows!

Second, the bane of many slides, the bulleted list, did not work well on the blackboard, and was used sparingly if ever. Actually, an itemized list is not a good way to convey information in general, not just in presentations. One of the unfortunate decisions of Leslie Lamport in the early L<sup>A</sup>T<sub>E</sub>X design was probably to provide an easy way to create itemized lists (Lamport, 1994). Too many authors overuse and misuse such lists.

Third, there is a very important issue of the speed and rhythm of a presentation. The text written on a blackboard appears slowly enough for the audience to understand it. This is especially true for equations: the process of derivation of formulas in real time has an enormous education potential. In contrast to this, the information on slides often appears too fast, and the audience often cannot grasp it, especially complex equations.

These simple considerations lead to the following principles of presentation design.

### 4 Principles of presentation design

The first principle of presentation design is: if you can do without slides, do not use them. The blackboard provides a good rhythm and tempo to lecture from. This rule is almost always applicable to lectures or training sessions, where a deep understanding of material is required. The situation is different for review presentations, where the aim is to give the audience a general picture of the subject and to stimulate an interest in the subject. In this case breadth trumps depth, and slides provide a better vehicle than a blackboard.

Slides may serve another useful function: they can be given to the audience to keep after the presentation. A better alternative would be to give the audience preprints instead (Tufte, 2003), but this is not always practical. This conference (Practical T<sub>E</sub>X 2006) provides an interesting case for comparing these two forms of materials. Authors were urged to provide preprints to be included in the program, which will eventually be published in *TUGboat*. These preprints take much more time and effort to prepare than slides with their “choppy” style. One could argue that they serve the audience better. Nevertheless, slides in some cases can be a cheap and efficient substitute for printable papers and manuals, as long as they are used sparingly.

This leads to the second principle: good slides must be designed for on-screen viewing by the audience after the presentation. This means that slides

may (and often should) have such typographic attributes as epigraphs, footnotes and references; if they are not necessary for the presentation itself, they will be of use to the interested audience after the lecture.

The third principle is that the slides, even if used, are not the centerpiece of the presentation. The most important part of the lecture is the body of thoughts the lecturer is trying to convey; the structure of the slides is secondary to these thoughts.

While this principle seems to be self-evident or even trivial, it is easy to point out many situations where it is violated. One of the transgressions is to divide the presentation into self-contained chunks exactly one slide long. Some slide making software even writes the name of slide on the top (or uses the ugly “Continued...”), thus urging authors to think in terms of slides. Ideas are naturally mapped into a traditional hierarchy of sections (and sometimes subsections, etc.), and the length of the section should be determined by its content, not by the length of the slide.

If we follow the lessons of the blackboard, we should use step-by-step building of slides, comparable to step-by-step writing on the blackboard. Unfortunately this is overused by many authors, who, as noted by Tufte, reveal a line of their slide, read it aloud, than reveal another line, etc., making a bad design even worse. A more fortunate example is shown in Figure 1. The last diagram there looks intimidating for a non- $\text{\TeX}$  audience, and might shock it into inattention. A step-by-step revealing of points with a discussion of each is a better way to explain the usual  $\text{\TeX}$  work flow, especially if accompanied by a judicious use of color.

Another transgression is the overuse of itemized lists, which is already mentioned above, and which is imposed on the authors by some slide making software systems. The structure of an itemized list is seldom good for a thought of any complexity. The usual misuse of dingbats as bullets makes these lists especially ugly.

A very rare example of a justified use of itemized lists and dingbats is shown on Figure 2. Note that in this case the bullets actually convey some information to the reader.<sup>1</sup>

## 5 $\text{\TeX}$ nical notes

This part of the paper is based mostly on the experience of the author, and even more than the other parts reflects his own tastes and preferences. I try

<sup>1</sup> The bullets are based on the images distributed with the Debian package `gnome-extra-icons` under GPL.

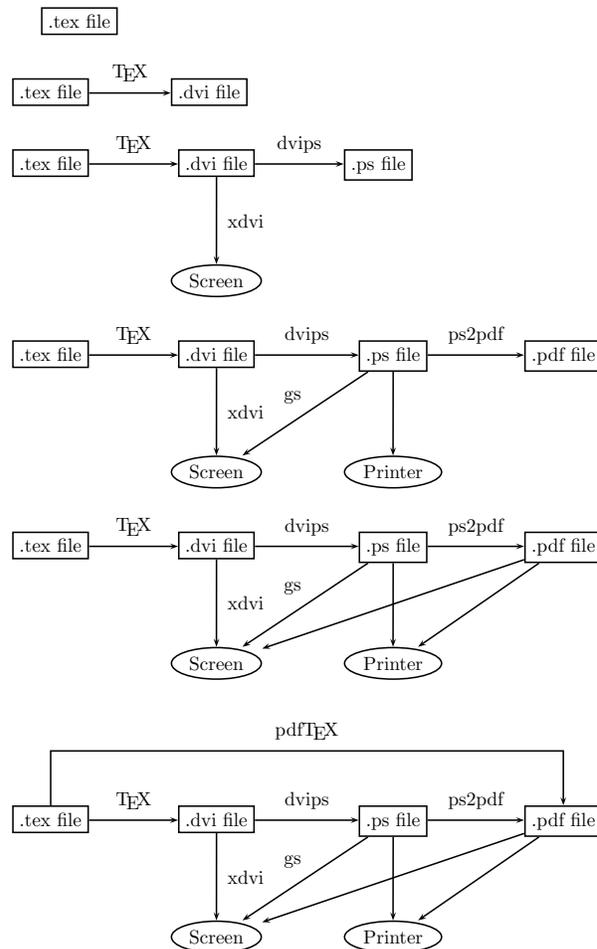


Figure 1: Step-by-step building of a flowchart

to explain what turned out to work for me, and do not claim to make a comprehensive list of tools and methods available (see the  $\text{\TeX}$  FAQ for examples and comparisons) or state that my choices are necessarily the best ones. This is why I illustrate these ideas with examples of my own presentations. I do not claim these presentations are especially good — they just reflect my approach to the design.

The remainder of this section is organized in chronological order: I describe the history of my journey into  $\text{\TeX}$ -based presentations and explain how I chose my tools and which lessons I learned.

The traditional `slides` class of  $\text{\LaTeX}$  (Mittelbach, 1997) was never sufficient for me because it has the `slide` environment, and I never liked to think in terms of slides. For the same reason I was not successful with the `Beamer` class (Tantau, 2005) either. While this class allows sectioning, the `frame` environment and explicit overlays are a little too low level for my taste. I wanted the separation

There are several kinds of users in Unix:



“Normal users”. These are people. John Doe  $\mapsto$  jdoe. The system knows a “real name” and *login*.



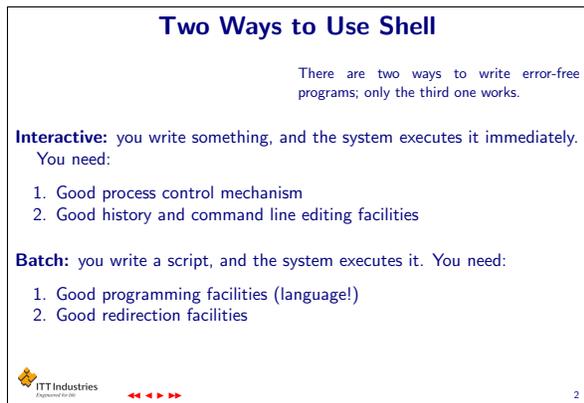
“System users”. They are owners of services: web service, time service, ftp service, etc.

Services are run by special processes, usually started up at boot up. These processes are called *daemons*.



Superuser or root. The classical system has exactly one superuser, who is allowed to do everything. It is used for system maintenance only.

**Figure 2:** A rare example of acceptable itemization



**Figure 3:** A PDF slide created with FoilT<sub>E</sub>X (Veytsman, 2001)

of the text into slides to be automatically generated by the sectioning commands, with some manual adjustments using `\clearpage`. This requirement was motivated mainly by laziness, of the kind described by Larry Wall (Wall, Christiansen, and Schwartz, 1996): I wanted the transition from slides to preprints and articles and back to be as smooth as possible. FoilT<sub>E</sub>X (Hafner, 2002), while lacking the sectioning, allowed the text to “overflow” slides, and this system was my initial choice. The addition of sectioning commands to this package was relatively simple (Veytsman, 1998). I used this scheme for half a decade. When I started (it was the previous millennium!), PDF format was not as widespread as now, so I used FoilT<sub>E</sub>X to format the transparencies used in the classroom, and L<sup>A</sup>T<sub>E</sub>X2HTML (Drakos and Moore, 2005) to create the Web pages for the students to use after class. This is how the course notes (Kumar and Veytsman, 1996; Veytsman and Kotelyanskii, 1997) were made.

HTML provides very limited facilities for typography. PDF format is definitely better suited for online presentations and slides. One of my first ex-

periments with slides in PDF format, suitable for class and online viewing, is Veytsman (2001). It was done in FoilT<sub>E</sub>X with pdfT<sub>E</sub>X. A slide from this presentation is shown as Figure 3. The slides made in this way are quite readable and easy to prepare. However, there are several problems with FoilT<sub>E</sub>X. First, it is not easy to prepare step-by-step slides. Second, FoilT<sub>E</sub>X imposes too much white space on slides, which is not well suitable for online viewing. The density of the material is too low. Another concern is navigation tools. They are present in the slides (Veytsman, 2001), and FoilT<sub>E</sub>X allows them to be placed easily on the bottom of the page, but not to the left or to the right.

A good question to ask is whether we need navigation tools as well as logos on the slides? They take up space without adding much to the information content. Tufte (2003) puts logos on slides into the “chartjunk” category. Most viewing software has a navigation menu, so a navigation panel seems to be a useless duplicate. However, a case could be made for navigation panels on slides. First, many people prefer to view slides in full screen mode, and adding a panel helps in this case. Second, the viewing software provides a generic navigation bar; the presentation author may want to use the customized one. Third, the proportions of a usual slide are wrong: they are wider than tall, and the text width is too large by any measure. By the way, FoilT<sub>E</sub>X logos make the situation even worse: by design they subtract from the *height* of the page. The usual Web pages have the same problem of too large text width. Good Web designers create wide right or left margins on their web pages. A navigation panel with the logos to the left or to the right of the slide would serve the same function as wide margins in books, journal papers or good Web pages.

Last but not least, it is good to give the viewer a general idea where he or she is in the presentation or lecture. Traditionally, lecturers wrote the current

George Mason University

School of Computational Sciences

Thermodynamics . . .

Closed Loop . . .

Computer . . .

Conclusions

Home Page

◀ ▶

◀ ▶

Page 6 of 19

Go Back

Find

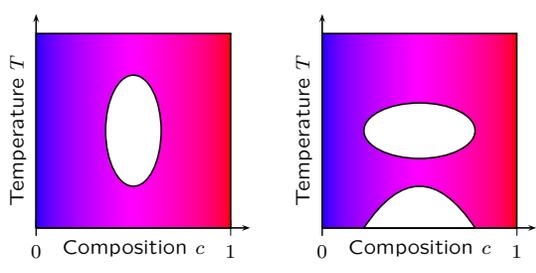
Full Screen

Close

Quit

## 2. Closed Loop Phase Diagrams

Guiacol-glycerol;  $\beta$ -picoline-water, many polymers<sup>1</sup>:



Van der Waals forces cannot produce this—there must be something else

<sup>1</sup>T. Narayanan and A. Kumar, “Reentrant Phase Transition in Multicomponent Liquid Mixtures,” *Phys. Reports* 249 (1994): 136–218.

Figure 4: A PDF slide created with pdfscreen (Veytsman, 2004)

topic (section or subsection) on the blackboard and changed it when they moved from topic to topic. Common presentation software tries to achieve the same purpose by adding a title to each slide. This is a bad idea because it imposes the rigid scheme “one thought, one slide”. The ugly scheme, “*Topic blah* on the first slide and *Topic blah (cont.)* on the subsequent ones”, only stresses the inadequacy of this design. Beamer (Tantau, 2005) provides another solution: it allows adding copies of the table of contents before each section with the current topic highlighted. While this is definitely better than titles on slides, there are still disadvantages to this approach: the flow of the lecture (or on-screen viewing) is interrupted by basically the same page. Also, this table of contents is visible only at the section breaks, not in the middle of a section. A discreet reminder in the form of short table of contents on the wide margins of a slide seems to be the ideal solution.

An astute reader could note that the last paragraph describes the approach of the pdfscreen package (Radhakrishnan, 2000). This package provides

powerful facilities for on-screen presentations. After I discovered and tried this package, it was clear that I had found what I was looking for. I have been using it for the last several years. Some examples of its use can be found in Veytsman (2004, 2006); see also Figures 4 and 5.

It is worth noting that all lectures except the first in the course (Veytsman, 2006) were taught using the blackboard only: the slides were made for the students to view after the lectures (when working on their homework and exams). The first introductory lecture was the only exception: it was taught using a projector. The aim of this lecture was to deliver a “general picture” rather than teach detailed knowledge.

There are several noteworthy details in the usage of pdfscreen. First, the default fonts for this package are serifed ones. I think it is better to use sans serif fonts for presentations, since they look better at low resolutions. The package tpslfonts



School of  
Computational  
Sciences

One Barrier  
Complex Transport...  
Two Barriers And...  
Homework

NANO500

◀ ▶

◀ ▶

Page 2 of 16

Go Back

Find

Full Screen

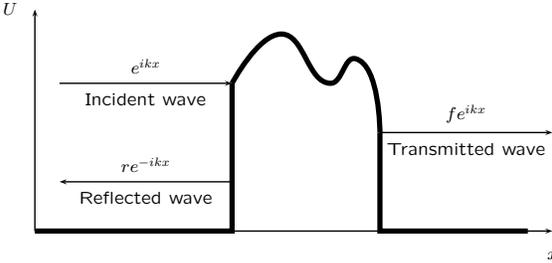
Close

Quit

## 1. One Barrier

### 1.1. Transmission And Reflection

Classical mechanics: particles are reflected by a barrier or penetrate it. Quantum mechanics: they do both.



Incident current  $\propto 1$ . Transmitted current  $\propto |f|^2$ . Reflected current  $\propto |r|^2$ .

Flux Transmission:  $K_f = |f|^2$ . Reflection Coefficient:  $K_r = |r|^2$ . Current conservation:

$$K_f + K_r = |f|^2 + |r|^2 = 1$$

Our goal is to calculate flux transmission and reflection coefficient.

Figure 5: A PDF slide created with pdfscreen (Veytsman, 2006)

(Lehmke, 2004) provides well designed and readable fonts, which are my personal favorites. Another package by Lehmke, T<sub>E</sub>Xpower (Lehmke and Nordhaug, 2005) is useful for providing step-by-step slides and overlays (although its page transitions are too glitzy for my taste).

Slides often require graphics. There are many different programs to generate them. Again, my personal favorite is PStricks (Van Zandt, 1993). A package like ps4pdf (Niepraschk, 2004) is useful for adding PStricks pictures in slides. A fragment of a Makefile that automatically generates all the intermediate steps is shown in Figure 6.

## 6 Conclusions

My experience of teaching and making presentations shows that slides are usually a bad idea both for in-class use and online viewing after class. In the first case slides cannot compete with the blackboard, and in the second case a good writeup is better than slides. Still, in some situations slides still can be used: for example, a need to show breadth rather

```

%.pdf: %.tex %-pics.pdf
$(RM) $*.toc
pdflatex $*
- bibtex $*
pdflatex $*
while (grep -q \
'Label(s) may have changed'\
$*.log) do pdflatex $*; done

%-pics.pdf: %-pics.ps
ps2pdf $<

%-pics.ps: %.dvi
dvips -Ppdf -o $$ $<

%.dvi: %.tex
latex $<

```

Figure 6: A fragment of a Makefile for use in PStricks and PDF slides

than depth, a subject requiring many pictures, etc. It is important to remember in this case that slides must be adapted to both in-class and after-class use, and that slides are a tool, not the centerpiece of the lecture.

L<sup>A</sup>T<sub>E</sub>X tools provide ample opportunity to produce beautiful and instructive slides. My attempts to do this, documented above, show the power of these tools.

### Acknowledgements

The author is grateful to Lance Carnes for the careful reading of the manuscript and many suggestions.

### References

- Drakos, Nikos, and R. Moore. *The L<sup>A</sup>T<sub>E</sub>X2HTML Translator*, 2005. <http://www.ctan.org/tex-archive/support/latex2html>.
- Hafner, Jim. *The FoilT<sub>E</sub>X Class Package*, 2002. <http://ctan.tug.org/tex-archive/nonfree/macros/latex/contrib/foiltex>.
- Kumar, Sanat, and B. Veytsman. “Mathematical Methods in Materials Science (MatSc597B)”. <http://www.plmsc.psu.edu/~www/matsc597>, 1996.
- Lamport, Leslie. *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System*. Addison-Wesley Publishing Company, Reading, MA, 2 edition, 1994. Illustrations by Duane Bibby.
- Lehmke, Stephan. *Package Tpslfonts: Configure Presentation Fonts*, 2004. <http://ctan.tug.org/tex-archive/macros/latex/contrib/tepower/tpslfonts>.
- Lehmke, Stephan, and H. F. Nordhaug. *The T<sub>E</sub>XPower Bundle*, 2005. <http://ctan.tug.org/tex-archive/macros/latex/contrib/tepower>.
- Mittelbach, Frank. *Producing Slides with L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*. August, 1997. <http://ctan.tug.org/tex-archive/macros/latex/base>.
- Muir, Star. “PowerPoint Use and Misuse: Strategies for Effective Teaching”. Lecture at George Mason University, 2006.
- Niepraschk, Rolf. *Ps4pdf Package*, 2004. <http://ctan.tug.org/tex-archive/macros/latex/contrib/ps4pdf>.
- Norvig, Peter. “The Gettysburg Powerpoint Presentation”. <http://www.norvig.com/Gettysburg>, 2000.
- Radhakrishnan, C. V. *Pdftscreen Manual*, 2000. <http://ctan.tug.org/tex-archive/macros/latex/contrib/pdftscreen>.
- Tantau, Till. *User’s Guide to the Beamer Class, Version 3.06*, 2005. <http://ctan.tug.org/tex-archive/macros/latex/contrib/beamer>.
- Tarasevich, Vsevolod. “Duel”. *Sovetskoe Foto (Soviet Photography)* (11), 1977. Available at <http://sov-photo.livejournal.com/35278.html>.
- Tufte, Edward R. *The Cognitive Style of Power Point*. Graphics Press LLC, Cheshire, CT, 2003.
- UK T<sub>E</sub>X Users Group. “UK List of T<sub>E</sub>X Frequently Asked Questions”. <http://www.tex.ac.uk/cgi-bin/texfaq2html>, 2006.
- Van Zandt, Timothy. *PSTricks: PostScript Macros for Generic T<sub>E</sub>X*, 1993. <http://ctan.tug.org/tex-archive/graphics/pstricks/obsolete/doc>.
- Veytsman, Boris. *Sectioning Commands in FoilT<sub>E</sub>X and Conversion to HTML Format: Foilhtml Package*, 1998. <http://ctan.tug.org/tex-archive/macros/latex/contrib/foilhtml>.
- Veytsman, Boris. “Introduction to Unix”. <http://users.lk.net/~borisv/unix>, 2001.
- Veytsman, Boris. “Closed Loop Phase Diagrams in Liquid Mixtures: From Theory to Simulations”. <http://users.lk.net/~borisv/20040902lecture.pdf>, 2004.
- Veytsman, Boris. “NANO 500: Introduction to Nanomaterials and Interactions”. <http://mason.gmu.edu/~bveytsma/nano500>, 2006.
- Veytsman, Boris, and M. Kotelyanskii. “Statistical Thermodynamics of Materials (MatSc597C)”. <http://www.plmsc.psu.edu/~www/matsc597c-1997>, 1997.
- Wall, Larry, T. Christiansen, and R. L. Schwartz. *Programming Perl*. O’Reilly & Associates, Inc., Bonn; Cambridge; Paris; Sebastopol; Tokyo, second edition, 1996.

# Automatic report generation with Web, T<sub>E</sub>X and SQL

Boris Veytsman

ITT, Advanced Engineering & Sciences  
12975 Worldgate Dr, Herndon, VA 20170  
boris dot veytsman (at) itt dot com

Maria Shmlevich

ITT, Advanced Engineering & Sciences  
12975 Worldgate Dr, Herndon, VA 20170, USA  
maria dot shmlevich (at) itt dot com

## Abstract

One of the most time-consuming tasks of a manager for a federal contractor is the creation of reports: weekly, monthly, quarterly and yearly as well as special reports at the end of a project or on any given date. Such reports are usually made by copying and pasting the daily reports of subordinates.

The system described here makes these reports automatically. The members of project team file their daily work results using a Web interface. These entries are kept in a SQL database. The report generation utility is launched through a Web interface. It creates a L<sup>A</sup>T<sub>E</sub>X file by selecting the data relevant to the given set of contracts and tasks, employees, time periods, etc., and collating the individual reports. The result is then run through `pdftex` or `latex2html` or `latex2rtf` to create either a PDF report or an editable (e.g., in Microsoft Word) file.

## 1 Introduction

In the last several decades applied science and technology in the US have seen unprecedented breakthroughs. Internet, GPS, space missions, a complete change in the civil aviation field and many advances in the military area are just a few examples of rapid technological progress. Of course there are many reasons for this, but it seems that one reason is the unique and fortunate method of technological cooperation between the government, universities and private contractors. In this scheme, the government agencies set the technological goals and solicit bids to achieve them. The winners of the bids get contracts for development of high-end technology with important military and civil uses.

Government agencies in this scheme are gate keepers of the people's money. They are obligated to control spending and check that the contracted research and development work is proceeding properly and the milestones are to be met on time. Therefore most agencies request detailed reports of the contractor's activity at regular intervals. These reports, however, pose the following problem. Obviously the taxpayer is interested only in the *results* of the contracted research and development. The time and money spent on the intermediate reports does

not contribute to the value and should be minimized. This is true both for the agency, which spends effort on the analysis of the reports, and the contractor, which spends effort on their preparation, and eventually passes the costs to the customer, thus increasing the total cost of the bid.

A report of high quality (including typographic quality!) is easier to analyze, so the report must be good. On the other hand, a good report might take a considerable effort to prepare. The goal is to make good reports with minimal effort and costs.

The traditional way of making intermediate reports is the following. The contractor's employees send e-mails to their managers describing their accomplishments. A manager copies and pastes these data into a Microsoft Word file and sends the file to the next level manager, who collates the received reports together. The task is repeated regularly, and each piece of information is copied and pasted several times: in weekly, monthly, quarterly and yearly reports. If the contract involves many tasks and subtasks, the work is overwhelming. This is unproductive work, since the real task of managers is management, not copying and pasting repetitive chunks of text.

Since most of this work is purely routine, it is

possible to teach a computer to do it, thus freeing engineers and managers for more creative tasks. This is the main idea of the system which was created at ITT in 2000–2001 and successfully used ever since.

## 2 Analysis

The first thing in the creation of a system for automatic report generation is to understand the structure of reports. A report is separated into *contracts*. The contracts are separated into *subcontracts*, and these are in turn separated in *tasks* and *subtasks*. Each individual report covers a subset of a hierarchy: it can include several contracts or just one contract, or several tasks from a subcontract, etc. It also covers a certain time period: week, month, quarter, year, etc.

The actual contents of the report are collated from the individual work by the engineers. Each of the engineers describes her or his work made during a particular week under each subtask, task, subcontract and contract. Sometimes a report includes the names of the engineers, and sometimes not, depending on the style chosen.

This structure is well suited for a SQL database. Each individual entry can be a record in the database, indexed by the subtask or task it belongs to, the engineer who made the entry, the time covered and the time it was made. The hierarchy “Contract-Subcontract-Task-Subtask” can easily fit into a SQL table with the usual parent-child relations. SQL operators can be used to extract from the tables the information that relates to the given task and time period.

We wanted the report to be available in several forms: a high quality PDF file as well as editable RTF and HTML formats. We chose L<sup>A</sup>T<sub>E</sub>X as the base format for the report because it can be used to produce beautiful PDF output, and the tools to transform it into HTML and RTF are widely available.

The interface to the software should be available from different computers: engineers’ and managers’ workstations. This makes an internal Web server a natural choice.

## 3 User interface

### 3.1 Authentication

A user (engineer or manager) logs in to the web server with her or his own user name and password. The database of logins and passwords is integrated with the system, so immediately after the user is authenticated, she or he is assigned a role (access level) in the system. There is a hierarchy of roles:

1. A normal user can input the information about his or her work into the database or correct it.
2. A manager can view the information and create reports, add or delete contracts and tasks.
3. An administrator can add or delete users, reset passwords and change access levels of the users.

Below we discuss these functions in more detail.

### 3.2 User access

A user should log in at least once a week and choose from the menu tasks and subtasks for which some work was performed by him or her. Then she or he inputs the work done under each category. There is an important option of choosing a special entry “Same as last week”; this will expand the time period of the entry of the previous week in the given category. The user can also set the priority of the tasks completed. The tasks with high priority are highlighted in the report.

### 3.3 Manager access

A manager can perform the functions of the user plus additional functions related to report creation and contracts and tasks changing.

A manager chooses from the menu the contracts, subcontracts, tasks and subtasks to cover, performance time, and report options: whether it should be in PDF, RTF or HTML format, whether it should include engineers’ names, etc. The report is created and a link for download is presented to the manager.

A manager can add or delete contracts, change subcontracts, tasks and subtasks. This will update the menus presented to all users.

### 3.4 Administrator access

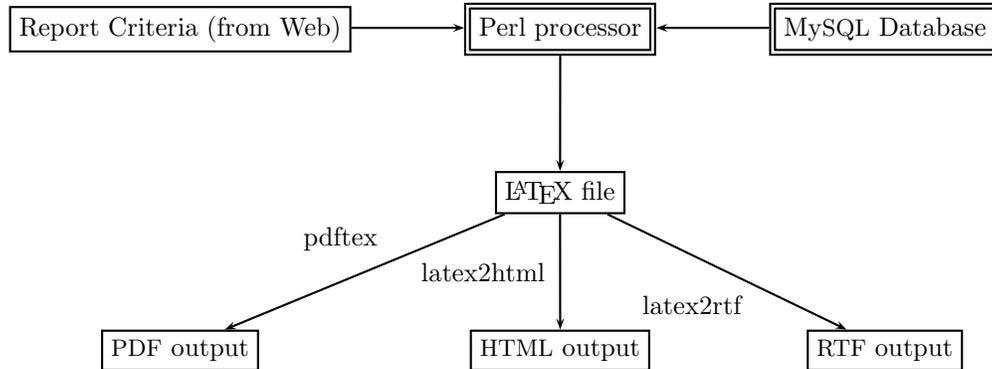
An administrator can change the information about users. She or he is presented with a menu, which includes changing of user personal information (name, e-mail), resetting passwords, changing access levels, etc.

### 3.5 Additional bells and whistles

The system generates reminders for the users to log in and enter their information, and sends lists of the procrastinators to the managers. It also generates periodic backup dumps of its databases.

## 4 Implementation notes

The system is implemented on a Linux computer using the Apache Web server, MySQL database, sendmail, t<sub>E</sub>X suite and latex2html and latex2rtf programs. It is essentially a zero administration server: since it was set up, only security patches have been



**Figure 1:** Report creation

applied to the machine, and everything else “just works”.

The flowchart for the report creation is shown in Figure 1. The Perl program extracts from the database the entries satisfying the selected criteria. They are collated into a L<sup>A</sup>T<sub>E</sub>X file. The hierarchy “Contract-Subcontract-Task-Subtask” is mapped to the hierarchy “Chapter-Section-Subsection-Subsubsection” of the `report` document class. The entries themselves are organized into itemized lists.

An example of the result is shown on Figure 2. In this example the engineer (Archimedes) started the work of moving Earth using a lever. His completed tasks include a high-priority development of the background and low-priority work on geometry.

## 5 Conclusions

We developed a system to perform an important and time-consuming task of generating periodic reports by a federal contractor. The system is based on open and free software. It provides a very efficient and cost-effective solution, which has been successfully working for half a decade.

## Contract 1

### Moving Earth With Lever (ERTHMV)

#### 1.1 Finding a place to stand

#### 1.2 Mathematical Background

##### 1.2.1 Development of series summation

- 12/23/0282: **Developed background** Archimedes
- 12/30/0282: **Worked on geometry** Archimedes

#### 1.3 Create a lever

2

**Figure 2:** Example of a page from a report

# Writing and checking complete proofs in $\text{\TeX}$

Bob Neveln  
Widener University

Bob Alps  
Towers Perrin  
Chicago, Illinois

## Abstract

$\text{\TeX}$  files are text files which are readable by other programs. Mathematical proofs written using  $\text{\TeX}$  can be checked by a Python program provided they are expressed in a sufficiently strict proof language. Such a language can be constructed using only a few extensions beyond the syntax of A.P. Morse's *A Theory of Sets*, one being the incorporation of explicit theorem number references into the syntax. Such a program has been applied to and successfully checked the theorems in a significant initial segment of a book length mathematical manuscript.

## 1 Introduction

The present work is an unplanned side-effect of a book project by the authors [7]. As work on the book progressed proofs were written more and more carefully. Programs in Python were developed to check the mathematical syntax, then to re-number theorems following insertions or deletions and finally to check the proofs written.

These developments were possible because the book is written in  $\text{\TeX}$  using a formal mathematical language. Although most mathematical text is intended to be formalizable, usually in terms of first order predicate logic, it is almost never formal as written. Checking proofs written in a conventional style would consequently require a formalization step requiring clarification of the author's intentions on many details. Checking proofs written in a formal language obviates these difficulties. In the work presented here we use a syntax derived from that of A. P. Morse. In his book, *A Theory of Sets* [5], he presented a formal syntax which was used to express all the definitions and theorems in his book, see [6]. A key feature of his treatment of mathematical language was the inclusion of definitions themselves into the formal syntax, see [1]. The first theorem of the book was given a complete proof, but no attempt was made to continue the presentation of complete proofs. Indeed with the small set of inference rules given this would not have been feasible.

The formal syntax of Morse's book enabled the creation of a program capable of parsing its language and checking some of its theorems using an expanded inference rule set as early as 1966 at Sandia Laboratories [3]. Soon after that most of the math-

ematics in the book was checked by W.W. Bledsoe working at MIT.

This paper describes some additions to Morse's syntax implemented in  $\text{\TeX}$  and Python programs which together enable writing and checking complete proofs. The resulting environment is a work in progress.

## 2 Tools and Files

Unix utilities are based on the idea that it is good to have many tools each of which does a single task well. Along those lines, the environment described here to enable writing and checking complete proofs consists of many different  $\text{\TeX}$  files and Python programs. As Richter noted in [8], it is easy to write Python scripts which conveniently operate on  $\text{\TeX}$  files. Those described here include a program which checks the syntax of the mathematics in the  $\text{\TeX}$  file, a program which rennumbers the propositions, a program which adds horizontal space to variable scope clauses in  $\text{\TeX}$  files, in addition to the program which checks a proof whose number is given as a command line argument.

The logic on which proofs depend is supplied in a variety of ways. Some logic is built into the parser; for example,  $(x < y < z)$  is parsed as  $(x < y \wedge y < z)$ . Some logic is built into the checking program which uses the commutative and associative properties of "and" as well as the transitivity of numerous relations including logical implication and set inclusion. Most of the logic resides in a file of rules of inference which is consulted in a blind linear search each time a step of the proof to be checked is attempted. Another file consists of propositions which are generally recognized as obvious such as

$$(x \in A \wedge A \subset B \rightarrow x \in B)$$

Further logic consisting of material which is at least as elementary, but ordinarily “below the radar” of everyday mathematics, is listed in a special appendix added to the work being checked. It uses the logic developed in [2].

Our tools create an environment which sets in motion a work cycle related to an ongoing paper or book, consisting of steps like those involved in writing a computer program:

1. Add or revise the statement or proof of a theorem in a T<sub>E</sub>X source file.
2. Run T<sub>E</sub>X to get a viewable DVI file and detect T<sub>E</sub>X errors.
3. Run the parser to find mathematical syntax errors.
4. Run the check program to find logical errors and gaps in the proof.

At the very end it is also useful to run a program which uses the parser to add horizontal spacing at points where T<sub>E</sub>X would otherwise crowd symbols.

Because the logical steps which can be checked at this time are quite small, the process is both arduous and tedious.

### 3 Proof Syntax

The basis for the proof syntax is the mathematical syntax of Tony Morse’s book [5]. Changes to Morse’s mathematical syntax including additional abbreviation schemes and restrictions on the format of bound variable forms are introduced but do not alter the mathematical language markedly. To get a notation capable of expressing complete proofs just a few additional elements suffice.

#### 3.1 Reference Numbering

An important element in the proof syntax described here is the inclusion of theorem numbers themselves into the syntax.

An example from the manuscript [7] follows:

```
\tabc 1.17 $(b\in\bfun \Iff \Patch_0 b\in\U)$
\lineb Proof:
\notea 1$(b \in \bfun$
\linec $\c\Patch_0 b\in \SI
        \rng b\setdif\dmn b$\By 1.16
\linec $\c\Patch_0 b \in \U)$ \By 01.14
\lineb
\notea 2$(\Patch_0 b\in\U$
\linec$\c\ex\Patch_0 b$ \By 01.8
\linec$\c b\in\bfun)$\By 1.13
\lineb \Bye .1, .2
\lineb
```

In this example a theorem numbered 1.17 is stated and proved. The statement involves the plain T<sub>E</sub>X macro ‘\in’ as well as other macros such as ‘\c’ for

‘\rightarrow’ and ‘\Iff’ for ‘\leftrightharrow’. Using Morse style mathematical language in T<sub>E</sub>X involves a large number of such macros, basically at least one for each defined formula as well as some special symbols which can be implemented in Metafont. The ‘\tabc’, ‘\notea’ and ‘\By’ macros perform space formatting, but also serve as reference handles for the checking program. For example Theorem 1.16, which is referred to at the end of the second line of the proof, must be identified by a ‘\tabc’ macro. The ‘\lineb’ macro has only a space formatting role. The ‘\Bye’ macro prints QED and indicates that the theorem itself is to be checked.

References such as the closing ‘.1’ and ‘.2’ refer to the notes tagged by the ‘\notea’ macros. The zero-plus references 01.14 and 01.8 point to the file of “obvious” theorems.

Propositions which are referenced must have a traditional number-dot-number identification which is used to invoke them in proofs. This numbering convention is similar to that produced by L<sup>A</sup>T<sub>E</sub>X but less flexible. It is used instead of L<sup>A</sup>T<sub>E</sub>X because its use requires slightly less labor and the labor involved in specifying references is a large component of the work of specifying a complete proof. A Python program is needed to renumber all references when theorems are inserted, deleted, or moved.

#### 3.2 Significant Punctuation

Reference notations may include punctuation. The punctuation marks must be identical to the corresponding marks in the rule of inference itself. If rules are marked in such a way that rules of a similar nature get similar punctuation, then a meaning is associated with the punctuation mark.

For instance, the semi-colon is used in references that have a major premise followed by minor premises. As an example, if in note 5 below we prove a result  $q$  by using a theorem ( $p \rightarrow q$ ) which is numbered 1.23 and we have previously obtained  $p$  in note 3 then we might have the following note to establish  $q$ :

Note 5 ( $-a \in Z$ )                    † 1.23; .3

In order for this note to be checked there must be a theorem 1.23 such as

Thm 1.23 ( $x \in Z \rightarrow -x \in Z$ )

a previous note 3 like this

Note 3 ( $a \in Z$ )

as well as a rule of inference (modus ponens) which has the form

From: ( $p \rightarrow q$ );  $p$

Infer:  $q$

The semi-colon in the reference limits the number of rules which match that reference. The intended meaning of the semi-colon is that it sets the “major premise” apart from the “minor premises”. At present approximately 250 of the stored inference rules use the semi-colon to separate major and minor premises. Another example of such a rule is the following rule:

From:  $(p \rightarrow q \leftrightarrow r); q$   
 Infer:  $(p \rightarrow r)$

Further developments towards a syntax of reference expressions will no doubt be found useful.

### 3.3 Given-Hence Blocks

Notes which are not proven but which merely state a “given” may be justified using  $\ddagger G$ , in place of a proof reference. These remain in force until a “hence” referring to them is encountered. The “hence” attaches the given notes to the “henced note” as explicit hypotheses. The “hence” note is tagged using  $\ddagger H$  as a proof reference. For example we might have:

Note 2  $(x \in A)$   $\ddagger G$   
 ...  
 Note 7  $(x \in B)$   $\ddagger .2, \dots$   
 Note 8  $(x \in A \rightarrow x \in B)$   $\ddagger .7 H .2$

The variables introduced in each given note are local to that block. Reference may be made to notes 2–7 only from within that block, only so long as note 2 is in force in other words.

### 3.4 Local Definitions

Sometimes it is useful to introduce locally defined variables. To do this we may “set” a variable to a described object. A note of this form is justified by  $\ddagger S$  and it retains validity as long as the last preceding given note. For example given a non-empty set  $A$  it is useful to have a name for a member of  $A$ .

Note 2  $(A \neq \emptyset)$   $\ddagger G$   
 Note 3  $(a \equiv \text{anx}(x \in A))$   $\ddagger S$   
 Note 4  $(a \in A)$   $\ddagger .2, .3$

This feature of the proof syntax depends on using a logic which allows descriptions, see [2].

### 3.5 Reasoning Chains

A note may consist of lines all but the first of which are introduced by some transitive relation. In this case each pair of consecutive lines defines a step to be checked on its own proof. When used as a reference the note is then telescoped. For example in this note:

Note 7  $(A \subset B$   $\ddagger \dots$   
 $\quad \quad \subset C)$   $\ddagger \dots$

the inclusions  $(A \subset B)$  and  $(B \subset C)$  are checked separately, but if note 7 is referred to later, just the inclusion  $(A \subset C)$  will be invoked by this reference.

## 4 The Unifier

Each step to be checked is matched against rules of inference in a blind linear search. Each rule whose sequence of arguments and punctuators matches with numerical references and punctuators in the reference note is submitted to a unifier. If a unification is found the step is checked.

The unifier is based on standard first order unification, but goes beyond this in two ways. Although much less general than [4], it allows the terms of a conjunction to be re-ordered in order to accomplish a match. It also attempts to match the second order variables which occur in Morse’s language.

It is written to succeed or fail quickly. It may fail to find a unifier even when one exists. For example if a conjunction with  $n$  conjuncts is matched against a conjunction  $(p \wedge q)$ , where ‘ $p$ ’ and ‘ $q$ ’ are unmatched variables, this unification will not be attempted because of the  $(2^n - 2)$  different possible matchings. A rule of inference must avoid presenting such unifications to the checker or it will be ignored. The unifier does not aim at any ambitious sort of completeness.

## 5 Results and Prospects

The manuscript being checked contains over 1200 theorems, with proofs in various stages of completion. Roughly 250 of these including the first 120 have been checked.

As the work proceeds, bugs are encountered in the checking program, as well as cases which should check but do not. The program is then revised, rules of inference are added, and “obvious” theorems are added to the zero-plus references file. There are now over 700 rules of inference and over 500 theorems in the zero-plus references file. The checking program now contains about 4500 lines of code. The manuscript also has appendices containing over 200 elementary results which can be referenced in the proofs.

Each execution of the program checks a single proof. Although Python is an interpreted language, a few seconds suffices for one run of the program on a machine of recent vintage.

The proof syntax at its present stage of development is and should be “low-level”. Once avenues of checkable proof begin flowing it will be time for the appearance of higher levels of expression which will attenuate to some extent the labor of picking through all the details of a proof.

## 6 Observations

We close with a few observations.

1. Including the details necessary to get a proof to check requires roughly an order of magnitude more time than writing a conventional proof.
2. Proofs stated in checkable detail become longer by a factor less than an order of magnitude.
3. Reading checkable proofs requires slightly more effort on the part of a reader with specialized knowledge than proofs which are written with such a reader in mind.
4. Checkable proofs can be read by any mathematician whether a specialist or not.

## 7 Conclusion

Despite its preliminary and incomplete nature the checking program as it stands now shows that it is practicable to write and check complete proofs, given a willingness to adopt a formal language and to submit to the discipline of itemizing all necessary references.

## References

- [1] R.A. Alps. *A Translation Algorithm for Morse Systems*, PhD dissertation, Northwestern University, 1979.
- [2] R.A. Alps and R.C. Neveln, A Predicate Logic Based on Indefinite Description and Two Notions of Identity. *Notre Dame Journal of Formal Logic* 22(3), 251–263, 1981.
- [3] W.W. Bledsoe and E.J. Gilbert. *Automatic Theorem Proof-Checking in Set Theory*, Sandia Laboratories Research Report SC-RR-67-525, July 1967.
- [4] J. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Theoretical Computer Science*, 67:203–260, 1989.
- [5] A.P. Morse. *A Theory of Sets*, Second Edition. Academic Press, 1986.
- [6] R.C. Neveln. *Basic Theory of Morse Languages*, PhD dissertation, Northwestern University, 1975.
- [7] Bob Neveln and Bob Alps. *Foundations of the Topology of Manifolds* (book in preparation).
- [8] William Richter.  $\text{\TeX}$  and Scripting Languages. *TUGboat* 25(1), 71–88, 2004.

# A beginner's guide to METAPOST for creating high-quality graphics

Troy Henderson

Department of Mathematical Sciences

United States Military Academy

West Point, NY 10996, USA

troy (at) tlhiv dot org

<http://www.tlhiv.org>

## Abstract

Individuals that use  $\text{\TeX}$  (or any of its derivatives) to typeset their documents generally take extra measures to ensure paramount visual quality. Such documents often contain mathematical expressions and graphics to accompany the text. Since  $\text{\TeX}$  was designed “for the creation of beautiful books — and especially for books that contain a lot of mathematics” [4], it is clear that it is sufficient (and in fact *exceptional*) at dealing with mathematics and text.  $\text{\TeX}$  was not designed for creating graphics; however, certain add-on packages can be used to create modest figures.  $\text{\TeX}$ , however, is capable of including graphics created with other utilities in a variety of formats. Because of their scalability, Encapsulated PostScript (EPS) graphics are the most common types used. This paper introduces METAPOST and demonstrates the fundamentals needed to generate high-quality EPS graphics for inclusion into  $\text{\TeX}$ -based documents.

## 1 Introduction

To accompany  $\text{\TeX}$ , Knuth developed METAFONT as a method of “creating entire families of fonts from a set of dimensional parameters and outline descriptions” [1]. Approximately ten years later, John Hobby began work on METAPOST — “a powerful graphics language based on Knuth’s METAFONT, but with PostScript output and facilities for including typeset text” [3]. Although several packages (e.g.,  $\text{\PictEX}$ ,  $\text{\Xypic}$ , and the native  $\text{\LaTeX}$  picture environment to name a few) are available for creating graphics within  $\text{\TeX}$ -based documents, they all rely on  $\text{\TeX}$ . Since  $\text{\TeX}$  was designed to typeset text, it seems natural that an external utility should be used to generate graphics instead. Furthermore, in the event that the graphics require typeset text, then the utility should use  $\text{\TeX}$  for this requirement. This premise is exactly the philosophy of METAPOST.

Since METAPOST is a programming language, it accommodates data structures and flow control, and compilation of the METAPOST source code yields EPS graphics. These features provide an elegant method for generating graphics. Figure 1 illustrates how METAPOST can be used programat-

ically. The figure is generated by rotating one of the circles multiple times to obtain the desired *circular chain*. The programming language constructs

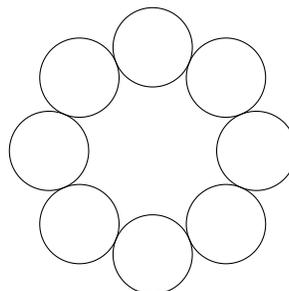


Figure 1: Rotated circles

of METAPOST also deliver a graceful mechanism for creating animations without having to manually create each frame of the animation. The primary advantage of EPS is that it can be scaled to any resolution without a loss in quality. It can also be easily converted to raster formats, e.g. Portable Network Graphics (PNG) and Joint Photographic Experts Group (JPEG), et al., or other vector formats including Portable Document Format (PDF) and Scalable Vector Graphics (SVG), et al.

---

<sup>1</sup> All graphics in this article (except Figure 2) are created with METAPOST, and the source code and any required external data files for each of these graphics are embedded as file attachments in the electronic PDF version of the article.

# METAPOST Previewer

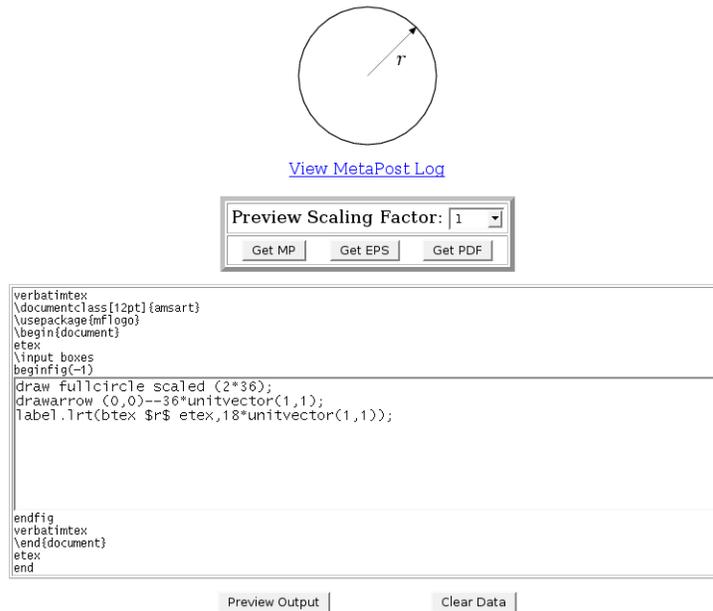


Figure 2: METAPOST Previewer

## 2 METAPOST compilation

A typical METAPOST source file consists of one or more figures. Compilation of the source file generates an EPS graphic for each figure. These EPS graphics are not self-contained in that fonts used in labels are not embedded into the graphic.

If `foo.mp` is a typical METAPOST source file, then its contents are of the following form:

```
beginfig(1);
    draw commands
endfig;
beginfig(2);
    draw commands
endfig;
...
beginfig(n);
    draw commands
endfig;
end;
```

Executing

```
mpost foo.mp
```

yields the following output:

```
This is MetaPost, Version {version}
(foo.mp [1] [2] ... [n] )
n output files written: foo.1 .. foo.n
Transcript written on foo.log.
```

For users who just want to “get started” using METAPOST, a METAPOST previewer is available at <http://www.tlhiv.org/MetaPostPreviewer>. This previewer (illustrated in Figure 2) is simply a graphical interface to METAPOST itself. It generates a single graphic with the option to save the output in both EPS and PDF formats. Users may also choose to save the source code and can view the compilation log to assist in debugging.

## 3 Data types

There are nine data types in METAPOST: *numeric*, *pair*, *path*, *transform*, *color*, *string*, *boolean*, *picture*, and *pen*. These data types allow users to store fragments of the graphics for later use. We will briefly discuss each of these data types and elaborate on how they are used in a typical METAPOST program.

- ◇ *numeric* — numbers
- ◇ *pair* — ordered pairs of numerics
- ◇ *path* — Bézier curves (and lines)
- ◇ *picture* — pictures
- ◇ *transform* — transformations such as shifts, rotations, and slants
- ◇ *color* — triplets in the unit cube with red, green, and blue (RGB) components
- ◇ *string* — strings to be labeled
- ◇ *boolean* — “true” or “false” values
- ◇ *pen* — stroke properties

Virtually all programming languages provide a way of storing and retrieving numerical values. This is precisely the purpose of the *numeric* data type in METAPOST. Since graphics drawn with METAPOST are simply two dimensional pictures, it is clear that an ordered pair is needed to identify each point in the picture. The *pair* data type provides this functionality. Each point in the plane consists of an  $x$  (i.e., abscissa) part and a  $y$  (i.e., ordinate) part. METAPOST uses the standard syntax for defining points in the plane, e.g.,  $(x, y)$  where both  $x$  and  $y$  are numeric data typed variables.

In order to store paths between points, the *path* data type is used. All paths in METAPOST are represented as cubic Bézier curves. Cubic Bézier curves are simply parametric splines of the form  $(x(t), y(t))$  where both  $x(t)$  and  $y(t)$  are piecewise cubic polynomials of a common parameter  $t$ . Since Bézier curves are splines, they pairwise interpolate the points. Furthermore, cubic Bézier curves are diverse enough to provide a “smooth” path between all of the points for which it interpolates. METAPOST provides several methods for affecting the Bézier curve between a list of points. For example, piecewise linear paths (i.e., linear splines) can be drawn between a list of points since all linear polynomials are also cubic polynomials. Furthermore, if a specific direction for the path is desired at a given point, this constraint can be forced on the Bézier curve.

The *picture* data type is used to store an entire picture for later use. For example, in order to create animations, usually there are objects that remain the same throughout each frame of the animation. So that these objects do not have to be manually drawn for each frame, a convenient method for re-drawing them is to store them into a picture variable for later use.

When constructing pairs, paths, or pictures in METAPOST, it is often convenient to apply affine transformations to these objects. As mentioned above, Figure 1 can be constructed by rotating the same circle several times before drawing it. METAPOST provides built-in affine transformations as “building blocks” from which other transformations can be constructed. These include shifts, rotations, horizontal and vertical scalings, and slantings.

There are five built-in colors in METAPOST: **black**, **white**, **red**, **green**, and **blue**. However, custom colors can be defined using the *color* data type. Colors in METAPOST are simply ordered triplets of the form  $(r, g, b)$  where  $r$ ,  $g$ , and  $b$  are numerics between 0 and 1. These values  $r$ ,  $g$ , and  $b$  identify what fraction of the color is red, green, and blue, respectively. For example, the built-in color **red** is

simply a synonym for  $(1, 0, 0)$  and **black** is a synonym for  $(0, 0, 0)$ . If a particular color is to be used several times throughout a figure, it is natural to store this color into a variable (of type *color*) for multiple uses.

The most common application of *string* data types is reusing a particular string that is typeset (or labeled). The *boolean* data type is the same as in other programming languages, used in conditional statements for testing. Finally, the *pen* data type is used to affect the actual stroke paths. The default unit of measurement in METAPOST is  $1\text{ bp} = 1/72\text{ in}$ , and the default thickness of all stroked paths is  $0.5\text{ bp}$ . An example for using the *pen* data type may include changing the thickness of several stroked paths. This new pen can be stored and then referenced for drawing each of the paths.

## 4 Common commands

The METAPOST manual [3] lists 26 built-in commands along with 23 function-like macros for which pictures can be drawn and manipulated using METAPOST. We will not discuss each of these commands here; however, we will focus on several of the most common commands and provide examples of their usage.

### 4.1 The draw command

The most common command in METAPOST is the **draw** command. This command is used to draw paths or pictures. In order to draw a path from  $\mathbf{z1}=(0,0)$  to  $\mathbf{z2}=(54,18)$  to  $\mathbf{z3}=(72,72)$ , we should first decide how we want the path to look. For example, if we want these points to simply be connected by line segments, then we use

```
draw z1--z2--z3;
```

However, if we want a smooth path between these points, we use

```
draw z1..z2..z3;
```

In order to specify the direction of the path at the points, we use the **dir** operator. In Figure 3 we see that the smooth path is horizontal at  $\mathbf{z1}$ , a  $45^\circ$  angle at  $\mathbf{z2}$ , and vertical at  $\mathbf{z3}$ . These constraints on the Bézier curve are imposed by

```
draw z1{right}..z2{dir 45}..{up}z3;
```

Notice that  $\mathbf{z2}\{\text{dir } 45\}$  forces the *outgoing* direction at  $\mathbf{z2}$  to be  $45^\circ$ . This implies an *incoming* direction at  $\mathbf{z2}$  of  $45^\circ$ . In order to require different incoming and outgoing directions, we would use

```
draw z1{right}..{dir  $\theta_i$ }z2{dir  $\theta_o$ }..{up}z3;
```

where  $\theta_i$  and  $\theta_o$  are the incoming and outgoing directions, respectively.

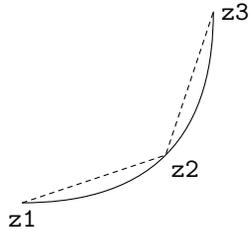


Figure 3: draw examples

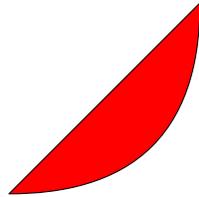


Figure 4: fill example

#### 4.2 The fill Command

Another common command in METAPOST is the `fill` command. This is used to fill closed paths (or cycles). In order to construct a cycle, `cycle` may be appended to the path declaration. For example,

```
path p;
p:=z1{right}..z2{dir 45}..{up}z3--cycle;
fill p withcolor red;
draw p;
```

produces Figure 4. Notice that `p` is essentially the same curved path as in Figure 3 with the additional piece that connects `z3` back to `z1` with a line segment using `--cycle`.

Just as it is necessary to fill closed paths, it may also be necessary to *unfill* closed paths. For example, the annulus in Figure 5 can be constructed by

```
color bbblue;
bbblue:=(3/5,4/5,1);
path p,q;
p:=fullcircle scaled (2*54);
q:=fullcircle scaled (2*27);
fill p withcolor bbblue;
unfill q;
draw p;
draw q;
```

The `fullcircle` path is a built-in path that closely approximates a circle in METAPOST with diameter 1 bp traversed counter-clockwise. This path is not exactly a circle since it is parameterized by a Bézier curve and not by trigonometric functions; however, visually it is essentially indistinguishable from an

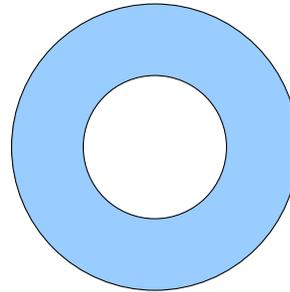


Figure 5: unfill example

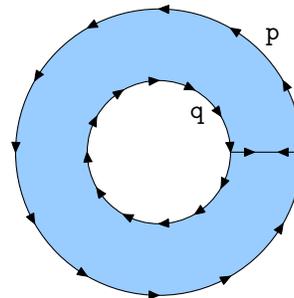


Figure 6: Avoiding an unfill

exact circle. Notice that `p` is a `fullcircle` of radius 54 bp (3/4 in) and `q` is a `fullcircle` of radius 27 bp (3/8 in). The annulus is constructed by filling `p` with the baby blue color `bbblue` and then unfilling `q`. The `unfill` command above is equivalent to

```
fill q withcolor background;
```

where `background` is a built-in color which is `white` by default.

Often the `unfill` command appears to be the natural method for constructing figures like Figure 5. However, the `fill` and `unfill` commands in Figure 5 can be replaced by

```
fill p--reverse q--cycle withcolor bbblue;
```

The path `p--reverse q--cycle` travels around `p` in a counter-clockwise directions (since this is the direction that `p` traverses) followed by a line segment to connect to `q`. It then traverses clockwise around `q` (using the `reverse` operator) and finally returns to the starting point along a line segment using `--cycle`. This path is illustrated in Figure 6. One reason for using this method to construct the annulus as opposed to the `unfill` command is to ensure *proper transparency* when placing the figure in an external document with a non-white background. If the former method is used and the annulus is placed on a non-white background, say magenta, then the result is Figure 7. It may be desired to have the interior of `q` be magenta instead

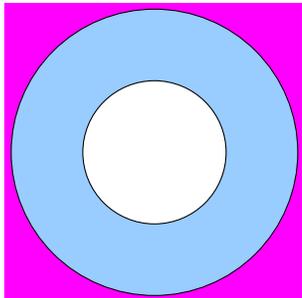


Figure 7: Improper transparency using unfill

of white. This could be accomplished by redefining `background`; however, the latter method described above is a much simpler solution.

### 4.3 Arrow commands

When drawing simple graphs and other illustrations, the use of arrows is often essential. There are two arrow commands in METAPOST for accommodating this need — `drawarrow` and `drawdblarrow`. Both of these commands require a path argument. For example,

```
drawarrow (0,0)--(72,72);
```

draws an arrow beginning at  $(0,0)$  and ending at  $(72,72)$  along the line segment connecting these points.

The path argument of both `drawarrow` and `drawdblarrow` need not be line segmented paths — they may be any METAPOST path. The only difference between `drawarrow` and `drawdblarrow` is that `drawarrow` places an arrow head at the end of the path and `drawdblarrow` places an arrow head at the beginning and the end of the path. As an example, to draw the curved path in Figure 3 with an arrow head at the end of the path (i.e., at `z3`), the following command can be used

```
drawarrow z1{right}..z2{dir 45}..{up}z3;
```

and is illustrated in Figure 8.

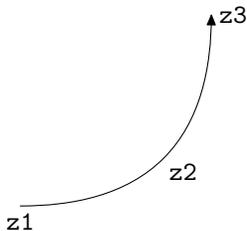


Figure 8: Using `drawarrow` along a path

### 4.4 The label command

One of the nicest features of METAPOST is that it relies on T<sub>E</sub>X (or L<sup>A</sup>T<sub>E</sub>X) to typeset labels within figures. Almost all figures in technical documents are accompanied by labels which help clarify the situation for which the figure is assisting to illustrate. Such labels may include anything from simple typesetting as in Figures 3, 6, and 8 to typesetting function declarations and even axes labeling.

The `label` command requires two arguments — a string to typeset and the point for which label is placed. For example, the command

```
label("A", (0,0));
```

will place the letter “A” at the coordinate  $(0,0)$  and the box around this label is centered vertically and horizontally at this point. Simple strings like “A” require no real typesetting to ensure that they appear properly in the figure. However, many typeset strings in technical figures require the assistance of T<sub>E</sub>X to properly display them. For example, Fig-

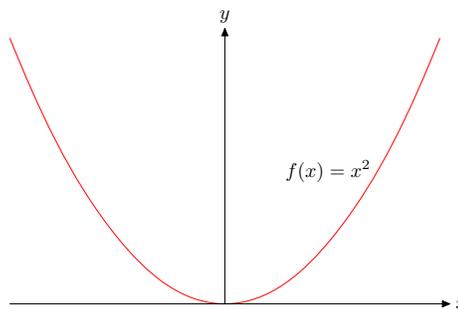


Figure 9: Labeling text

ure 9 is an example where typesetting is preferred. That is, the axes labels and the function declaration look less than perfect if T<sub>E</sub>X is not used. For reasons such as this, METAPOST provides a way to *escape* to T<sub>E</sub>X in order to assist in typesetting the labels. Therefore, instead of labeling the “A” as above,

```
label(btex A etex, (0,0));
```

provides a much nicer technique for typesetting the label. The `btex...etex` block instructs METAPOST to process everything in between `btex` and `etex` using T<sub>E</sub>X. Therefore, the function declaration in Figure 9 is labeled using

```
label(btex $f(x)=x^2$ etex, (a,b));
```

where  $(a,b)$  is the point for which the label is to be centered.

Since many METAPOST users prefer to typeset their labels using L<sup>A</sup>T<sub>E</sub>X instead of plain T<sub>E</sub>X, METAPOST provides a convenient method for accommodating this, done in the preamble of the METAPOST source file. The following code ensures that

the `btex...etex` block escapes to L<sup>A</sup>T<sub>E</sub>X (instead of plain T<sub>E</sub>X) for text processing.

```

verbatim
%&latex
\documentclass{minimal}
\begin{document}
etex
beginfig(n);
  <draw commands>
endfig;
end

```

Often times it is desirable to typeset labels with a justification that is not centered. For example, one may wish to place an “A” not centered horizontally about (0,0) but placed above (0,0). METAPOST provides eight suffixes to accommodate such needs. The suffixes `.lft`, `.rt`, `.bot`, and `.top` align the label on the left, right, bottom, and top, respectively, of the designated point. A hybrid of these four justifications provide four additional ones, namely, `.llft`, `.ulft`, `.lrt`, and `.urt` to align the label on the lower left, upper left, lower right, and upper right, respectively, of the designated point. For example,

```
label.top(btex A etex,(0,0));
```

places the “A” directly above (0,0). Figure 10 demonstrates each of the suffixes and their corresponding placement of the labels.

top		ulft	urt
lft	□	llft	✕
rt	□	lft	lft
bot	□		

Figure 10: Label suffixes

## 5 Graphing functions

Among the most common types of figures for T<sub>E</sub>X users are those which are the graphs of functions of a single variable. Hobby recognized this and constructed a package to accomplish this task. It is invoked by

```
input graph;
```

METAPOST has the ability to construct data (i.e., ordered pairs) for graphing simple functions. However, for more complicated functions, the data should probably be constructed using external programs such as MATLAB (or Octave), Maple, Mathematica, Gnuplot, et. al.

A typical data file, say `data.d`, to be used with the `graph` package may have contents

```

0.0  0.0
0.2  0.447214
0.4  0.632456
0.6  0.774597
0.8  0.894427
1.0  1.0

```

This data represents the graph of  $f(x) = \sqrt{x}$  for six equally spaced points in  $[0, 1]$ . To graph this data, the size of the graph must first be decided. Choosing a width of 144 bp and a height of 89 bp, a minimally controlled plot (as in Figure 11) of this data can be generated by

```

draw begingraph(144bp,89bp);
  gdraw "data.d";
endgraph;

```

The `graph` package provides many commands used to customize generated graphs, and these commands are fully documented in the manual [2] for the `graph` package.

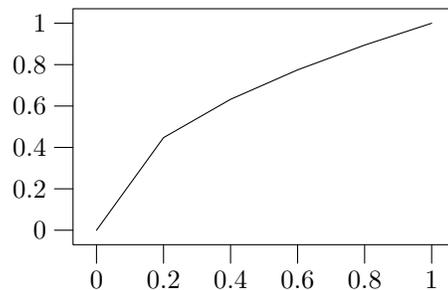


Figure 11:  $f(x) = \sqrt{x}$  using the `graph` package

## 6 Including METAPOST figures in L<sup>A</sup>T<sub>E</sub>X

In order to include a METAPOST figure in L<sup>A</sup>T<sub>E</sub>X, the `graphicx` package is suggested. Below is an example of including a METAPOST figure (with name `foo.1`) in a L<sup>A</sup>T<sub>E</sub>X document.

```

\documentclass{article}
\usepackage{graphicx}
\usepackage{ifpdf}
\ifpdf
  \DeclareGraphicsRule{*}{mps}{*}{}
\fi
\begin{document}
...
\includegraphics{foo.1}
...
\end{document}

```

The `ifpdf` package and `\ifpdf...\fi` command is used to prompt PDF $\LaTeX$  to convert the METAPOST graphic to PDF “on the fly” using Hans Hagen’s `mptopdf`. This conversion is necessary since PDF $\LaTeX$  performs no PostScript processing.

## 7 Conclusion

METAPOST is an elegant programming language, and it produces beautiful graphics. The graphics are vectorial and thus can be scaled to any resolution without degradation. There are many advanced topics that are not discussed in this article (e.g., loops, flow control, subpaths, intersections, etc.), and the METAPOST manual [3] is an excellent resource for these advanced topics. However, the METAPOST manual may seem daunting for beginners. There are many websites containing METAPOST examples, and several of these are referenced at <http://www.tug.org/metapost>. Finally, we mention that Knuth uses nothing but METAPOST for his diagrams.

## References

- [1] N. H. F. Beebe. Metafont. <http://www.math.utah.edu/~beebe/fonts/metafont.html>, 2006.
- [2] 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>.
- [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. *The  $\TeX$ book*, volume A of *Computers and Typesetting*. Addison Wesley, Boston, 1986.

# Graphics with PGF and TikZ

Andrew Mertz, William Slough

Department of Mathematics and Computer Science

Eastern Illinois University

Charleston, IL 61920

aemertz (at) eiu dot edu, waslough (at) eiu dot edu

## Abstract

Beautiful and expressive documents often require beautiful and expressive graphics. PGF and its front-end TikZ walk a fine line between power, portability and usability, giving a TeX-like approach to graphics. While PGF and TikZ are extensively documented, first-time users may prefer learning about these packages using a collection of graduated examples. The examples presented here cover a wide spectrum of use and provide a starting point for exploration.

## 1 Introduction

Users of TeX and L<sup>A</sup>TeX intending to create and use graphics within their documents have a multitude of choices. For example, the UK TeX FAQ [1] lists a half dozen systems in its response to “Drawing with TeX”. One of these systems is PGF and its associated front-end, TikZ [4].

All of these systems have similar goals: namely, to provide a language-based approach which allows for the creation of graphics which blend well with TeX and L<sup>A</sup>TeX documents. This approach stands in contrast to the use of an external drawing program, whose output is subsequently included in the document using the technique of graphics inclusion.

PGF provides a collection of low-level graphics primitives whereas TikZ is a high-level user interface. Our intent is to provide an overview of the capabilities of TikZ and to convey a sense of both its power and relative simplicity. The examples used here have been developed with Version 1.0 of TikZ.

## 2 The name of the game

Users of TeX are accustomed to acronyms; both PGF and TikZ follow in this tradition. PGF refers to **P**ortable **G**raphics **F**ormat. In a tip of the hat to the recursive acronym GNU (i.e., GNU’s not Unix), TikZ stands for “**T**ikZ **i**st **k**ein **Z**eichenprogramm”, a reminder that TikZ is not an interactive drawing program.

## 3 Getting started

TikZ supports both plain TeX and L<sup>A</sup>TeX input formats and is capable of producing PDF, PostScript, and SVG outputs. However, we limit our discussion to one choice: L<sup>A</sup>TeX input, with PDF output, processed by pdfL<sup>A</sup>TeX.

TikZ provides a one-step approach to adding

graphics to a L<sup>A</sup>TeX document. TikZ commands which describe the desired graphics are simply intermingled with the text. Processing the input source yields the PDF output.

Figure 1 illustrates the layout required for a document which contains TikZ-generated graphics. Of central interest is the `tikzpicture` environment, which is used to specify one graphic. Within the preamble, the `tikz` package must be specified, along with optional PGF-based libraries. Exactly which additional libraries are needed will depend on the type of graphics being produced. The two PGF libraries shown here allow for a variety of arrowheads and “snakes”, a class of wavy lines.

```
\documentclass[11pt]{article}
...
\usepackage{tikz}
% Optional PGF libraries
\usepackage{pgflibraryarrows}
\usepackage{pgflibrarysnakes}
...
\begin{document}
...
\begin{tikzpicture}
...
\end{tikzpicture}
...
\end{document}
```

Figure 1: Layout of a TikZ-based document.

Commands which describe the graphic to be drawn appear within a `tikzpicture` environment. In the simplest case, these commands describe paths consisting of straight line segments joining points in the plane. For more complex graphics, other primitive graphics objects can appear; e.g., rectangles,

```

\begin{tikzpicture}
  \draw (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\end{tikzpicture}

```

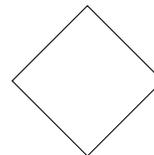


Figure 2: Drawing a diamond with a closed path.

```

\begin{tikzpicture}
  \draw[step=0.25cm,color=gray] (-1,-1) grid (1,1);
  \draw (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\end{tikzpicture}

```

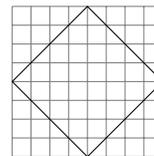


Figure 3: Adding a grid.

circles, arcs, text, grids, and so forth.

Figure 2 illustrates how a diamond can be obtained, using the `draw` command to cause a “pen” to form a closed path joining the four points  $(1,0)$ ,  $(0,1)$ ,  $(-1,0)$ , and  $(0,-1)$ , specified with familiar Cartesian coordinates. The syntax used to specify this path is very similar to that used by MetaPost [2]. Unlike MetaPost, TikZ uses one centimeter as the default unit of measure, so the four points used in this example lie on the  $x$  and  $y$  axes, one centimeter from the origin.

In the process of developing and “debugging” graphics, it can be helpful to include a background grid. Figure 3 expands on the example of Figure 2 by adding a `draw` command to cause a grid to appear:

```

\draw[step=0.25cm,color=gray]
  (-1,-1) grid (1,1);

```

In this command, the grid is specified by providing two diagonally opposing points:  $(-1,-1)$  and  $(1,1)$ . The two options supplied give a step size for the grid lines and a specification for the color of the grid lines, using the `xcolor` package [3].

#### 4 Specifying points and paths in TikZ

Two key ideas used in TikZ are *points* and *paths*. Both of these ideas were used in the diamond examples. Much more is possible, however. For example, points can be specified in any of the following ways:

- Cartesian coordinates
- Polar coordinates
- Named points
- Relative points

As previously noted, the Cartesian coordinate  $(a,b)$  refers to the point  $a$  centimeters in the  $x$ -direction and  $b$  centimeters in the  $y$ -direction.

A point in polar coordinates requires an angle  $\alpha$ , in degrees, and distance from the origin,  $r$ . Unlike Cartesian coordinates, the distance does not have a

default dimensional unit, so one must be supplied. The syntax for a point specified in polar coordinates is  $(\alpha : r \textit{ dim})$ , where *dim* is a dimensional unit such as `cm`, `pt`, `in`, or any other TeX-based unit. Other than syntax and the required dimensional unit, this follows usual mathematical usage. See Figure 4.

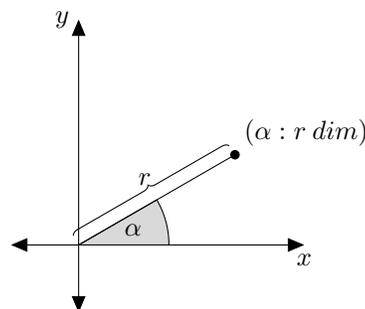


Figure 4: Polar coordinates in TikZ.

It is sometimes convenient to refer to a point by name, especially when this point occurs in multiple `draw` commands. The command:

```

\path (a,b) coordinate (P);

```

assigns to  $P$  the Cartesian coordinate  $(a,b)$ . In a similar way,

```

\path (α:r dim) coordinate (Q);

```

assigns to  $Q$  the polar coordinate with angle  $\alpha$  and radius  $r$ .

Figure 5 illustrates the use of named coordinates and several other interesting capabilities of TikZ. First, infix-style arithmetic is used to help define the points of the pentagon by using multiples of 72 degrees. This feature is made possible by the `calc` package [5], which is automatically included by TikZ. Second, the `draw` command specifies five line segments, demonstrating how the drawing pen can be moved by omitting the `--` operator.

```

\begin{tikzpicture}
  % Define the points of a regular pentagon
  \path (0,0) coordinate (origin);
  \path (0:1cm) coordinate (P0);
  \path (1*72:1cm) coordinate (P1);
  \path (2*72:1cm) coordinate (P2);
  \path (3*72:1cm) coordinate (P3);
  \path (4*72:1cm) coordinate (P4);

  % Draw the edges of the pentagon
  \draw (P0) -- (P1) -- (P2) -- (P3) -- (P4) -- cycle;

  % Add "spokes"
  \draw (origin) -- (P0) (origin) -- (P1) (origin) -- (P2)
        (origin) -- (P3) (origin) -- (P4);
\end{tikzpicture}

```

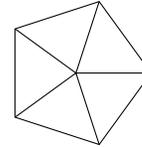


Figure 5: Using named coordinates.

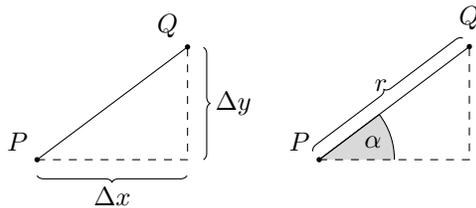


Figure 6: A relative point,  $Q$ , determined with Cartesian or polar offsets.

The concept of the *current point* plays an important role when multiple actions are involved. For example, suppose two line segments are drawn joining points  $P$  and  $Q$  along with  $Q$  and  $R$ :

```
\draw (P) -- (Q) -- (R);
```

Viewed as a sequence of actions, the drawing pen begins at  $P$ , is moved to  $Q$ , drawing a first line segment, and from there is moved to  $R$ , yielding a second line segment. As the pen moves through these two segments, the current point changes: it is initially at  $P$ , then becomes  $Q$  and finally becomes  $R$ .

A relative point may be defined by providing offsets in each of the horizontal and vertical directions. If  $P$  is a given point and  $\Delta x$  and  $\Delta y$  are two offsets, a new point  $Q$  may be defined using a `++` prefix, as follows:

```
\path (P) ++(\Delta x,\Delta y) coordinate (Q);
```

Alternately, the offset may be specified with polar coordinates. For example, given angle  $\alpha$  and radius  $r$ , with a dimensional unit *dim*, the command:

```
\path (P) ++(\alpha:r dim) coordinate (Q);
```

specifies a new point  $Q$ . See Figure 6.

There are two forms of relative points—one which updates the current point and one which does

not. The `++` prefix updates the current point while the `+` prefix does not.

Consider line segments drawn between points defined in a relative manner, as in the example of Figure 7. The path is specified by offsets: the drawing pen starts at the origin and is adjusted first by the offset  $(1,0)$ , followed by the offset  $(1,1)$ , and finally by the offset  $(1,-1)$ .

By contrast, Figure 8 shows the effect of using the `+` prefix. Since the current point is not updated in this variation, every offset which appears is performed relative to the initial point,  $(0,0)$ .

### Beyond line segments

In addition to points and line segments, there are a number of other graphic primitives available. These include:

- Grids and rectangles
- Circles and ellipses
- Arcs
- Bézier curves

As previously discussed, a grid is specified by providing two diagonally opposing points and other options which affect such things as the color and spacing of the grid lines. A rectangle can be viewed as a simplified grid—all that is needed are two diagonally opposing points of the rectangle. The syntax

```
\draw (P) rectangle (Q);
```

draws the rectangle specified by the two “bounding box” points  $P$  and  $Q$ . It is worth noting that the current point is updated to  $Q$ , a fact which plays a role if the `\draw` command involves more than one drawing action. Figure 9 provides an example where

```
\begin{tikzpicture}
\draw (0,0) -- ++(1,0) -- ++(1,1) -- ++(1,-1);
\end{tikzpicture}
```

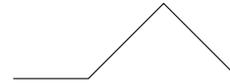


Figure 7: Drawing a path using relative offsets.

```
\begin{tikzpicture}
\draw (0,0) -- +(1,0) -- +(0,-1) -- +(-1,0) -- +(0,1);
\end{tikzpicture}
```

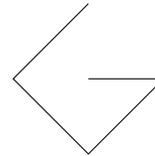


Figure 8: Drawing a path using relative offsets without updating the current point.

```
\begin{tikzpicture}
\draw (0,0) rectangle (1,1)
rectangle (3,2)
rectangle (4,3);
\end{tikzpicture}
```

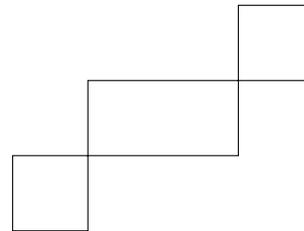


Figure 9: Drawing rectangles.

```
\begin{tikzpicture}
\draw (0,0) circle (1cm)
circle (0.6cm)
circle (0.2cm);
\end{tikzpicture}
```

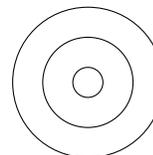


Figure 10: Drawing circles — one `draw` command with multiple actions.

```
\begin{tikzpicture}
\draw (0,0) circle (1cm);
\draw (0.5,0) circle (0.5cm);
\draw (0,0.5) circle (0.5cm);
\draw (-0.5,0) circle (0.5cm);
\draw (0,-0.5) circle (0.5cm);
\end{tikzpicture}
```

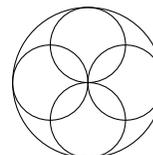


Figure 11: Drawing circles — a sequence of `draw` commands.

three rectangles are drawn in succession. Each rectangle operation updates the current point, which then serves as one of the bounding box points for the following rectangle.

A circle is specified by providing its center point and the desired radius. The command:

```
\draw (a,b) circle (r dim);
```

causes the circle with radius  $r$ , with an appropriate dimensional unit, and center point  $(a, b)$  to be drawn. The current point is not updated as a result. Figures 10 and 11 provide examples.

The situation for an ellipse is similar, though two radii are needed, one for each axis. The syntax:

```
\draw (a,b) ellipse (r1 dim and r2 dim);
```

causes the ellipse centered at  $(a, b)$  with semi-axes

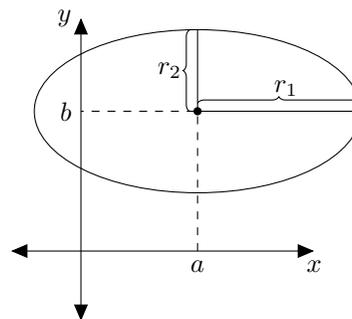


Figure 12: An ellipse in TikZ.

```

\begin{tikzpicture}
  \draw (0,0) ellipse (2cm and 1cm)
        ellipse (0.5cm and 1 cm)
        ellipse (0.5cm and 0.25cm);
\end{tikzpicture}

```

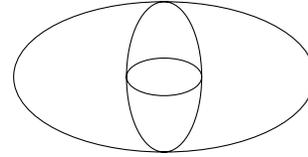


Figure 13: Three ellipses produced with a single `draw` command.

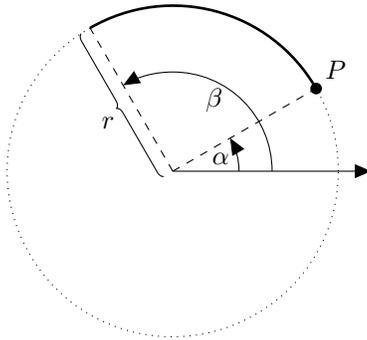


Figure 14: An arc in TikZ.

```

\begin{tikzpicture}
  \draw (0:0.7cm) -- (0:1.5cm)
        arc (0:60:1.5cm) -- (60:0.7cm)
        arc (60:0:0.7cm) -- cycle;
\end{tikzpicture}

```



Figure 15: Combining arcs and line segments.

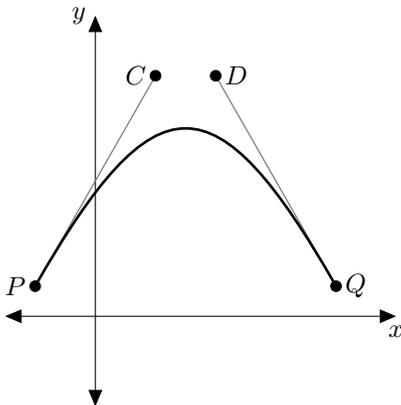


Figure 16: A Bézier curve.

$r_1$  and  $r_2$  to be drawn. See Figure 12. Like `circle`, the `ellipse` command does not change the current point, so multiple ellipses which share the same center point can be drawn with a single `draw` command, as Figure 13 shows.

Arcs may also be specified in TikZ. For a circular arc, what is required is an initial point on the circle, the radius of the circle and an indication of how much of the circle to be swept out. In more detail, the syntax

```

\draw (P) arc ( $\alpha$ : $\beta$ : $r$  dim);

```

draws the arc shown in Figure 14. At first glance it might seem unusual to use the point  $P$  and not the center point of the circle. However, when one realizes that the `arc` might be just one of several components of a `draw` command, it is very natural to use the point  $P$ , as it will be the current point.

For example, Figure 15 shows how to draw a portion of an annulus by drawing two arcs and two line segments. This particular figure is drawn by directing the pen in a counter-clockwise fashion—the horizontal line segment, the outer circular arc, a line segment, and finally the inner arc.

TikZ also provides the ability to produce Bézier curves. The command

```

\draw (P) .. controls (C)
        and (D) .. (Q);

```

draws the curve shown in Figure 16. Four points are needed: an initial point  $P$ , a final point  $Q$ , and two control points. The location of the control points controls the extent of the curve and the slope of the curve at the initial and final points.

Bézier curves provide for a wealth of variety, as Figure 17 indicates.

An alternate syntax for Bézier curves allows for a more convenient specification of the curvature at the starting and ending points. Using polar coordinates with respect to these two points provides this capability. The syntax is as follows:

```

\draw (P) .. controls +( $\alpha$ : $r_1$  dim)
        and +( $\beta$ : $r_2$  dim) .. (Q);

```

See Figure 18.

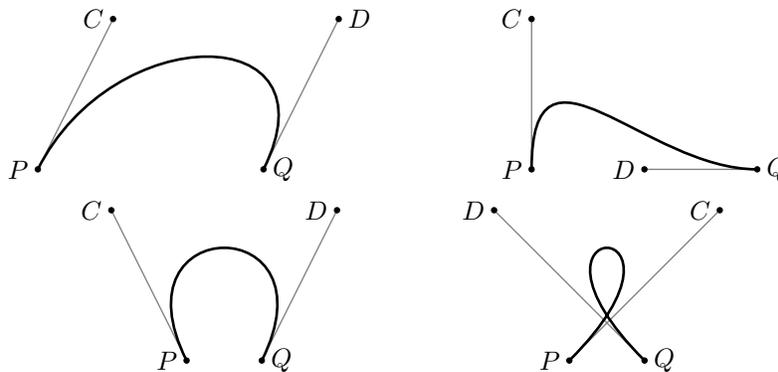


Figure 17: Various Bézier curves.

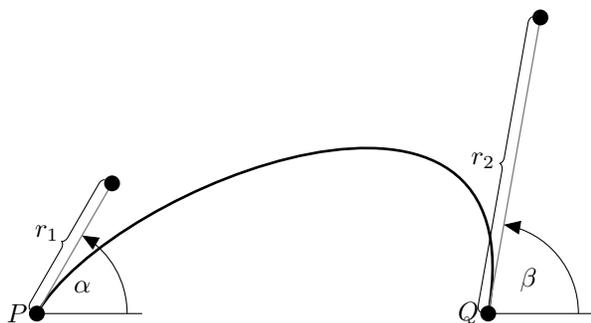


Figure 18: A Bézier curve specified with relative coordinates.

## 5 From coordinates to nodes

A *node* is a generalization of the coordinate primitive. Two characteristics of a node are its *shape* and its *text*. A node allows for arbitrary  $\text{\TeX}$  text to appear within a diagram. The command

```
\path (0,0)
  node[draw,shape=circle] (v0)
  {\$v_0$};
```

defines a node named `v0`, centered at the origin, with a circular shape and text component `\$v_0$`. The `draw` option causes the associated shape (in this case, a circle) to be drawn. Figure 19 illustrates how nodes can be used to draw an undirected graph. Notice how line segments which join nodes stop at the boundary of the shape rather than protruding into the center point of the node. In this example, we have made use of the `tikzstyle` command to factor out code that would otherwise be repeated in each of the `node` commands.

Additionally, this example illustrates the use of the option `[scale=2]`, which indicates the result is to be scaled by a factor of 2. Using scale factors allows the picture to be designed in convenient units, then resized as desired. However, scaling a `TikZ`

picture does not scale the font size in use.

There are various features within `TikZ` which provide fine control over nodes. Many of these are related to how line segments or curves connect a pair of nodes. For example, one can provide specific locations on the node's shape where connections should touch, whether or not to shorten the connection, how and where to annotate the connection with text, and so forth.

## 6 Loops

`TikZ` provides a loop structure which can simplify the creation of certain types of graphics. The basic loop syntax is as follows:

```
\foreach \var in {iteration list}
{
  loop body
}
```

The loop variable, `\var`, takes on the values given in the iteration list. In the simplest case, this list can be a fixed list of values, such as `{1,2,3,4}` or as an implied list of values, such as `{1,...,4}`.

Consider the following loop. Four coordinates, `X1` through `X4` are introduced at  $(1, 0)$ ,  $(2, 0)$ ,  $(3, 0)$ , and  $(4, 0)$ , respectively. In addition, a small filled circle is drawn at each coordinate.

```
\foreach \i in {1,...,4}
{
  \path (\i,0) coordinate (X\i);
  \fill (X\i) circle (1pt);
}
```

Figure 20 shows how to extend this idea to yield a bipartite graph. As one might expect, `foreach` loops can be nested, a feature utilized here to specify all the edges in the graph.

Iteration lists need not consist of consecutive integers. An implicit step size is obtained by providing the first two values of the list in addition to

```

\begin{tikzpicture}[scale=2]
  \tikzstyle{every node}=[draw,shape=circle];
  \path (0:0cm) node (v0) {$v_0$};
  \path (0:1cm) node (v1) {$v_1$};
  \path (72:1cm) node (v2) {$v_2$};
  \path (2*72:1cm) node (v3) {$v_3$};
  \path (3*72:1cm) node (v4) {$v_4$};
  \path (4*72:1cm) node (v5) {$v_5$};

  \draw (v0) -- (v1)
        (v0) -- (v2)
        (v0) -- (v3)
        (v0) -- (v4)
        (v0) -- (v5);
\end{tikzpicture}

```

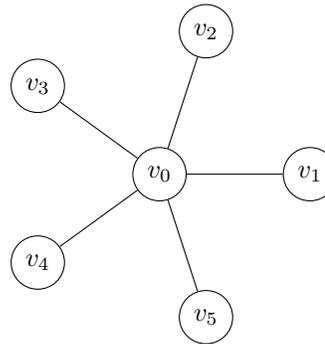


Figure 19: An undirected graph drawn with nodes.

```

\begin{tikzpicture}[scale=2]
  \foreach \i in {1,...,4}
  {
    \path (\i,0) coordinate (X\i);
    \fill (X\i) circle (1pt);
  }
  \foreach \j in {1,...,3}
  {
    \path (\j,1) coordinate (Y\j);
    \fill (Y\j) circle (1pt);
  }
  \foreach \i in {1,...,4}
  {
    \foreach \j in {1,...,3}
    {
      \draw (X\i) -- (Y\j);
    }
  }
\end{tikzpicture}

```

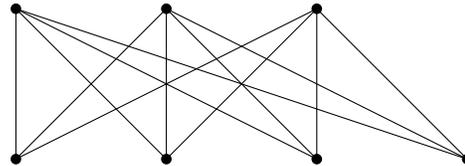


Figure 20: A bipartite graph drawn using loops.

the final value. For example,

```

\foreach \angle in {0,60,...,300}
{
  loop body
}

```

causes `\angle` to take on values of the form  $60k$ , where  $0 \leq k \leq 5$ .

Specifying pairs of values in an iteration list provides simultaneous iteration over these values. For example,

```

\foreach \angle / \c in
  {0/red,120/green,240/blue}
{
  loop body
}

```

produces three iterations of the loop body, successively assigning the pairs (0, red), (120, green), and (240, blue) to the variables `\angle` and `\c`.

## 7 Plotting

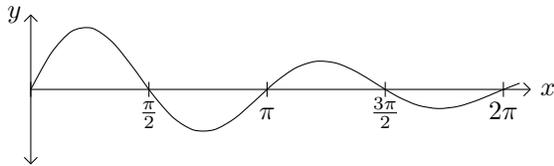
A list of points can be plotted using the TikZ `plot` command. Lists can be generated three ways: on-the-fly by `gnuplot` [6], read from a file, or specified within a `plot` itself. These approaches are supported by the following commands:

```

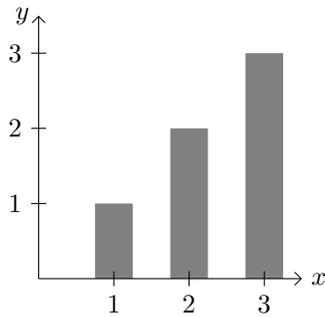
\draw plot function{gnuplot formula};
\draw plot file{filename};
\draw plot coordinates{point sequence};

```

Using other TikZ commands, these graphs can be enhanced with symbols or other desired annotations.



**Figure 21:** The graph of a function, with tick marks and annotations.



**Figure 22:** A graph that includes a bar chart.

Figure 21 provides an example of one such plot, the graph of  $y = \sin(2x)e^{-x/4}$ . The curve itself is generated with the command:

```
\draw[smooth,domain=0:6.5]
  plot function{\sin(2*x)*exp(-x/4)};
```

This command causes `gnuplot`<sup>†</sup> to generate points of the graph, saving them in a file, which is subsequently processed by `TikZ`. The `smooth` option joins these points with a curve, in contrast to line segments. Although not used in this example, the `samples` option can be used to control the number of generated points. The `domain` option specifies the desired range of  $x$  values. Everything else which appears in this graph, including axes, tick marks, and multiples of  $\pi/2$  have been added with additional `TikZ` commands.

A list of points can be used to create a bar chart, as illustrated in Figure 22. Each of the bars is drawn by command:

```
\draw[ycomb,
  color=gray,
  line width=0.5cm]
  plot coordinates{(1,1) (2,2) (3,3)};
```

The `ycomb` option specifies vertical bars are to be drawn and `line width` establishes the width of the bars.

<sup>†</sup> To generate points with `gnuplot`, `TeX` must be configured to allow external programs to be invoked. For `TeX Live`, this can be accomplished by adjusting `texmf.cnf` to allow a shell escape.

## 8 Clipping and scope

It is sometimes useful to be able to specify regions of a graphic where drawing is allowed to take place—any drawing which falls outside this defined region is “clipped” and is not visible.

This feature is made available by the `\clip` command, which defines the clipping region. For example,

```
\clip (-0.5,0) circle (1cm);
```

specifies that all future drawing should take place relative to the clipping area consisting of the circle centered at  $(-0.5,0)$  with radius 1 cm. Figure 23 shows how to fill a semicircle with clipping. The yin-yang symbol, a popular example, can be easily obtained by superimposing four filled circles on this filled semicircle:



When multiple `\clip` commands appear, the effective clipping region is the intersection of all specified regions. For example,

```
\clip (-0.5,0) circle (1cm);
\clip (0.5,0) circle (1cm);
```

defines a clipping area corresponding to the intersection of the two indicated circles. All subsequent commands which cause drawing to occur are clipped with respect to this region.

A scoping mechanism allows a clipping region to be defined for a specified number of commands. This is achieved with a `scope` environment. Any commands inside this environment respect the clipping region; commands which fall outside behave as usual. For example,

```
\begin{scope}
  \clip (-0.5,0) circle (1cm);
  \clip (0.5,0) circle (1cm);
  \fill (-2,1.5) rectangle (2,-1.5);
\end{scope}
```

shades the intersection of two overlapping circles, since the filled rectangle is clipped to this region. Commands which follow this `scope` environment are not subject to this clipping region. Figure 24 shows a complete example which makes use of `\clip` and scoping.

The scoping mechanism may also be used to apply options to a group of actions, as illustrated in Figure 25. In this example, options to control color and line width are applied to each of three successive `\draw` commands, yielding the top row of the figure. At the conclusion of the `scope` environment, the remaining `\draw` commands revert to the `TikZ` defaults, yielding the lower row of the figure.

```

\begin{tikzpicture}
  \draw (0,0) circle (1cm);
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{tikzpicture}

```

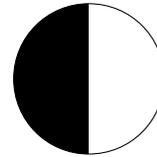


Figure 23: An example of clipping.

```

\begin{tikzpicture}
  \draw (-2,1.5) rectangle (2,-1.5);
  \begin{scope}
    \clip (-0.5,0) circle (1cm);
    \clip (0.5,0) circle (1cm);
    \fill[gray] (-2,1.5) rectangle (2,-1.5);
  \end{scope}
  \draw (-0.5,0) circle (1cm);
  \draw (0.5,0) circle (1cm);
\end{tikzpicture}

```

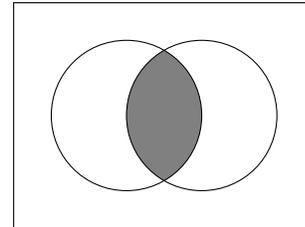


Figure 24: Using clipping and scope to show set intersection.

```

\begin{tikzpicture}[scale=1.5]
  \begin{scope}[color=gray,line width=4pt]
    \draw (0,0) -- (1,1);
    \draw (1,0) -- (0,1);
    \draw (-0.5,0.5) circle (0.5cm);
  \end{scope}
  \draw (0,0) -- (-1,-1);
  \draw (0,-1) -- (-1,0);
  \draw (0.5,-0.5) circle (0.5cm);
\end{tikzpicture}

```

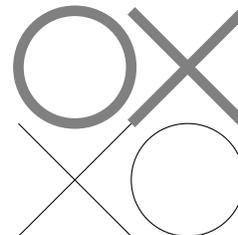


Figure 25: Using scope to apply options.

## 9 Summary

TikZ, a high-level interface to PGF, is a language-based tool for specifying graphics. It uses familiar graphics-related concepts, such as point, line, and circle and has a concise and natural syntax. It meshes well with pdfL<sup>A</sup>T<sub>E</sub>X in that no additional processing steps are needed. Another positive aspect of TikZ is its ability to blend T<sub>E</sub>X fonts, symbols, and mathematics within the generated graphics.

We are especially indebted to Till Tantau for developing TikZ and for contributing it to the T<sub>E</sub>X community.

## References

- [1] Robin Fairbairns, ed. *The UK T<sub>E</sub>X FAQ*. <ftp://cam.ctan.org/tex-archive/help/uk-tex-faq/letterfaq.pdf>.
- [2] John Hobby. *Introduction to MetaPost*. [http://cm.bell-labs.com/who/hobby/92\\_2-21.pdf](http://cm.bell-labs.com/who/hobby/92_2-21.pdf).
- [3] Uwe Kern. *Extending L<sup>A</sup>T<sub>E</sub>X's color facilities: the xcolor package*. <http://www.ctan.org/tex-archive/macros/latex/contrib/xcolor>.
- [4] Till Tantau. *TikZ and PGF, Version 1.01*. <http://sourceforge.net/projects/pgf/>.
- [5] Kresten Krab Thorup, Frank Jensen, and Chris Rowley. *The calc package: infix arithmetic in L<sup>A</sup>T<sub>E</sub>X*. <ftp://tug.ctan.org/pub/tex-archive/macros/latex/required/tools/calc.pdf>.
- [6] Thomas Williams and Colin Kelley. *gnuplot*. <http://www.gnuplot.info/>.

# Drawing medical pedigree trees with $\text{\TeX}$ and PSTricks

Boris Veytsman

Computational Materials Science Center, MS 5A2  
George Mason University  
Fairfax, VA 22030  
borisv (at) lk dot net

Leila Akhmadeeva

Bashkir State Medical University  
3 Lenina Str. Ufa, 450077, Russia  
leila\_ufa (at) mail dot ru

## Abstract

Medical pedigrees look like genealogical trees, but also have certain interesting features. Usually they are drawn by hand by medical geneticists. This is a cumbersome and time-consuming process. Freely available programs for drawing genealogies are not fully suitable for this task because of the special format of medical pedigrees.

We discuss a package for drawing pedigrees based on PSTricks. The information is input by geneticists in a spreadsheet; a Perl program extracts it and calls  $\text{\TeX}$  to produce the final output.

## 1 Introduction

A medical pedigree is a very important tool for clinicians, genetic researchers and medical educators. As noted by Bennett, Steinhaus, Uhrich, O’Sullivan, Resta, Lochner-Doyle, Markei, Vincent, and Hamanishi, 1995, “The construction of an accurate family pedigree is a fundamental component of a clinical genetic evaluation and of human genetic research.” Regrettably, at present most geneticists make pedigrees manually. There are several programs for making pedigrees (see the list at <http://www.kumc.edu/gec/prof/genecomp.html#pedigree>), but they are rather expensive, lack multi-language support and the quality of typesetting is wanting. Maybe this is why they are not in wide use.

We tried to write a program suitable for constructing complex pedigrees from genetics data in English, Russian and possibly other languages.  $\text{\TeX}$  and PSTricks (Van Zandt, 1993) seemed to be a natural choice. First, the quality of  $\text{\TeX}$  typesetting is unsurpassed. Pedigrees combine graphical and textual data, therefore a package like PSTricks is handy to produce them. There are good facilities for multi-language support in  $\text{\TeX}$ , therefore we could easily deploy them.  $\text{\TeX}$  code can be easily generated by other programs, which is another advantage of implementing the drawing as a  $\text{\TeX}$  file.

The idea to draw genealogical trees with  $\text{\TeX}$  and PSTricks is by no means new. One can enjoy the beautiful trees at <http://www.tug.org/PSTricks/main.cgi?file=Examples/Genealogy/genealogy>.

The problem is, medical pedigrees differ from genealogical trees. They might not even be trees from the mathematical point of view: any marriage between relatives adds a cycle to the graph of relationships. Even if there are no such marriages, pedigrees are not layered rooted trees in the terminology of graph theory (Di Battista, Eades, Tamassia, and Tollis, 1999). The difference is the following. Layered rooted trees have an “oldest” node (global ancestor). This node has no ancestors. It has descendants, and each of them is an ancestor for its own layered rooted tree. On the other hand, a geneticist drawing a pedigree is interested both in the male and female ancestors of the patient, as well as in the male and female ancestors of *them*, etc. Therefore a pedigree might have several “local ancestors”: nodes that do not have ancestors. This makes the problem of drawing pedigrees quite interesting.

We divided the problem of drawing pedigrees into two parts. First, we developed a set of PSTricks macros (Veytsman and Akhmadeeva, 2006a) to draw (almost) any pedigree. They can be used “manually”, i.e., the user can put nodes at any place on the

canvas and make any connections between them, using `\rput`, `\psmatrix` or even `\pstree` (if the pedigree is in fact a layered tree). Second, we developed a Perl program (Veytsman and Akhmadeeva, 2006b) that can process relationship data in the CSV (comma separated values) format and output  $\TeX$  code for an important subset of pedigrees: when there is no inbreeding, and only the people having common genes with the primary patient (*proband*) are shown. Both these parts are discussed below.

## 2 PSTricks macros

From the perspective of graph theory a pedigree is a collection of nodes and connections. There are three main kinds of nodes in our macros: `\PstPerson` to show persons, `\PstAbortion` to show terminated pregnancies, and a *marriage node*,<sup>1</sup> which is implemented as a simple `\pnode`.<sup>2</sup> Each node except `\PstAbortion` may have *descendants*: children of this person or of this marriage. Each node except the marriage node has exactly one *ancestor node*: a person or a marriage. Additionally a marriage node has a male and a female spouse.

The nodes may have a number of properties: they might be affected by a disease (or not), they might be deceased, they are male or female. These properties are shown by optional arguments to the corresponding command. Each node drawing command has one obligatory argument: the name of the node, which is used to draw node connections.

The first and simplest connection command is `\PstDescent`, which shows the relationship between an ancestor (a person or marriage node) and descendant. It is implemented internally as an `\ncangle` command. There are special commands to show relationships between twins and their parents, between spouses, infertility of a union or a person, etc.

An extremely complex pedigree is used as an example in the paper of Bennett, Steinhaus, Uhrich, O’Sullivan, Resta, Lochner-Doyle, Markei, Vincent, and Hamanishi (1995). In Figure 1 we reproduce this pedigree. The corresponding code is shown in Figures 2, 3 and 4.

A version of node-drawing commands makes *tree nodes*. They are useful if the pedigree is a tree or can be constructed from a tree by adding several connections. An example of such a pedigree is shown in Figure 5, and the corresponding code in Figure 6. Note that the pedigree on this figure is

<sup>1</sup> For our purposes both official marriages and unofficial unions are loosely referred to as marriages.

<sup>2</sup> There are also special nodes for special circumstances: *twin node* is used to show the relation between twins, a special node is used to show infertility of person or a marriage, etc.

*not* a tree from the mathematical point of view due to the marriage between Peter and Joan.

The user manual (Veytsman and Akhmadeeva, 2006a) is distributed with the macros. By tradition, PSTricks-related code works with both  $\LaTeX$  and plain  $\TeX$ , so our code is written to work in both these modes too. However, most of testing was done in the  $\LaTeX$  mode only.

## 3 Perl program

The  $\TeX$  code in the examples above is straightforward. Nevertheless, it might be too much to expect from geneticists to write it themselves. Therefore if we want the code to work not only for the authors, but also for medical and science professionals, we must find a way to generate this code from the relationship data automatically. This is the aim of the second part of our project (Veytsman and Akhmadeeva, 2006b).

The idea of the program is to take the data in a simple format, easy for the users to understand (and easy to generate in turn from databases or other sources) and convert them into  $\TeX$  code above. The input format was chosen to be a plain text CSV format (originally the acronym meant Comma Separated Values, but it is often used now for any plain text separated data). In this format each person record is a line separated into fields by the symbol “|”. The fields show the name of the person, the dates of birth and death, the genetic condition, etc. An important field is *Id*, a unique identifier of the person. The relations between the persons are set by the fields “Mother” and “Father”, which contain Ids of the parents of the given person. In this way we do not need to specifically show the relations between spouses, siblings, etc.: spouses in our system are the persons who have common children, and siblings are the persons with common parent(s). An example of an input file is shown in Figure 7. Such files are easy to generate by common spreadsheet programs and from databases.

The code generated from the input file in Figure 7 is shown in Figure 8, and the typeset pedigree in Figure 9.

Our program works with a subset of all possible pedigrees. Still, this subset covers most pedigrees used by geneticists. First, we do not include inbreeding unions in the pedigrees, so our graphs are in fact trees. Second, we show only the persons that have common genes with the proband.<sup>3</sup> Note that a pedigree that violates these rules may not allow

<sup>3</sup> The proband is the first person among the relatives who came to a geneticist; he or she is the primary patient.

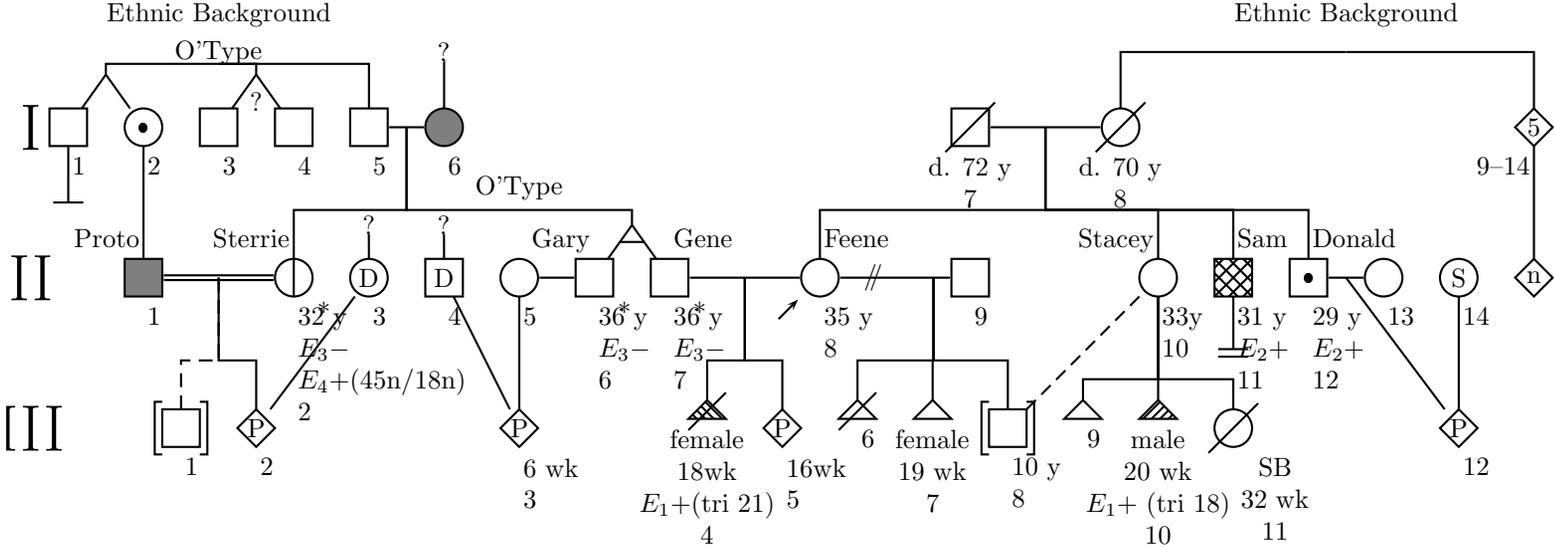


Figure 1: A complex pedigree from Bennett, Steinhaus, Ulrich, O'Sullivan, Resta, Lochner-Doyle, Markei, Vincent, and Hamanishi (1995)

```

\psset{armB=1.1, hatchsep=1.5pt}
\rput(3.5,8){Ethnic Background}
\rput(18.5,8){Ethnic Background}
\rput(3.5,7.5){\rnode{0Type1}{0'Type}}
\rput(18.5,7.5){\pnode{Origin2}}
\rput(6.5,7.5){\rnode{Quest1}{?}}
\rput(1,6.5){\Huge I}
\rput(1.5,6.5){\pstPerson[male, belowtext=1]{I1}}
\rput(2.5,6.5){\pstPerson[female, obligatory, belowtext=2]{I2}}
\rput(3.5,6.5){\pstPerson[male, belowtext=3]{I3}}
\rput(4.5,6.5){\pstPerson[male, belowtext=4]{I4}}
\rput(5.5,6.5){\pstPerson[male, belowtext=5]{I5}}
\rput(6.5,6.5){\pstPerson[female, affected,
  belowtext=6]{I6}}
\rput(2,7.2){\pnode{Twins1}}
\rput(4,7.2){\pnode{Twins2}}
\pstTwins[armB=0]{0Type1}{Twins1}{I1}{I2}
\pstTwins[qzygotic, armB=0, mzlinepos=0.8]{0Type1}{Twins2}{I3}{I4}
\pstDescent[armB=0]{0Type1}{I5}
\pstDescent[armB=0]{Quest1}{I6}
\pstRelationship[descentnode=I5I6]{I5}{I6}
\rput(1.5,5.5){\pstChildless{CI1}}
\ncline{I1}{CI1}
\rput(13.5,6.5){\pstPerson[male, deceased, belowtext=trp=t,
  belowtext=\parbox{2cm}{\centering d. 72 y\7}]{I7}}
\rput(15.5,6.5){\pstPerson[female, deceased, belowtext=trp=t,
  belowtext=\parbox{2cm}{\centering d. 70 y\8}]{I8}}
\pstRelationship[descentnode=I7I8]{I7}{I8}
\rput(21,6.5){\pstPerson[insidetext=5, belowtext=9--14,
  belowtext=rt]{I9}}
\pstDescent[armB=0]{Origin2}{I8}
\pstDescent[armB=0]{Origin2}{I9}

```

**Figure 2:** Code for Figure 1: Generation I

two-dimensional drawing without self-intersections at all.

The biggest problem is that even with these rules the pedigree tree is in the general case not a layered tree, and the algorithm by Reingold and Tilford (Di Battista, Eades, Tamassia, and Tollis, 1999, § 3.1) would not work. We therefore describe a generalization of the Reingold-Tilford algorithm.

First we will summarize the main idea of the Reingold-Tilford algorithm. We draw a tree down in the  $y$  direction. The algorithm is recursive. We start from the root of the tree. If it has no descendants, it is easy to draw it. If it has descendants, each descendant is a rooted layered tree. We draw them, recursively applying the algorithm. Then we move them in the horizontal direction as close as possible, and put the root one layer above with an  $x$  coordinate in the middle of the descendants.

Now we can generalize this algorithm for our case.

There are two kinds of nodes in the pedigree graph: person nodes and marriage nodes. A node has a *predecessor* and *children*. A marriage node does not have a predecessor, but has *male spouse* and *female spouse* (usually male spouses are on the left and female spouses are on the right on pedigrees). Any node has a *downward tree* of its children, grandchildren, etc. The downward tree may be empty.

Any node in an acyclic graph can be a root. However, in layered trees there is a special root: the one that has no predecessor. Similarly, we will call a *local root* a node that has no predecessor. All marriage nodes are local roots. Some person nodes can be local roots as well.

```

\rput(1,4.5){\Huge II}
\rput(2.5,4.5){\pstPerson[male, affected, belowtext=1,
  abovetext=Proto, abovetextrp=rB]{II1}}
\rput(4.5,4.5){\pstPerson[female, asymptomatic,
  belowtext=\parbox{3cm}{32 y\ $E_3-$\ $E_4+(45n/18n)\2},
  abovetext={Sterrie}, abovetextrp=rB, evaluated}{II2}}
\pstDescent{I2}{II1}\pstDescent{I5I6}{II2}
\pstRelationship[consanguinic, descentnode=II1II2]{II1}{II2}
\rput(5.5,5.2){\rnode{Quest2}{?}}
\rput(5.5,4.5){\pstPerson[female, insidetext=D, belowtext=3]{II3}}
\rput(6.5,5.2){\rnode{Quest3}{?}}
\rput(6.5,4.5){\pstPerson[male, insidetext=D, belowtext=4]{II4}}
\ncline{Quest2}{II3}\ncline{Quest3}{II4}
\rput(7.5,4.5){\pstPerson[female, belowtext=5]{II5}}
\rput(8.5,4.5){\pstPerson[male, abovetext=Gary, abovetextrp=rB,
  belowtext=\parbox{2cm}{36 y\ $E_3-$\6}, evaluated}{II6}}
\rput(9.5,4.5){\pstPerson[male, abovetext={Gene},
  belowtext=\parbox{2cm}{36 y\ $E_3-$\7}, evaluated}{II7}}
\rput(9,5.2){\pnode{Twins3}}
\pstTwins[monozygotic]{I5I6}{Twins3}{II6}{II7}
\pstRelationship{II5}{II6}
\rput(7.5,5.7){0>Type}
\rput(11.5,4.5){\pstPerson[female, proband,
  belowtext=\parbox{1cm}{35 y\8}, abovetext=Feene}{II8}}
\pstRelationship[descentnode=II7II8]{II7}{II8}
\rput(13.5,4.5){\pstPerson[male, belowtext=9]{II9}}
\pstRelationship[broken, descentnode=II8II9,
  descentnodepos=0.85]{II8}{II9}
\rput(16,4.5){\pstPerson[abovetext=Stacey, female,
  abovetextrp=rB, belowtext=\parbox{1cm}{33y\ 10}]{II10}}
\def\affectedstyle{fillstyle=crosshatch}
\rput(17,4.5){\pstPerson[male, affected, abovetext=Sam,
  belowtext=\parbox{3cm}{31 y\ $E_2+$\ 11}, hatchsep=3pt]{II11}}
\rput(17,3.5){\pstChildless[infertile]{C2}}
\ncline{II11}{C2}
\rput(18,4.5){\pstPerson[male, obligatory, abovetext=Donald,
  belowtext=\parbox{3cm}{29 y\ $E_2+$\ 12}]{II12}}
\pstDescent{I7I8}{II8}\pstDescent{I7I8}{II10}
\pstDescent{I7I8}{II11}\pstDescent{I7I8}{II12}
\rput(19,4.5){\pstPerson[female, belowtext=13]{II13}}
\pstRelationship[descentnode=II12II13]{II12}{II13}
\rput(20,4.5){\pstPerson[female, insidetext=S, belowtext=14]{II14}}
\rput(21,4.5){\pstPerson[insidetext=n]{II15}}
\pstDescent{I9}{II15}

```

**Figure 3:** Code for Figure 1: Generation II



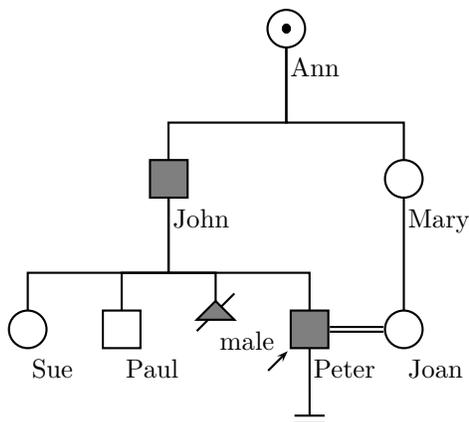


Figure 5: Example of pedigree with tree-making commands

```

\begin{pspicture}(0,1)(7,7)
  \rput(3,4){%
    \pstree{\TpstPerson[female, obligatory, belowtext=Ann]{Ann}}{%
      \def\psedge{\pstDescent}\psset{armB=1}
      \pstree{\TpstPerson[male, affected, belowtext=John]{John}}{%
        \TpstPerson[female, belowtext=Sue]{Sue}
        \TpstPerson[male, belowtext=Paul]{Paul}
        \TpstAbortion[affected, belowtext=male]{A1}
        \pstree[thislevelsep=1.2]{\TpstPerson[male,
          belowtext=Peter, affected, proband]{Peter}}{%
          \def\psedge{\ncline}
          \TpstChildless[infertile]{C1}
        }
      }
    }
  \pstree{\TpstPerson[female, belowtext=Mary]{Mary}}{
    \TpstPerson[female, belowtext=Joan]{Joan}
  }
}
\pstRelationship[consanguinic]{Peter}{Joan}
\end{pspicture}

```

Figure 6: Code producing Figure 5

Id	Name	Sex	DoB	DoD	Mother	Father	Proband	Condition	Comment
P	John Smith	male	1970/02/05		M1	F1	yes	affected	Evaluated 2005/12/01
M1	Mary Smith (Brown)	female	1940/02/05		GM2	GF2		normal	
F1	Bill Smith	male	1938/04/03		GM1	GF1		affected	
GM1	Joan Smith	female	1902/07/01	1975/12/13				asymptomatic	
GF1	Joseph Smith	male	unknown	unknown				normal	
GF2	Jim Brown	male	1905/11/01					normal	
GM2	Lisa Brown	female	1910/03/03					normal	
S1	Rebecca Smith	female	1972/12/25		M1	F1		affected	
S2	Alexander Smith	male	1975/11/12		M1	F1		normal	
A1	Ann Gold (Smith)	female	1941/09/02		GM1	GF1		asymptomatic	Aunt of the proband
C1	Jenny Smith	female	1969/12/03		A1			affected	Cousin of the proband

Figure 7: Example of data file

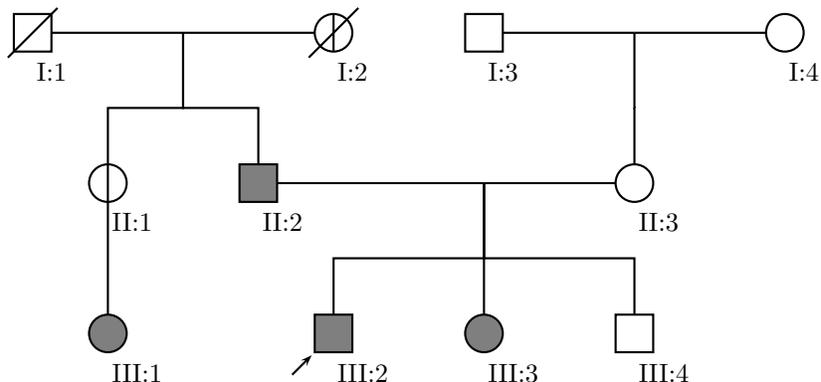
```

\begin{pspicture}(-8,-4)(6,4)
\rput(-6, 2){\pstPerson[male, normal, belowtext=I:1, deceased]{GF1}}
\rput(-4, 2){\pnode{GF1_m_GM1}}
\rput(-2, 2){\pstPerson[female, asymptomatic, belowtext=I:2, deceased]{GM1}}
\rput(0, 2){\pstPerson[male, normal, belowtext=I:3]{GF2}}
\rput(2, 2){\pnode{GF2_m_GM2}}
\rput(4, 2){\pstPerson[female, normal, belowtext=I:4]{GM2}}
\rput(-5, 0){\pstPerson[female, asymptomatic, belowtext=II:1]{A1}}
\rput(-3, 0){\pstPerson[male, affected, belowtext=II:2]{F1}}
\rput(0, 0){\pnode{F1_m_M1}}
\rput(2, 0){\pstPerson[female, normal, belowtext=II:3]{M1}}
\rput(-5, -2){\pstPerson[female, affected, belowtext=III:1]{C1}}
\rput(-2, -2){\pstPerson[male, affected, belowtext=III:2, proband]{P}}
\rput(0, -2){\pstPerson[female, affected, belowtext=III:3]{S1}}
\rput(2, -2){\pstPerson[male, normal, belowtext=III:4]{S2}}
\pstDescent{GF1_m_GM1}{A1}
\pstDescent{GF1_m_GM1}{F1}
\ncline{GF1_m_GM1}{GM1}
\ncline{GF1_m_GM1}{GF1}
\pstDescent{GF2_m_GM2}{M1}
\ncline{GF2_m_GM2}{GM2}
\ncline{GF2_m_GM2}{GF2}
\pstDescent{A1}{C1}
\pstDescent{F1_m_M1}{P}
\pstDescent{F1_m_M1}{S1}
\pstDescent{F1_m_M1}{S2}
\ncline{F1_m_M1}{M1}
\ncline{F1_m_M1}{F1}
\end{pspicture}

\begin{description}
\item[I:1] Joseph Smith; born: unknown; died: unknown.
\item[I:2] Joan Smith; born: 1902/07/01; died: 1975/12/13.
\item[I:3] Jim Brown; born: 1905/11/01.
\item[I:4] Lisa Brown; born: 1910/03/03.
\item[II:1] Ann Gold (Smith); born: 1941/09/02; Aunt of the proband.
\item[II:2] Bill Smith; born: 1938/04/03.
\item[II:3] Mary Smith (Brown); born: 1940/02/05.
\item[III:1] Jenny Smith; born: 1969/12/03; Cousin of the proband.
\item[III:2] John Smith; born: 1970/02/05; Evaluated 2005/12/01.
\item[III:3] Rebecca Smith; born: 1972/12/25.
\item[III:4] Alexander Smith; born: 1975/11/12.
\end{description}

```

**Figure 8:** Example of program output (data from Figure 7)



- I:1** Joseph Smith; born: unknown; died: unknown.
- I:2** Joan Smith; born: 1902/07/01; died: 1975/12/13.
- I:3** Jim Brown; born: 1905/11/01.
- I:4** Lisa Brown; born: 1910/03/03.
- II:1** Ann Gold (Smith); born: 1941/09/02; Aunt of the proband.
- II:2** Bill Smith; born: 1938/04/03.
- II:3** Mary Smith (Brown); born: 1940/02/05.
- III:1** Jenny Smith; born: 1969/12/03; Cousin of the proband.
- III:2** John Smith; born: 1970/02/05; Evaluated 2005/12/01.
- III:3** Rebecca Smith; born: 1972/12/25.
- III:4** Alexander Smith; born: 1975/11/12.

**Figure 9:** Example of the typeset pedigree (code from Figure 8)

as far as we can without intersection between them and the downward tree.

This process is shown in Figure 10. Obviously this algorithm converges and leads to typesetting the pedigree without intersections between the subtrees and subpedigrees.

The program is implemented in Perl and can process input in English or Russian, creating a pedigree legend in any of these languages (the Russian examples can be found in the manual (Veytsman and Akhmadeeva, 2006b)). It is quite straightforward to add language modules for any other language or script that  $\text{\TeX}$  can typeset.

#### 4 Installation notes

A couple of words about the installation of the packages.

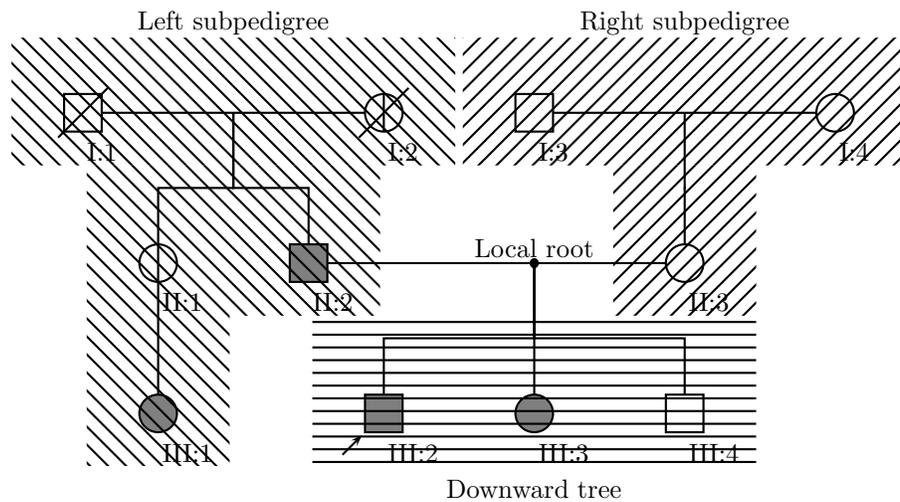
The installation of the  $\text{\TeX}$  part follows the usual guidelines (<http://www.tex.ac.uk/cgi-bin/textfaq2html?label=instpackages>).

The  $\text{\TeX}$  package depends on a number of other packages, which should be installed on your system. You need fresh versions of `pstricks` and `pst-xkey`. If you want to typeset the documentation you also need `pstricks-add`, but if you are satisfied with the PDF manual provided with the package, you might skip this step.

The Perl part includes the executable, library and manual pages. There is a `Makefile`, and in most cases the command `make install` suffices.

#### 5 Conclusions and future work

Our programs were written mostly as a proof of concept. Surprisingly (or unsurprisingly if we recall the



**Figure 10:** Subpedigrees and downward tree

properties of  $\text{\TeX}$ ) the typesetting quality of the output is rather high. The next logical step is to make them user-friendly, so any genetic specialist can use them without reading manual.

Since we cannot count on the medical and genetic personnel to have  $\text{\TeX}$  and Perl on their computers, we envision a “Pedigree Live” disk akin to  $\text{\TeX}$  Live: a CD with Perl and a subset of  $\text{\TeX}$  on it, which “just works” after inserting in the computer. This requires creating a cross-platform user interface and the selection of programs and tools to be included on a CD.

### Acknowledgements

One of the authors (LA) would like to thank Russian Foundation for Fundamental Research (travel grant 06-04-58811), Russian Federation President Council for Grants Supporting Young Scientists and Flagship Science Schools (grant MD-4245.2006.7) and TUG for support at the Practical  $\text{\TeX}$  2006 conference.

### References

- Bennett, Robin L., K. A. Steinhaus, S. B. Uhrich, C. K. O’Sullivan, R. G. Resta, D. Lochner-Doyle, D. S. Markei, V. Vincent, and J. Hamanishi. “Recommendations for Standardized Human Pedigree Nomenclature”. *Am. J. Hum. Genet.* **56**(3), 745–752, 1995.
- Di Battista, Giuseppe, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. An Alan R. Apt Book. Prentice Hall, New Jersey, 1999.
- Van Zandt, Timothy. *PSTricks: PostScript Macros for Generic  $\text{\TeX}$* , 1993. <http://tug.org/PSTricks/doc>.
- Veytsman, Boris, and L. Akhmadeeva. *Creating Medical Pedigrees with PSTricks and  $\text{\LaTeX}$* , 2006a. <http://ctan.tug.org/tex-archive/graphics/pstricks/contrib/pedigree/pst-pdgr>.
- Veytsman, Boris, and L. Akhmadeeva. *A Program For Automatic Pedigree Construction With pst-pdgr. User Manual and Algorithm Description*, 2006b. <http://ctan.tug.org/tex-archive/graphics/pstricks/contrib/pedigree/pedigree-perl>.

# Rolling your own Document Class: Using L<sup>A</sup>T<sub>E</sub>X to keep away from the Dark Side

Peter Flynn

Silmaril Consultants

peter (at) silmaril dot ie

<http://silmaril.ie/cgi-bin/blog>

## Abstract

Document classes in L<sup>A</sup>T<sub>E</sub>X provide automation to improve consistency, productivity, and accuracy in creating and maintaining documents, thereby avoiding the inefficiencies of wordprocessors. However, users who want to package their macros or applications as a document class are often put off by the apparent complexity of the sample classes in the standard distribution. This paper describes what the code in the `article` document class file does and suggests solutions to some of the popular requirements for changes.

## 1 Know thine enemy

One of the key features of T<sub>E</sub>X systems is the extensibility offered by re-usable pieces of programming called macros. Rudimentary macros exist in many text-handling packages (in fact they were at the heart of the first editors for markup applications), and some wordprocessors make use of general-purpose programming languages such as *Visual Basic* or *Java*; but only typesetters have dedicated languages to doing typesetting, and T<sub>E</sub>X's is by far the most accessible.

This has led to several large and well-known macro packages (L<sup>A</sup>T<sub>E</sub>X, ConT<sub>E</sub>Xt, Texinfo, Eplain, etc) which have all but taken the place of Knuth's original language as the end-user's primary interfaces. Most users now only have to use the macro commands of their chosen interface instead of having to write their own macros afresh or maintain a large private collection of personal macros.

This is not to say that there is no place for homebrew macros in plain T<sub>E</sub>X: some people have perfectly valid reasons for avoiding the aforementioned packages and continuing to use T<sub>E</sub>X in the raw. Using one of the above 'standards' does not always mean that you avoid raw T<sub>E</sub>X in your own code, because you may need some advanced operations which operate at a lower level than normal. It nevertheless remains true that the use of macros to perform groups of frequently-used functions provides a level of automation not found in most word-processing systems, and is a major factor in helping users become and remain more productive.

### 1.1 Standard document classes

The standard document classes installed with L<sup>A</sup>T<sub>E</sub>X (`article`, `report`, `book`, and `letter`) were written in a hybrid of L<sup>A</sup>T<sub>E</sub>X and plain T<sub>E</sub>X code. Sometimes this was because the function Lamport wanted was not worth writing a single-use L<sup>A</sup>T<sub>E</sub>X macro for; sometimes it is because (as Knuth describes in another context) "T<sub>E</sub>X is only 'half obedient' while these definitions are half finished" [4, p.352]; and sometimes because of the need mentioned above to perform lower-level functions. While the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> developers and maintainers have replaced much of the earlier plain T<sub>E</sub>X code with updated L<sup>A</sup>T<sub>E</sub>X equivalents, the code remains fairly dense and is not immediately obvious to the beginner; and the mix of syntax variants can be confusing to the user accustomed to the fairly small set of commands used for common L<sup>A</sup>T<sub>E</sub>X documents. Plain T<sub>E</sub>X itself has some 900 'control sequences' (commands) of which about 350 are 'primitives' (indivisible low-level operations), whereas many regular L<sup>A</sup>T<sub>E</sub>X users get by with some 20–30 commands, if even that.

Users who have started to write their own macros, or who have encountered the need to modify L<sup>A</sup>T<sub>E</sub>X's defaults for whatever reason, sometimes find the need to encapsulate their favourite format as a document class, governing the entire document, rather than just a package (style file) handling one or two specific features. In this paper we will dissect one of the common document classes and examine what the features and functions are.

### 1.2 Caveats

This paper uses the `article` class as the example. The `book` and `report` classes are structured very similarly

and the user who has examined the following sections should have no difficulty in identifying the differences.

The letter class, however, is a very different animal. It implements a vertically-centered format once common in typewritten letters but rarely seen nowadays, and has no provision for many of the features users expect to be able to find in a letter template. For this reason I do not refer any further to this format.

The ConT<sub>E</sub>Xt system implements a different and extensible set of document classes—including letters—in a radically different manner to L<sup>A</sup>T<sub>E</sub>X and has been discussed and presented extensively in recent years. The Eplain macros implement many of the features of the L<sup>A</sup>T<sub>E</sub>X internal mechanisms, but without imposing any document format at all, leaving the plain T<sub>E</sub>X user free to write those herself.

### 1.3 More background

The essential documentation to read before you start writing your own classes is *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> for class and package writers* [8] (available on all modern T<sub>E</sub>X installations by typing `texdoc clsguide`, and *The L<sup>A</sup>T<sub>E</sub>X Companion* [6, App:A.4]. These describe in detail the additional commands available to class and package authors. There are also some special declarations explained in *Companion* [6, p. 847]. The article by Hefferon [3] which I refer to later is a good example of how to build on an existing class. If you have to deal with an obsolete L<sup>A</sup>T<sub>E</sub>X 2.09 style file, there is an older paper in *TUGboat* [1].

## 2 Dissection of `article.cls`

In this example, we use the file from the T<sub>E</sub>X Live 2005 distribution (so the line numbers refer to that version only). Lines 1–53 are comments and are omitted here for brevity: they explain where the file came from and how it can be used. This is auto-generated because the document class and package files in the standard distributions of L<sup>A</sup>T<sub>E</sub>X are derived from master copies maintained in docT<sub>E</sub>X (`.dtx`) format [7], which combines documentation and L<sup>A</sup>T<sub>E</sub>X code in a single file, much in the same way that Knuth’s *WEB* system does for many programming languages [9].<sup>1</sup> A short explanation of the sources of the class files is in the *T<sub>E</sub>X FAQ* [2, label:ltxcmds].

<sup>1</sup> If you intend making your document class available to the rest of the L<sup>A</sup>T<sub>E</sub>X community (eg via CTAN), you should make it a docT<sub>E</sub>X document so that you can combine documentation with your code. Actually, you should probably be doing this anyway...

### 2.1 Version and identification

The first thing a document class or package must do is identify itself by name, and specify the oldest version of L<sup>A</sup>T<sub>E</sub>X with which it will work (it is assumed that it will therefore work with all later versions).

```

54 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
55 \ProvidesClass{article}
56 [2004/02/16 v1.4f
57 Standard LaTeX document class]

```

In your new document class file you should set the date and version to the earliest version you have tested your code with (probably your current version). The name of the document class being provided gets checked against the name requested in the `\documentclass` declaration, and L<sup>A</sup>T<sub>E</sub>X will give a warning if there is a discrepancy. You may provide a label for the class as well: this will appear in the log file. The linebreaks and indentation are for human readability only.

```

\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesClass{ladingbill} [2006/07/01 v0.01 Bill of Lading
specialist LaTeX document class]

```

### 2.2 Initial code and compatibility

On a number of occasions, classes define values as null or a default for later use, so that subsequent code won’t trip up as it would if they were undefined. In most cases you will probably need to keep the internal definitions (such as `\@ptsize` here) for use later on (see section 2.4.1 on p. 113).<sup>2</sup>

```

58 \newcommand{\@ptsize}{}
59 \newif\if@restonecol
60 \newif\if@titlepage
61 \@titlepagefalse

```

The conditionals `\if@restonecol` (which flags the restoration of one-column layout after using L<sup>A</sup>T<sub>E</sub>X’s built-in two-column usage, as distinct from using the `multicol` package) and `\if@titlepage` (which flags use of the separate title-page layout—set to false in the following line) are used in the default `\maketitle` command in section 2.4.4 on

<sup>2</sup> The use of the `@` sign may be unfamiliar to newcomers: in normal L<sup>A</sup>T<sub>E</sub>X it is classified as an ‘other’ character [4, p. 37]. This means it cannot be used as part of a control sequence (command) in your document. But in class and package files, L<sup>A</sup>T<sub>E</sub>X reclassifies it as a ‘letter’, and uses it in command definitions which are intended to be inaccessible to the normal user. Its use here indicates that the `\@ptsize` command is going to be given a value that the end-user should not be able to interfere with, or even know exists.

p. 116. If you're planning to rewrite `\maketitle` to your own design you may need to take these conditionals into account.<sup>3</sup>

If you are going to invoke additional packages to provide facilities needed by your options, use the `\RequirePackage` command here, before the options section. If the additional packages are unconnected with your option definitions, use the `\RequirePackage` command after the options are executed (see section 2.3.4 on p. 113).

## 2.3 Options

In an ideal world we wouldn't have to support obsolete versions of software, but the L<sup>A</sup>T<sub>E</sub>X defaults still allow v2.09-type `\documentstyle` declarations to be processed, with a warning. However, for a modern class file this is not necessary, so in your own class you can omit all the tests for `\ifcompatibility` and their `\else` and terminating `\fi` commands, here and throughout, leaving just the code that was in the `\else` blocks.

### 2.3.1 Paper sizes

How many paper size options you want to support in your class is entirely up to you. You should allow at least A4 and Letter for normal office work.

```

article.cls
62 \ifcompatibility\else
63 \DeclareOption{a4paper}
64   {\setlength\paperheight {297mm}%
65    \setlength\paperwidth  {210mm}}
66 \DeclareOption{a5paper}
67   {\setlength\paperheight {210mm}%
68    \setlength\paperwidth  {148mm}}
69 \DeclareOption{b5paper}
70   {\setlength\paperheight {250mm}%
71    \setlength\paperwidth  {176mm}}
72 \DeclareOption{letterpaper}
73   {\setlength\paperheight {11in}%
74    \setlength\paperwidth  {8.5in}}
75 \DeclareOption{legalpaper}
76   {\setlength\paperheight {14in}%
77    \setlength\paperwidth  {8.5in}}
78 \DeclareOption{executivepaper}
79   {\setlength\paperheight {10.5in}%
80    \setlength\paperwidth  {7.25in}}
81 \DeclareOption{landscape}
82   {\setlength\@tempdima  {\paperheight}%
83    \setlength\paperheight {\paperwidth}%
84    \setlength\paperwidth  {\@tempdima}}
85 \fi

```

<sup>3</sup> How much to cater for and how much to ignore will depend on how much your class deviates from the default. Many L<sup>A</sup>T<sub>E</sub>X users will expect to be able to use options like `twocolumn` and `titlepage` simply because they are available in the default classes. But if you are writing a much more prescriptive format, you may want to remove these options entirely, which means removing all references to conditional flags which depend on them.

In some cases you may be writing for a highly specific environment such as your own office or employer, where only one size is required. If so, just omit all the other declarations and add the one option to the `\ExecuteOptions` command (see section 2.3.4 on p. 113).

### 2.3.2 Type sizes and layout options

As mentioned above, the compatibility settings in this block can be removed in your own class, because modern class files use default option settings via the `\DeclareOption` command instead.

```

article.cls
86 \ifcompatibility
87   \renewcommand\@ptsize{0}
88 \else
89   \DeclareOption{10pt}{\renewcommand\@ptsize{0}}
90 \fi
91   \DeclareOption{11pt}{\renewcommand\@ptsize{1}}
92   \DeclareOption{12pt}{\renewcommand\@ptsize{2}}
93 \ifcompatibility\else
94   \DeclareOption{oneside}{\@twosidefalse \@mparswitchfalse}
95 \fi
96   \DeclareOption{twoside}{\@twosidetrue \@mparswitchtrue}
97   \DeclareOption{draft}{\setlength\overfullrule{5pt}}
98 \ifcompatibility\else
99   \DeclareOption{final}{\setlength\overfullrule{0pt}}
100 \fi
101   \DeclareOption{titlepage}{\@titlepagetrue}
102 \ifcompatibility\else
103   \DeclareOption{notitlepage}{\@titlepagefalse}
104 \fi
105 \ifcompatibility\else
106   \DeclareOption{onecolumn}{\@twocolumnfalse}
107 \fi
108   \DeclareOption{twocolumn}{\@twocolumntrue}
109   \DeclareOption{leqno}{\input{leqno.clo}}
110   \DeclareOption{fleqn}{\input{fleqn.clo}}
111   \DeclareOption{openbib}{\%
112     \AtEndOfPackage{\%
113       \renewcommand\@openbib@code{\%
114         \advance\leftmargin\bibindent
115         \itemindent -\bibindent
116         \listparindent \itemindent
117         \parsep \z@
118         }%
119       \renewcommand\newblock{\par}}%
120   }

```

The other options should probably be retained, as users may expect them to work, bearing in mind the comment about two-column and title-page settings above. Note that the `openbib` declaration is 10 lines long, and defers itself to end of the package

as a `\renewcommand` so that it doesn't conflict with the command being declared later.

As with paper sizes above, if your class only needs one specific size setup, just invoke it in `\ExecuteOptions`.

### 2.3.3 Your own options

Now is the time to add your own option declarations, if any. Note that option names have no backslash; otherwise the `\DeclareOption` command works the same way as the `\newcommand` command (but with no parameters).

Details of how to preserve the options of an existing class you are 'borrowing' via the `\LoadClass` command are discussed in section 3.1 on p. 122.

### 2.3.4 Applying options

Two commands control when the options are applied:

```

121 \ExecuteOptions{letterpaper,10pt,oneside,onecolumn,final}
122 \ProcessOptions

```

`\ExecuteOptions` applies all the options you specify in the argument, in order, as your selected defaults. The `\ProcessOptions` command then applies any options the user has selected in their `\documentclass` declaration.

## 2.4 Layout

A large number of size and shape settings depend on the selected point size (default 10pt, otherwise as selected in your options). The exact sizes of type chosen for all the different type-size commands are kept in three Class Option files, `size10.clo`, `size11.clo`, and `size12.clo`. There are some others available from CTAN, such as James Kilfiger's `size14.clo` for readers needing larger type editions, but the three mentioned above cover the vast majority of normal text setting.

If you are going to invoke additional packages that are unconnected with your option definitions, put the `\RequirePackage` commands here (see section 3.2 on p. 122). Be aware that some packages expect certain variables or definitions already to be present, so their invocation may need to be deferred until after everything else. In this case, enclose the `\RequirePackage` command in a `\AtEndOfPackage` or `\AtBeginDocument` command. This will invoke the package[s] at the specified point in processing, and thus avoid error messages or interference with code in the class file that has not yet been executed.

### 2.4.1 Type size

To invoke the right settings, the `\@ptsize` command is embedded in the argument to an `\input` command:

```

123 \input{size1\@ptsize.clo}
124 \setlength\lineskip{1\p@}
125 \setlength\normallineskip{1\p@}
126 \renewcommand\baselinestretch{}
127 \setlength\parskip{0\p@ \@plus \p@}

```

A number of basic settings are then made using the internal definition of a point (`\p@`). The class option files contain a lot of other size-specific settings as well as the font size specifications for the chosen body size.

**2.4.1.1 Identity and basic sizes** The class option files (we show `size10.clo` here) identify themselves in the same way as class files, but using the `\ProvidesFile` instead of `\ProvidesClass`.

```

54 \ProvidesFile{size10.clo}
55 [2004/02/16 v1.4f
56 Standard LaTeX file (size option)]
57 \renewcommand\normalsize{%
58 \setfontsize\normalsize\@xpt\@xipt
59 \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
60 \abovedisplayshortskip \z@ \@plus3\p@
61 \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
62 \belowdisplayskip \abovedisplayskip
63 \let\@listi\@listI}
64 \normalsize
65 \newcommand\small{%
66 \setfontsize\small\@ixpt{11}%
67 \abovedisplayskip 8.5\p@ \@plus3\p@ \@minus4\p@
68 \abovedisplayshortskip \z@ \@plus2\p@
69 \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
70 \def\@listi{\leftmargin\leftmargini
71 \topsep 4\p@ \@plus2\p@ \@minus2\p@
72 \parsep 2\p@ \@plus\p@ \@minus\p@
73 \itemsep \parsep}%
74 \belowdisplayskip \abovedisplayskip
75 }
76 \newcommand\footnotesize{%
77 \setfontsize\footnotesize\@viiipt{9.5}%
78 \abovedisplayskip 6\p@ \@plus2\p@ \@minus4\p@
79 \abovedisplayshortskip \z@ \@plus\p@
80 \belowdisplayshortskip 3\p@ \@plus\p@ \@minus2\p@
81 \def\@listi{\leftmargin\leftmargini
82 \topsep 3\p@ \@plus\p@ \@minus\p@
83 \parsep 2\p@ \@plus\p@ \@minus\p@
84 \itemsep \parsep}%
85 \belowdisplayskip \abovedisplayskip
86 }
87 \newcommand\scriptsize{\setfontsize\scriptsize\@viipt\@viiipt}
88 \newcommand\tiny{\setfontsize\tiny\@vpt\@vipt}
89 \newcommand\large{\setfontsize\large\@xiipt{14}}

```

```

90 \newcommand\Large{\setfontsize\Large\@xivpt{18}}
91 \newcommand\LARGE{\setfontsize\LARGE\@xviipt{22}}
92 \newcommand\huge{\setfontsize\huge\@xxpt{25}}
93 \newcommand\Huge{\setfontsize\Huge\@xxvpt{30}}

```

The first block defines the standard L<sup>A</sup>T<sub>E</sub>X sizes. These are named using roman numerals (eg `\@xiipt` for 12pt) because digits are not allowed in control sequence names. The more frequently-used sizes also define the display math spacing and the spacing for top-level lists (list definition names also use roman numerals like `\@listi`).

**2.4.1.2 Spacing** This section controls paragraph indentation (differing between one-column and two-column setting); the dimensions of the three ‘shortcut’ spacing commands (small, med, and big) but not the actual commands themselves, which are defined in L<sup>A</sup>T<sub>E</sub>X itself; and some top-of-page and bottom-of-page spacing settings (normally reset using the `geometry` package).

```

----- size10.clo -----
94 \if@twocolumn
95   \setlength\parindent{1em}
96 \else
97   \setlength\parindent{15\p@}
98 \fi
99 \setlength\smallskipamount{3\p@ \@plus 1\p@ \@minus 1\p@}
100 \setlength\medskipamount{6\p@ \@plus 2\p@ \@minus 2\p@}
101 \setlength\bigskipamount{12\p@ \@plus 4\p@ \@minus 4\p@}
102 \setlength\headheight{12\p@}
103 \setlength\headsep {25\p@}
104 \setlength\topskip {10\p@}
105 \setlength\footskip{30\p@}
106 \if@compatibility \setlength\maxdepth{4\p@} \else
107 \setlength\maxdepth{.5\topskip} \fi

```

**2.4.1.3 Text area** Text width and text height are set to depend on the columnar setting and a multiple of line-heights respectively.

```

----- size10.clo -----
108 \if@compatibility
109   \if@twocolumn
110     \setlength\textwidth{410\p@}
111   \else
112     \setlength\textwidth{345\p@}
113   \fi
114 \else
115   \setlength\@tempdima{\paperwidth}
116   \addtolength\@tempdima{-2in}
117   \setlength\@tempdimb{345\p@}
118   \if@twocolumn
119     \ifdim\@tempdima>2\@tempdimb\relax
120     \setlength\textwidth{2\@tempdimb}

```

```

121   \else
122     \setlength\textwidth{\@tempdima}
123   \fi
124 \else
125   \ifdim\@tempdima>\@tempdimb\relax
126     \setlength\textwidth{\@tempdimb}
127   \else
128     \setlength\textwidth{\@tempdima}
129   \fi
130 \fi
131 \fi
132 \if@compatibility\else
133   \settopoint\textwidth
134 \fi
135 \if@compatibility
136   \setlength\textheight{43\baselineskip}
137 \else
138   \setlength\@tempdima{\paperheight}
139   \addtolength\@tempdima{-2in}
140   \addtolength\@tempdima{-1.5in}
141   \divide\@tempdima\baselineskip
142   \@tempcnta=\@tempdima
143   \setlength\textheight{\@tempcnta\baselineskip}
144 \fi
145 \addtolength\textheight{\topskip}

```

(The compatibility-mode settings were absolute values in points.) As with paper size and type size, you can preselect exact values for the text area and margins (see next section) using the `geometry` package.

**2.4.1.4 Page margins** Margins also depend on the column settings, and on the one-side/two-side setting.

```

----- size10.clo -----
146 \if@twocolumn
147   \setlength\marginparsep {10\p@}
148 \else
149   \setlength\marginparsep{11\p@}
150 \fi
151 \setlength\marginparpush{5\p@}
152 \if@compatibility
153   \if@twoside
154     \setlength\oddsidemargin {44\p@}
155     \setlength\evensidemargin {82\p@}
156     \setlength\marginparwidth {107\p@}
157   \else
158     \setlength\oddsidemargin {63\p@}
159     \setlength\evensidemargin {63\p@}
160     \setlength\marginparwidth {90\p@}
161   \fi
162 \if@twocolumn
163   \setlength\oddsidemargin {30\p@}

```

```

164 \setlength\evensidemargin {30\p@}
165 \setlength\marginparwidth {48\p@}
166 \fi
167 \else
168 \if@twoside
169 \setlength\@tempdima {\paperwidth}
170 \addtolength\@tempdima {-\textwidth}
171 \setlength\oddsidemargin {.4\@tempdima}
172 \addtolength\oddsidemargin {-1in}
173 \setlength\marginparwidth {.6\@tempdima}
174 \addtolength\marginparwidth {-\marginparsep}
175 \addtolength\marginparwidth {-0.4in}
176 \else
177 \setlength\@tempdima {\paperwidth}
178 \addtolength\@tempdima {-\textwidth}
179 \setlength\oddsidemargin {.5\@tempdima}
180 \addtolength\oddsidemargin {-1in}
181 \setlength\marginparwidth {.5\@tempdima}
182 \addtolength\marginparwidth {-\marginparsep}
183 \addtolength\marginparwidth {-0.4in}
184 \addtolength\marginparwidth {-.4in}
185 \fi
186 \ifdim \marginparwidth >2in
187 \setlength\marginparwidth{2in}
188 \fi
189 \@settopoint\oddsidemargin
190 \@settopoint\marginparwidth
191 \setlength\evensidemargin {\paperwidth}
192 \addtolength\evensidemargin{-2in}
193 \addtolength\evensidemargin{-\textwidth}
194 \addtolength\evensidemargin{-\oddsidemargin}
195 \@settopoint\evensidemargin
196 \fi
197 \if@compatibility
198 \setlength\topmargin{27pt}
199 \else
200 \setlength\topmargin{\paperheight}
201 \addtolength\topmargin{-2in}
202 \addtolength\topmargin{-\headheight}
203 \addtolength\topmargin{-\headsep}
204 \addtolength\topmargin{-\textheight}
205 \addtolength\topmargin{-\footskip}% this might be wrong
206 \addtolength\topmargin{-.5\topmargin}
207 \@settopoint\topmargin
208 \fi

```

Again, the compatibility-mode settings are absolute, whereas the modern defaults are computed.

**2.4.1.5 Footnote space** Spacing for footnotes and floats is flexible (plus and minus a certain amount) so that the page-breaking routine doesn't become too rigid.

```

size10.clo
209 \setlength\footnotesep{6.65\p@}
210 \setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}
211 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
212 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
213 \setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}
214 \setlength\dblfloatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
215 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
216 \setlength\@fptop{0\p@ \@plus 1fil}
217 \setlength\@fpsep{8\p@ \@plus 2fil}
218 \setlength\@fpbot{0\p@ \@plus 1fil}
219 \setlength\@dblftop{0\p@ \@plus 1fil}
220 \setlength\@dblfpsep{8\p@ \@plus 2fil}
221 \setlength\@dblfpbot{0\p@ \@plus 1fil}
222 \setlength\partopsep{2\p@ \@plus 1\p@ \@minus 1\p@}

```

**2.4.1.6 Lists** Finally, for the values dependent on type size, the dimensions of lists are set. As mentioned above, names are fabricated using roman numerals (i to vi).

```

size10.clo
223 \def\@listi{\leftmargin\leftmargini
224 \parsep 4\p@ \@plus2\p@ \@minus\p@
225 \topsep 8\p@ \@plus2\p@ \@minus4\p@
226 \itemsep4\p@ \@plus2\p@ \@minus\p@}
227 \let\@listI\@listi
228 \@listi
229 \def\@listii {\leftmargin\leftmarginii
230 \labelwidth\leftmarginii
231 \advance\labelwidth-\labelsep
232 \topsep 4\p@ \@plus2\p@ \@minus\p@
233 \parsep 2\p@ \@plus\p@ \@minus\p@
234 \itemsep \parsep}
235 \def\@listiii{\leftmargin\leftmarginiii
236 \labelwidth\leftmarginiii
237 \advance\labelwidth-\labelsep
238 \topsep 2\p@ \@plus\p@\@minus\p@
239 \parsep \z@
240 \partopsep \p@ \@plus\z@ \@minus\p@
241 \itemsep \topsep}
242 \def\@listiv {\leftmargin\leftmarginiv
243 \labelwidth\leftmarginiv
244 \advance\labelwidth-\labelsep}
245 \def\@listv {\leftmargin\leftmarginv
246 \labelwidth\leftmarginv
247 \advance\labelwidth-\labelsep}
248 \def\@listvi {\leftmargin\leftmarginvi
249 \labelwidth\leftmarginvi
250 \advance\labelwidth-\labelsep}
251 \endinput

```

### 2.4.2 Spacing penalties

Three penalties are set which get invoked in various decisions on paragraph-breaking. You probably don't want to change these unless you are doing deep surgery.

```

128 \lowpenalty 51
129 \medpenalty 151
130 \highpenalty 301
131 \setcounter{topnumber}{2}
132 \renewcommand\topfraction{.7}
133 \setcounter{bottomnumber}{1}
134 \renewcommand\bottomfraction{.3}
135 \setcounter{totalnumber}{3}
136 \renewcommand\textfraction{.2}
137 \renewcommand\floatpagefraction{.5}
138 \setcounter{dbltopnumber}{2}
139 \renewcommand\dbltopfraction{.7}
140 \renewcommand\dblfloatpagefraction{.5}

```

The fractions and numbers refer to the proportions of the page that can be taken up by figures and tables, and the number of floats allowed, when calculating the location of floats.

### 2.4.3 Running heads

Depending on the imposition (one-sided or two-sided), the default running heads are specified as in the original L<sup>A</sup>T<sub>E</sub>X manual [5].

```

141 \if@twoside
142   \def\ps@headings{%
143     \let\@oddfoot\@empty\let\@evenfoot\@empty
144     \def\@evenhead{\thepage\hfil\slshape\leftmark}%
145     \def\@oddhead{\slshape\rightmark\hfil\thepage}%
146     \let\@mkboth\markboth
147     \def\sectionmark##1{%
148       \markboth {\MakeUppercase{
149         \ifnum \c@secnumdepth >\z@
150           \thesection\quad
151           \fi
152         ##1}}{}}%
153     \def\subsectionmark##1{%
154       \markright {%
155         \ifnum \c@secnumdepth >\@ne
156           \thesubsection\quad
157           \fi
158         ##1}}
159   \else
160     \def\ps@headings{%
161       \let\@oddfoot\@empty
162       \def\@oddhead{\slshape\rightmark\hfil\thepage}%
163       \let\@mkboth\markboth
164       \def\sectionmark##1{%

```

```

165     \markright {\MakeUppercase{
166       \ifnum \c@secnumdepth >\@m@ne
167         \thesection\quad
168         \fi
169       ##1}}}}
170   \fi
171 \def\ps@myheadings{%
172   \let\@oddfoot\@empty\let\@evenfoot\@empty
173   \def\@evenhead{\thepage\hfil\slshape\leftmark}%
174   \def\@oddhead{\slshape\rightmark\hfil\thepage}%
175   \let\@mkboth\@gobbletwo
176   \let\sectionmark\@gobble
177   \let\subsectionmark\@gobble
178 }

```

In many cases it may be preferable to use the fancyhdr package instead. This lets you specify a very wide range of header and footer layouts, with left/right switching for double-sided work.

### 2.4.4 Titling

This is possibly the first big change you'll need to make. There are two `\maketitle` commands defined, one for use on a separate title page (without facilities for attribution), and one for normal use on the starting page (with attributions, and allowing for two columns, using the `\@maketitle` command as well). Both are controlled by the `\if@titlepage` switch.

```

179 \if@titlepage
180   \newcommand\maketitle{\begin{titlepage}%
181     \let\footnotesize\small
182     \let\footnoterule\relax
183     \let \footnote \thanks
184     \null\vfil
185     \vskip 60\p@
186     \begin{center}%
187       {\LARGE \@title \par}%
188       \vskip 3em%
189       {\large
190         \lineskip .75em%
191         \begin{tabular}[t]{c}%
192           \@author
193           \end{tabular}\par}%
194       \vskip 1.5em%
195       {\large \@date \par}% % Set date in \large size.
196     \end{center}\par
197     \@thanks
198     \vfil\null
199   \end{titlepage}%
200   \setcounter{footnote}{0}%
201   \global\let\thanks\relax
202   \global\let\maketitle\relax
203   \global\let\@thanks\@empty
204   \global\let\@author\@empty
205   \global\let\@date\@empty
206   \global\let\@title\@empty
207   \global\let\title\relax
208   \global\let\author\relax
209   \global\let\date\relax

```

```

210 \global\let\and\relax
211 }
212 \else
213 \newcommand\maketitle{\par
214 \begin{group
215 \renewcommand\thefootnote{\@fnsymbol\c@footnote}%
216 \def\@makefnmark{\rlap{\@textsuperscript{\normalfont\@thefnmark}}}%
217 \long\def\@makefnmark#1{\parindent 1em\noindent
218 \hb@xt@1.8em{\
219 \hss\@textsuperscript{\normalfont\@thefnmark}}#1}%
220 \if@twocolumn
221 \ifnum \col@number=\@ne
222 \maketitle
223 \else
224 \twocolumn[\maketitle]%
225 \fi
226 \else
227 \newpage
228 \global\@topnum\z@ % Prevents figures from going at top of page.
229 \maketitle
230 \fi
231 \thispagestyle{plain}\@thanks
232 \endgroup
233 \setcounter{footnote}{0}%
234 \global\let\thanks\relax
235 \global\let\maketitle\relax
236 \global\let\@maketitle\relax
237 \global\let\@thanks\@empty
238 \global\let\@author\@empty
239 \global\let\@date\@empty
240 \global\let\@title\@empty
241 \global\let\title\relax
242 \global\let\author\relax
243 \global\let\date\relax
244 \global\let\and\relax
245 }
246 \def\@maketitle{%
247 \newpage
248 \null
249 \vskip 2em%
250 \begin{center}%
251 \let \footnote \thanks
252 {\LARGE \@title \par}%
253 \vskip 1.5em%
254 {\large
255 \lineskip .5em%
256 \begin{tabular}[t]{c}%
257 \@author
258 \end{tabular}\par}%
259 \vskip 1em%
260 {\large \@date}%
261 \end{center}%
262 \par
263 \vskip 1.5em}
264 \fi

```

In all of these you can redefine the size, location, and spacing of the three basic titling elements, `\@title`, `\@author`, and `\@date`. (`\author` itself is defined as part of the L<sup>A</sup>T<sub>E</sub>X core.) If you are not using two-column setting, or a title-page option, you could replace the whole lot with a single `\renewcommand{\maketitle}{...}` of your own design.

You can also make up your own additional elements, for example an optional subtitle:

```

\def\@subtitle{\relax}
\newcommand{\subtitle}[1]{\gdef\@subtitle{#1}}
\renewcommand{\maketitle}{
\begin{titlepage}
\huge\@author\par
\Large\@title\par
\if\@subtitle\relax\else\large\@subtitle\par\fi
\normalsize\@date\par
\end{titlepage}
}

```

This lets the phantom `\@subtitle` exist unused, set to `\relax` unless an author explicitly uses the `\subtitle` command, because the titling routine can test whether it is still set to `\relax`, and if not, format it accordingly. This technique can be used to add many of the items of metadata used by publishers, such as author affiliations, email and web addresses, and dates of submission.

## 2.5 Structure

Unless you are doing a very rigid class for data-handling, you probably want to keep the basic sectional structures for normal continuous text as they are, and only change the formatting.

```

----- article.cls -----
265 \setcounter{secnumdepth}{3}
266 \newcounter {part}
267 \newcounter {section}
268 \newcounter {subsection}[section]
269 \newcounter {subsubsection}[subsection]
270 \newcounter {paragraph}[subsubsection]
271 \newcounter {subparagraph}[paragraph]
272 \renewcommand \thepart {\@Roman\c@part}
273 \renewcommand \thesection {\@arabic\c@section}
274 \renewcommand\thesubsection {\thesection.\@arabic\c@subsection}
275 \renewcommand\thesubsubsection{\thesubsection.\@arabic\c@subsubsection}
276 \renewcommand\theparagraph {\thesubsubsection.\@arabic\c@paragraph}
277 \renewcommand\thesubparagraph {\theparagraph.\@arabic\c@subparagraph}
278 \newcommand\part{}
279 \if@noskipsec \leavevmode \fi
280 \par
281 \addvspace{4ex}%
282 \@afterindentfalse
283 \secdef\part\@spart}

```

The `\part` command is defined separately, as it operates like `\chapter` in other classes, with more space and a prefix (the `book` and `report` classes define a separate `\chapter` command).

```

----- article.cls -----
285 \def\@part[#1]#2{%
286 \ifnum \c@secnumdepth >\m@ne
287 \refstepcounter{part}%
288 \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
289 \else

```

```

290 \addcontentsline{toc}{part}{#1}%
291 \fi
292 {\parindent \z@ \raggedright
293 \interlinepenalty \OM
294 \normalfont
295 \ifnum \c@secnumdepth >\m@ne
296 \Large\bfseries \partname\nobreakspace\thepart
297 \par\nobreak
298 \fi
299 \huge \bfseries #2%
300 \markboth{}{\par}%
301 \nobreak
302 \vskip 3ex
303 \@afterheading}
304 \def\@spart#1{%
305 {\parindent \z@ \raggedright
306 \interlinepenalty \OM
307 \normalfont
308 \huge \bfseries #1\par}%
309 \nobreak
310 \vskip 3ex
311 \@afterheading}

```

The sectional formatting is one of the most common features of a document class that need to change. Details of the operation of the `\@startsection` command are in the L<sup>A</sup>T<sub>E</sub>X manual [5] if you want to do a complete rewrite, but in many cases one of the packages like `sectsty` can be used to change fonts or spacing without you having to redo everything from scratch.

```

article.cls
312 \newcommand\section{\@startsection {section}{1}{\z@}%
313 {-3.5ex \@plus -1ex \@minus -.2ex}%
314 {2.3ex \@plus .2ex}%
315 {\normalfont\Large\bfseries}}
316 \newcommand\subsection{\@startsection{subsection}{2}{\z@}%
317 {-3.25ex\@plus -1ex \@minus -.2ex}%
318 {1.5ex \@plus .2ex}%
319 {\normalfont\large\bfseries}}
320 \newcommand\subsubsection{\@startsection{subsubsection}{3}{\z@}%
321 {-3.25ex\@plus -1ex \@minus -.2ex}%
322 {1.5ex \@plus .2ex}%
323 {\normalfont\normalsize\bfseries}}
324 \newcommand\paragraph{\@startsection{paragraph}{4}{\z@}%
325 {3.25ex \@plus 1ex \@minus .2ex}%
326 {-1em}%
327 {\normalfont\normalsize\bfseries}}
328 \newcommand\subparagraph{\@startsection{subparagraph}{5}{\parindent}%
329 {3.25ex \@plus 1ex \@minus .2ex}%
330 {-1em}%
331 {\normalfont\normalsize\bfseries}}

```

## 2.6 Indents and margins

In this section the class file defines the internal margins set around block elements like lists. For controlling lists, L<sup>A</sup>T<sub>E</sub>X provides four levels of indentation. As explained earlier, because digits are not permit-

ted in command names, all these parameters end in the Roman-numeral equivalents.

```

article.cls
332 \if@twocolumn
333 \setlength\leftmargini {2em}
334 \else
335 \setlength\leftmargini {2.5em}
336 \fi
337 \leftmargin \leftmargini
338 \setlength\leftmarginii {2.2em}
339 \setlength\leftmarginiii {1.87em}
340 \setlength\leftmarginiv {1.7em}
341 \if@twocolumn
342 \setlength\leftmarginv { .5em}
343 \setlength\leftmarginvi { .5em}
344 \else
345 \setlength\leftmarginv {1em}
346 \setlength\leftmarginvi {1em}
347 \fi
348 \setlength \labelsep { .5em}
349 \setlength \labelwidth{\leftmargini}
350 \addtolength\labelwidth{-\labelsep}
351 \@beginparpenalty -\@lowpenalty
352 \@endparpenalty -\@lowpenalty
353 \@itempenalty -\@lowpenalty
354 \renewcommand\theenumi{\@arabic\c@enumi}
355 \renewcommand\theenumii{\@alph\c@enumii}
356 \renewcommand\theenumiii{\@roman\c@enumiii}
357 \renewcommand\theenumiv{\@Alph\c@enumiv}
358 \newcommand\labelenumi{\theenumi.}
359 \newcommand\labelenumii{\theenumii)}
360 \newcommand\labelenumiii{\theenumiii.}
361 \newcommand\labelenumiv{\theenumiv.}
362 \renewcommand\p@enumii{\theenumi}
363 \renewcommand\p@enumiii{\theenumii}
364 \renewcommand\p@enumiv{\p@enumiii\theenumiii}
365 \newcommand\labelitemi{\textbullet}
366 \newcommand\labelitemii{\normalfont\bfseries \textendash}
367 \newcommand\labelitemiii{\textasteriskcentered}
368 \newcommand\labelitemiv{\textperiodcentered}
369 \newenvironment{description}
370 {\list{}{\labelwidth\z@ \itemindent-\leftmargin
371 \let\makelabel\descriptionlabel}}
372 {\endlist}
373 \newcommand*\descriptionlabel[1]{\hspace\labelsep
374 \normalfont\bfseries #1}

```

The variables and their meaning are described in more detail in the L<sup>A</sup>T<sub>E</sub>X manual [5] and the *Companion* [6], but essentially:

`\leftmargin $rr$`  are the list level indentations from outer page margin to the start of the text;  
`\labelsep` is the space between the number or bullet and the start of the text;

`\labelwidth` is how much space to allow for the numbering or bulleting;

`\theenumrr` controls the style of numbering;

`\labellenumrr` controls the style of bulleting.

In all these cases, you can remove the conditional code surrounding the variants for two-column work, and have just one setting, if you are not going to provide for two-column setting.

The `\description` environment works slightly differently: the `\makelabel` command is equated to a `\descriptionlabel` command to indent and format the item label. This is easily redefined, for example to make the labels use the sans-serif font instead of the default roman typeface, and add an automatic em-rule afterwards:

```
\renewcommand*\descriptionlabel[1]{
  \hspace\labelsep
  \relax\sffamily{\bfseries #1}~---\space
  \ignorespaces}
```

## 2.7 Abstract

The default abstract is formatted differently according to where it appears: on the first page or on a page by itself after a separate title page.

```

----- article.cls -----
375 \if@titlepage
376   \newenvironment{abstract}{%
377     \titlepage
378     \null\vfil
379     \@beginparpenalty\@lowpenalty
380     \begin{center}%
381       \bfseries \abstractname
382       \@endparpenalty\@M
383     \end{center}}%
384   {\par\vfil\null\endtitlepage}
385 \else
386   \newenvironment{abstract}{%
387     \if@twocolumn
388       \section*{\abstractname}%
389     \else
390       \small
391       \begin{center}%
392         {\bfseries \abstractname\vspace{-.5em}\vspace{\z@}}%
393       \end{center}%
394       \quotation
395     \fi}
396   {\if@twocolumn\else\endquotation\fi}
397 \fi
```

One common requirement is for the Abstract formatting to follow the pattern of a subsection when it appears on a separate page, eg

```
\newenvironment{abstract}{%
  \titlepage
  \subsection*{\abstractname}}%
{\par\vfil\null\endtitlepage}
```

Some styles require turning off the initial indentation when the abstract is on the first page, for consistency with the default Anglo-American style used in sections:

```
\newenvironment{abstract}{%
  \if@twocolumn
    \section*{\abstractname}%
  \else
    \small
    \begin{center}%
      {\bfseries \abstractname\vspace{-.5em}
        \vspace{\z@}}%
    \end{center}%
    \quotation\noindent\ignorespaces
  \fi}
{\if@twocolumn\else\endquotation\fi}
```

Note that if you will be adding to an existing class in the manner described in section 3.1 on p. 122, these last two examples will use the `\renewenvironment` command instead.

## 2.8 Structural elements

The default classes contain some rudimentary environments for verse and quotations, and a compatibility setting for L<sup>A</sup>T<sub>E</sub>X 2.09 users, which can be omitted from new classes (make sure you keep one definition of the `\titlepage` environment, though!

```

----- article.cls -----
398 \newenvironment{verse}
399   {\let\\\@centercr
400    \list{}{\itemsep \z@
401     \itemindent -1.5em%
402     \listparindent\itemindent
403     \rightmargin \leftmargin
404     \advance\leftmargin 1.5em}%
405   \item\relax}
406 \endlist}
407 \newenvironment{quotation}
408   {\list{}{\listparindent 1.5em%
409    \itemindent \listparindent
410    \rightmargin \leftmargin
411    \parsep \z@ \@plus\p@}%
412   \item\relax}
413 \endlist}
414 \newenvironment{quote}
415   {\list{}{\rightmargin\leftmargin}%
416   \item\relax}
```

```

417         {\endlist}
418 \ifcompatibility
419 \newenvironment{titlepage}
420   {%
421     \if@twocolumn
422       \@restonecoltrue\onecolumn
423     \else
424       \@restonecolfalse\newpage
425     \fi
426     \thispagestyle{empty}%
427     \setcounter{page}\z@
428   }%
429   {\if@restonecol\twocolumn \else \newpage \fi
430   }
431 \else
432 \newenvironment{titlepage}
433   {%
434     \if@twocolumn
435       \@restonecoltrue\onecolumn
436     \else
437       \@restonecolfalse\newpage
438     \fi
439     \thispagestyle{empty}%
440     \setcounter{page}\@ne
441   }%
442   {\if@restonecol\twocolumn \else \newpage \fi
443   \if@twoside\else
444     \setcounter{page}\@ne
445   \fi
446   }
447 \fi
448 \newcommand\appendix{\par
449 \setcounter{section}{0}%
450 \setcounter{subsection}{0}%
451 \gdef\thesection{\@Alph\c@section}}

```

The `quotation` environment is another which benefits from the removal of the initial indentation:

```

\newenvironment{quotation}
  {\list{}{\listparindent 1.5em%
  \itemindent \z@
  \rightmargin \leftmargin
  \parsep \z@ \@plus\p@}%
  \item\relax}
{\endlist}

```

For the reasons noted in section 2.7 on p. 119, this may need to be a `\renewcommand`.

This section ends with a definition for `\appendix` which switches the `\section` settings to produce labels with A, B, C, etc instead of 1, 2, 3.

## 2.9 Figures and tables

These are controlled by a number of dimensions which you may already be familiar with, such as `\tabcolsep` for the gap between table columns. The `\fboxsep` and `\fboxrule` dimensions control the gap and rule thickness around boxed text.

```

----- article.cls -----
452 \setlength\arraycolsep{5\p@}
453 \setlength\tabcolsep{6\p@}
454 \setlength\arrayrulewidth{.4\p@}
455 \setlength\doublerulesep{2\p@}
456 \setlength\tabbingsep{\labelsep}
457 \skip\@mpfootins = \skip\footins
458 \setlength\fboxsep{3\p@}
459 \setlength\fboxrule{.4\p@}
460 \renewcommand \theequation {\@arabic\c@equation}
461 \newcounter{figure}
462 \renewcommand \thefigure {\@arabic\c@figure}
463 \def\fps@figure{tbp}
464 \def\ftype@figure{1}
465 \def\ext@figure{lof}
466 \def\fnun@figure{\figurename\nobreakspace\thefigure}
467 \newenvironment{figure}
468   {\@float{figure}}
469   {\end@float}
470 \newenvironment{figure*}
471   {\@dblfloat{figure}}
472   {\end@dblfloat}
473 \newcounter{table}
474 \renewcommand\thetable{\@arabic\c@table}
475 \def\fps@table{tbp}
476 \def\ftype@table{2}
477 \def\ext@table{lot}
478 \def\fnun@table{\tablename\nobreakspace\thetable}
479 \newenvironment{table}
480   {\@float{table}}
481   {\end@float}
482 \newenvironment{table*}
483   {\@dblfloat{table}}
484   {\end@dblfloat}
485 \newlength\abovecaptionskip
486 \newlength\belowcaptionskip
487 \setlength\abovecaptionskip{10\p@}
488 \setlength\belowcaptionskip{0\p@}
489 \long\def\@makecaption#1#2{%
490   \vskip\abovecaptionskip
491   \sbox\@tempboxa{#1: #2}%
492   \ifdim \wd\@tempboxa >\hsize
493     #1: #2\par
494   \else
495     \global \@minipagefalse
496     \hbext@\hsize{\hfil\box\@tempboxa\hfil}%

```

```
497 \fi
498 \vskip\belowcaptionskip}
```

At the end of this section is the `\makecaption` command, another popular candidate for redesign, but consider also using the `ccaption` package.

## 2.10 Legacy support

The obsolescent commands `\rm`, `\it`, `\bf`, etc are declared here to function as their modern equivalents.

```
----- article.cls -----
499 \DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
500 \DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
501 \DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
502 \DeclareOldFontCommand{\bf}{\normalfont\bfseries}{\mathbf}
503 \DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
504 \DeclareOldFontCommand{\sl}{\normalfont\slshape}{\@nomath\sl}
505 \DeclareOldFontCommand{\sc}{\normalfont\scshape}{\@nomath\sc}
506 \DeclareRobustCommand*\cal{\@fontswitch\relax\mathcal}
507 \DeclareRobustCommand*\mit{\@fontswitch\relax\mathnormal}
```

## 2.11 Table of contents

The Table of Contents section starts with some commands which evaluate to dimensions, plus the `\tableofcontents` command itself.

```
----- article.cls -----
508 \newcommand\@pnumwidth{1.55em}
509 \newcommand\@tocmarg{2.55em}
510 \newcommand\@dotsep{4.5}
511 \setcounter{tocdepth}{3}
512 \newcommand\tableofcontents{%
513   \section*{\contentsname
514     \@mkboth{%
515       \MakeUppercase\contentsname}{\MakeUppercase\contentsname}}%
516   \@starttoc{toc}%
517 }
518 \newcommand*\l@part[2]{%
519   \ifnum \c@tocdepth >2\relax
520     \addpenalty\@secpenalty
521     \advvspace{2.25em \@plus\p@}%
522     \setlength\@tempdima{3em}%
523     \begingroup
524       \parindent \z@ \rightskip \@pnumwidth
525       \parfillskip -\@pnumwidth
526       {\leavevmode
527         \large \bfseries #1\hfil \hb@xt@\@pnumwidth{\hss #2}}\par
528         \nobreak
529         \ifcompatibility
530           \global\@nobreaktrue
531           \everypar{\global\@nobreakfalse\everypar{}}%
532         \fi
533       \endgroup
534     \fi}
535 \newcommand*\l@section[2]{%
536   \ifnum \c@tocdepth >\z@
537     \addpenalty\@secpenalty
538     \advvspace{1.0em \@plus\p@}%
539     \setlength\@tempdima{1.5em}%
540     \begingroup
541       \parindent \z@ \rightskip \@pnumwidth
```

```

542     \parfillskip -\@pnumwidth
543     \leavevmode \bfseries
544     \advance\leftskip\@tempdima
545     \hskip -\leftskip
546     #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss #2}\par
547   \endgroup
548   \fi}
549 \newcommand*\l@subsection{\@dottedtocline{2}{1.5em}{2.3em}}
550 \newcommand*\l@subsubsection{\@dottedtocline{3}{3.8em}{3.2em}}
551 \newcommand*\l@paragraph{\@dottedtocline{4}{7.0em}{4.1em}}
552 \newcommand*\l@subparagraph{\@dottedtocline{5}{10em}{5em}}
553 \newcommand\listoffigures{%
554   \section*{\listfigurename}%
555   \@mkboth{\MakeUppercase\listfigurename}%
556             {\MakeUppercase\listfigurename}}
557 \starttoc{lof}%
558 }
559 \newcommand*\l@figure{\@dottedtocline{1}{1.5em}{2.3em}}
560 \newcommand\listoftables{%
561   \section*{\listtablename}%
562   \@mkboth{%
563     \MakeUppercase\listtablename}%
564     {\MakeUppercase\listtablename}}
565 \starttoc{lot}%
566 }
567 \let\l@table\l@figure
```

There are `\l@ttt` commands (`\l@part`, `\l@section`, etc) which produce the ToC lines from the `.aux` file. The List of Tables and List of Figures are implemented in the same way as the ToC. As with other features, consider the `tocloft` package for common modifications.

## 2.12 Bibliography and index

Bibliography styles themselves are implemented in `.bst` files, but the style of the section can be changed here, including indentation and spacing.

```
----- article.cls -----
568 \newdimen\bibindent
569 \setlength\bibindent{1.5em}
570 \newenvironment{thebibliography}[1]
571   {\section*{\refname}%
572     \@mkboth{\MakeUppercase\refname}{\MakeUppercase\refname}%
573     \list{\@biblabel{\@arabic\c@enumiv}}%
574           {\settowidth\labelwidth{\@biblabel{#1}}%
575             \leftmargin\labelwidth
576             \advance\leftmargin\labelsep
577             \@openbib@code
578             \usecounter{enumiv}%
579             \let\p@enumiv\@empty
580             \renewcommand\theenumiv{\@arabic\c@enumiv}}%
581   \sloppy
582   \clubpenalty4000
583   \@clubpenalty \clubpenalty
584   \widowpenalty4000%
585   \sfcode'\.\@m}
586   {\def\noitemerr
587     {\@latex@warning{Empty 'thebibliography' environment}}%
588   \endlist}
589 \newcommand\newblock{\hskip .11em\@plus.33em\@minus.07em}
590 \let\@openbib@code\@empty
591 \newenvironment{theindex}
592   {\if@twocolumn
593     \@restonecolfalse
```

```

594     \else
595         \@restonecoltrue
596     \fi
597     \twocolumn[\section*{\indexname}]%
598     \@mkboth{\MakeUppercase\indexname}%
599             {\MakeUppercase\indexname}%
600     \thispagestyle{plain}\parindent\z@
601     \parskip\z@ \@plus .3\p@\relax
602     \columnseprule \z@
603     \columnsep 35\p@
604     \let\item\@idixitem}
605     {\if@restonecol\onecolumn\else\clearpage\fi}
606 \newcommand\@idixitem{\par\hangindent 40\p@}
607 \newcommand\subitem{\@idixitem \hspace*{20\p@}}
608 \newcommand\subsubitem{\@idixitem \hspace*{30\p@}}
609 \newcommand\indexspace{\par \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}

```

### 2.13 Odds 'n' ends

The final section starts with the footnote ‘fence’ and the footnote alignment. There is also a list of the section names, which are the ones which get customised for other languages when you use the `babel` multilingual/multicultural package.

```

      article.cls
610 \renewcommand\footnoterule{%
611     \kern-3\p@
612     \hrule\@width.4\columnwidth
613     \kern2.6\p@}
614 \newcommand\@makefnrtxt[1]{%
615     \parindent 1em%
616     \noindent
617     \hb@xt@1.8em{\hss\@makefnmark}#1}
618 \newcommand\contentsname{Contents}
619 \newcommand\listfigurename{List of Figures}
620 \newcommand\listtablename{List of Tables}
621 \newcommand\refname{References}
622 \newcommand\indexname{Index}
623 \newcommand\figurename{Figure}
624 \newcommand\tablename{Table}
625 \newcommand\partname{Part}
626 \newcommand\appendixname{Appendix}
627 \newcommand\abstractname{Abstract}
628 \def\today{\ifcase\month\or
629     January\or February\or March\or April\or May\or June\or
630     July\or August\or September\or October\or
631     November \or December\fi \space\number\day, \number\year}
632 \setlength\columnsep{10\p@}
633 \setlength\columnseprule{0\p@}
634 \pagestyle{plain}
635 \pagenumbering{arabic}
636 \if@twoside
637 \else
638     \raggedbottom
639 \fi
640 \if@twocolumn
641     \twocolumn
642     \sloppy

```

```

643     \flushbottom
644 \else
645     \onecolumn
646 \fi
647 \endinput

```

To end with, there is the `\today` date, which non-Americans can recode as:

```

\def\today{\number\day\space\ifcase\month\or
    January\or February\or March\or April\or May\or June\or
    July\or August\or September\or October\or November\or
    December\fi\space\number\year}

```

The last few lines include the column spacing, page style, and page numbering setups. Single-sided work is allowed to have a slightly variable text height (the `\raggedbottom` command), and two-column setting has a strict height but slightly greater tolerance on justification.

## 3 Rolling your own

Having seen what the article class does and how it works, you have a choice: create your new class file from scratch, or build onto an existing class.

Writing a wholly new class requires a significant knowledge of L<sup>A</sup>T<sub>E</sub>X and T<sub>E</sub>X internals, but will have the advantage of being dedicated to the specific task on hand, and may offer more scope for automation, particularly if the process of generating the output is to be embedded within a larger application.

### 3.1 Re-using an existing class

Building on the work of other classes is more common, and has been described for a specific application (Minutes of meetings) in [3]. This involves loading the existing class file, handling any existing or new options, and then adding or modifying the commands and environments it provides.

We have already seen the use of `\renewcommand` (section 2.4.4 on p.116) and its counterpart for environments, `\renewenvironment` (section 2.7 on p.119), but you need to ensure the command and environments you are replacing are correctly preloaded. Hefferon [3] describes in detail the use of the `\LoadClass` and `\DeclareOption*` commands to specify the class on which you want to base yours, how to preserve existing options, and how to add your own.

### 3.2 Packages

As well as rewriting or modifying the code of an existing class, you can also invoke extra packages. In most cases this is faster, more reliable, and easier to do than rewriting the code of the existing class.

We have mentioned several useful packages:

**geometry** for the text area and page margins;  
**multicol** for multiple columns of text;  
**fancyhdr** for running headers and footers;  
**sectsty** for changes to section and title styles;  
**ccaption** for changes to the layout of Table and Figure captions;  
**tocloft** for changes to the layout of the Table of Contents and Lists of Figures and Tables;  
**babel** for working in multiple languages.

In your new class file, use the `\RequirePackage` command after the options (see section 2.3.4 on p. 113). If an option needs to refer to a specific package, put the `\RequirePackage` after the version and identification section but before your options (see section 2.2 on p. 111).

### 3.3 Four last things

The *Companion* [6, p. 888] specifies that ‘every class file *must* contain four things’:

1. a definition of `\normalsize`;
2. a value for `\textwidth`;
3. a value for `\textheight`;
4. a specification for `\pagenumbering`.

Beyond that, it’s up to you! If you have been documenting your class file in docT<sub>E</sub>X format as you go along, as explained in the first paragraph in section 2, you should now consider releasing it for general use by submitting it to the CTAN maintainers so that others can use it.

### Acknowledgments

This article originally appeared in the *PracT<sub>E</sub>X Journal* (2006:4) where it was set full out. The challenge in a two-column layout of fitting wide lines of verbatim code from files whose line-numbers are needed for reference was met by a suggestion from Karl Berry to use the Latin Modern Typewriter Light Condensed font (`lmttlc`) in the `\VerbatimInput` command of the `fancyvrb` package.

### References

- [1] Johannes Braams. Document classes and packages in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. *TUGboat*, 15(3), Sep 1994. <http://www.tug.org/TUGboat/Contents/contents15-3.html>.
- [2] Robin Fairbairns (Ed). Frequently Asked Questions about T<sub>E</sub>X. Technical report, UK T<sub>E</sub>X Users Group, Cambridge, UK, Nov 2005. <http://www.tex.ac.uk/cgi-bin/textfaq2html?label=ltxcmds>.
- [3] Jim Hefferon. Minutes in less than hours: Using L<sup>A</sup>T<sub>E</sub>X resources. *The PracT<sub>E</sub>X Journal*, 2005(4), Oct 2005. <http://www.tug.org/pracjourn/2005-4/hefferon/>.
- [4] Donald Knuth. *The T<sub>E</sub>Xbook*. Addison-Wesley, Reading, MA, Jun 1986.
- [5] Leslie Lamport. *L<sup>A</sup>T<sub>E</sub>X: A document preparation system*. Addison-Wesley, Reading, MA, 2nd edition, 1994.
- [6] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, Christine Detig, and Joachim Schrod. *The L<sup>A</sup>T<sub>E</sub>X Companion: Tools and Techniques for Computer Typesetting*. Addison-Wesley, Reading, MA, 2nd edition, May 2004.
- [7] Scott Pakin. How to package your L<sup>A</sup>T<sub>E</sub>X package. *CTAN*, Nov 2005. <http://www.ctan.org/tex-archive/info/dtvtut/dtvtut.pdf>.
- [8] The L<sup>A</sup>T<sub>E</sub>X3 Project. L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> for class and package writers, Dec 2003. `$TEXMFMAIN/texmf-dist/doc/latex/base/clsguide.{dvi|pdf}`.
- [9] Wayne Sewell. *Literate Programming in WEB*. Van Nostrand Reinhold, New York, NY, 1989.

# L<sup>A</sup>T<sub>E</sub>X resources

Jim Hefferon

St. Michael's College

Vermont, USA

ftpmaint (at) alan dot smcvt dot edu

You were lucky. We lived for three months in a paper bag in a septic tank.

Monty Python's *Four Yorkshiremen* sketch

When I started T<sub>E</sub>X-ing, things were right tough. I got my distribution in twenty four separate emails, which I had to stick together, to run a decode program, to convert to a file, to drop to the disk to make the executables. And, there was no shelf of books to consult, no pile of tutorials to peruse, no Internet group with fancy-pants search capabilities.

But it built character. It made you hard . . . or nuts — why the heck did I ever continue? Of course, I continued because of the output, which was wonderful.

No, the old days were the bad old days. Today there are many resources for someone who is trying to begin working with T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. So many, in fact, that a person can be unsure of which to use.

This is my guide to which books to check out, documentation files to print, and software packages to know about. It is meant for, say, a student or administrative assistant who has work to do and finds that they need to get it done with T<sub>E</sub>X.

I'm a Linux person so I can't give Mac or Windows advice, I unfortunately am comfortable in no language other than English, and I've read only the books that I've read. So I admit that my opinions are biased. But what can you do with opinions besides impose them on others?

## 1 Carrying on

I know that I am going to regret, in this age of search engines, writing an article that will appear online containing both terms “L<sup>A</sup>T<sub>E</sub>X” and “Carrying on” but I mean this in the way that it is used in the L<sup>A</sup>T<sub>E</sub>X manual: here's the information.

### 1.1 Books and journals

Someone starting out should use L<sup>A</sup>T<sub>E</sub>X. I keep these two books in reach.

- *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System (2nd ed.)*<sup>1</sup> by L<sup>A</sup>mpert is the manual by the software's author. It is well-written, if perhaps spare.

<sup>1</sup> ISBN-13: 978-0-201-52983-8

- *The L<sup>A</sup>T<sub>E</sub>X Companion*<sup>2</sup> by Mittelbach, et al., is a monumental effort that summarizes most of the important packages and techniques.

Also widely recommended for its how-to material is *A Guide to L<sup>A</sup>T<sub>E</sub>X*<sup>3</sup> by Kopka and Daly. I like to have access to Knuth's *T<sub>E</sub>Xbook*<sup>4</sup> and to *T<sub>E</sub>X By Topic*<sup>5</sup> by Eijkhout. However, I don't use these often; for instance, Knuth's book is across campus in the library.

I read two journals. The journal of the T<sub>E</sub>X Users Group is *TUGboat*.<sup>6</sup> Getting this is one of the major benefits of joining TUG (many languages and areas have their own [user group](#);<sup>7</sup> consider joining for the publications and for the meetings). The online journal *The PracT<sub>E</sub>X Journal*<sup>8</sup> is a recent entry but perfect for someone feeling their way around the landscape.

### 1.2 Shorter writings

Suck these down off the Internet, print them out, and three-hole punch them.

- The most-often recommended tutorial is *The Not-So Short Guide to L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub>*.<sup>9</sup>
- The American Math Society's material is documented in the *AMS Math Guide*.<sup>10</sup>
- I look for symbols that I don't even know the name of in the *Comprehensive List of Symbols*.<sup>11</sup>
- To understand how to incorporate and place graphics I refer to *Using Imported Graphics in L<sup>A</sup>T<sub>E</sub>X and pdfL<sup>A</sup>T<sub>E</sub>X*.<sup>12</sup>
- Sometimes the best way to learn the right thing to do is to be smacked for doing something that you shouldn't. If you know that you have bad

<sup>2</sup> ISBN-13: 978-0-201-36299-2

<sup>3</sup> ISBN-13: 978-0-321-17385-0

<sup>4</sup> ISBN-13: 978-0-201-13448-3

<sup>5</sup> <http://www.eijkhout.net/tbt/>

<sup>6</sup> <http://www.tug.org/TUGboat/>

<sup>7</sup> <http://www.tug.org/usergroups.html>

<sup>8</sup> <http://www.tug.org/pracjourn/>

<sup>9</sup> <http://www.ctan.org/tex-archive/info/lshort/>

<sup>10</sup> <http://www.ctan.org/tex-archive/macros/latex/required/amslatex/>

<sup>11</sup> <http://www.ctan.org/tex-archive/info/symbols/comprehensive/>

<sup>12</sup> <http://www.ctan.org/tex-archive/info/epslatex/>

habits, or if you need to find out that you have them, then *l2tabu*<sup>13</sup> will tell you what is taboo.

### 1.3 Web pages

Many web pages offer help with T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. One that I cannot live without is Robin Fairbairn's *English FAQ*.<sup>14</sup> Another favorite is the *TUG web resources page*.<sup>15</sup>

Even with those two, I sometimes just google for an answer. The advice that I get is typically useful, although it can be outdated.

### 1.4 Discussion

Internet talk about T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X has been going on for ... as long as there has been an Internet. From time to time I scan the Usenet group *comp.text.tex*<sup>16</sup> for ideas and help; you can also search this group.

### 1.5 CTAN

The *Comprehensive T<sub>E</sub>X Archive Network*<sup>17</sup> is our community's archive. You can *search*<sup>18</sup> or *browse the tree*<sup>19</sup> including the *L<sup>A</sup>T<sub>E</sub>X subtree*.<sup>20</sup> We (I run one of the core nodes) hold about 5000 packages of T<sub>E</sub>X-related materials.

### 1.6 Supporting tools

There are many tools that help you work with L<sup>A</sup>T<sub>E</sub>X. To type the input I use Emacs with the *AUC-T<sub>E</sub>X*<sup>21</sup> macro package. I output everything to PDF so I view it with *Adobe Reader*<sup>22</sup> or *xpdf*<sup>23</sup> (which lets me easily refresh the document and comes up faster than the Reader on a slow connection, but sometimes shows my document a bit differently).

I don't use bibliography tools but the standard is *BibT<sub>E</sub>X*.<sup>24</sup>

### 1.7 Add-on L<sup>A</sup>T<sub>E</sub>X packages

There are many packages to enhance L<sup>A</sup>T<sub>E</sub>X, but these are the ones that I find essential. They are all available from CTAN; however, most likely they are all already included in your T<sub>E</sub>X installation, because they are all popular.

<sup>13</sup> <http://www.ctan.org/tex-archive/info/l2tabu>

<sup>14</sup> <http://www.tex.ac.uk/faq>

<sup>15</sup> <http://tug.org/interest.html>

<sup>16</sup> <http://groups.google.com/group/comp.text.tex>

<sup>17</sup> <http://www.ctan.org/>

<sup>18</sup> <http://www.ctan.org/search.html>

<sup>19</sup> <http://www.ctan.org/tex-archive/>

<sup>20</sup> <http://www.ctan.org/tex-archive/macros/latex/>

<sup>21</sup> <http://www.gnu.org/software/auctex/>

<sup>22</sup> <http://www.adobe.com/products/acrobat/readermain.html>

<sup>23</sup> <http://www.foolabs.com/xpdf/>

<sup>24</sup> <http://en.wikipedia.org/wiki/BibTeX>

- Make the most of the mathematical capabilities of (L<sup>A</sup>)T<sub>E</sub>X with the *AMS L<sup>A</sup>T<sub>E</sub>X*<sup>25</sup> package.
- Page layout is tricky. Adjust the size and orientation of your page with *geometry*.<sup>26</sup> Get control over headers and footers with *fancyhdr*.<sup>27</sup>
- Import graphics into a L<sup>A</sup>T<sub>E</sub>X document with the *graphics*<sup>28</sup> package. This package includes the *color* material. If there is more that you want to do in color than this package seems to provide then use *xcolor*.<sup>29</sup>
- Make an index with *makeidx*.<sup>30</sup>
- The *verbatim*<sup>31</sup> package has a number of useful environments, including a `comment` environment to omit parts of the document. For computer code, I use *listings*.<sup>32</sup>
- I like footnotes numbered per-page so I use *footmisc*.<sup>33</sup>
- The *hyperref*<sup>34</sup> package gives you hyperdocument features, such as making table of contents entries link to the corresponding document part.
- Typeset web addresses with *url*,<sup>35</sup> which is also great for computer file names and works either with or without *hyperref*.
- *Beamer*<sup>36</sup> gives me fine presentation slides.

## 2 Signing off

There are many more resources around than there used to be, thank goodness. The ones here are what I would mention to someone who is only trying to use T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X to get their work out the door.

To repeat, these are my opinions only. If you don't like them — if for instance you think footnotes numbered per-page should be a capital offense — then you can go live in a paper bag!

<sup>25</sup> <http://www.ams.org/tex/amslatex.html>

<sup>26</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/geometry/>

<sup>27</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/fancyhdr/>

<sup>28</sup> <http://www.ctan.org/tex-archive/macros/latex/required/graphics/>

<sup>29</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/xcolor/>

<sup>30</sup> <http://www.ctan.org/tex-archive/macros/latex/base/>

<sup>31</sup> <http://www.ctan.org/tex-archive/macros/latex/required/tools/>

<sup>32</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/listings/>

<sup>33</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/footmisc/>

<sup>34</sup> <http://www.tug.org/applications/hyperref/>

<sup>35</sup> <http://www.ctan.org/tex-archive/macros/latex/contrib/misc/url.sty>

<sup>36</sup> <http://latex-beamer.sourceforge.net/>

# L<sup>A</sup>T<sub>E</sub>X for academics and researchers who (think they) don't need it

Peter Flom

National Development and Research Institutes, Inc.

71 West 23rd St., 8th floor

New York, NY 10010

flom (at) ndri dot org

<http://www.peterflom.com>

## Abstract

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.

## 1 Introduction

Why should you learn L<sup>A</sup>T<sub>E</sub>X? To the uninitiated, L<sup>A</sup>T<sub>E</sub>X code looks bizarre. Like this:

```
\item Order each distribution. Let
 $x_{(1)} \dots x_{(n)}$  be the ordered x
values, and  $y_{(1)} \dots y_{(m)}$  be the
ordered y values; and let  $m \leq n$ .
\item If  $m=n$  then  $x_i$  and  $y_i$  are both
 $(i-0.5)/m$  quantiles of their distributions;
in this case, simply graph  $x_i$  against
 $y_i$  (if  $m$  is very large, say, more than
500, then fewer quantiles are needed).
```

Why would anyone bother with all that arcane stuff to prepare documents? Why not just use Word or some other word processor?

Well, there are lots of good reasons. For people in some disciplines, L<sup>A</sup>T<sub>E</sub>X is virtually indispensable. You've managed to get along without it. But if you give it a try, you may wonder *how* you got along without it. It takes some getting used to, and you'll probably have some problems at first, but it's not really that hard; simple things can be done very quickly, and there are lots of resources to help. Once you are used to it, L<sup>A</sup>T<sub>E</sub>X is *easier* to use than word processors; in particular, it makes certain parts of writing scholarly articles much easier. Most of these articles include bibliographies and L<sup>A</sup>T<sub>E</sub>X has tools to manage bibliographies. Different publishers require different formats, L<sup>A</sup>T<sub>E</sub>X has tools to manage these, and some publishers may even provide their own L<sup>A</sup>T<sub>E</sub>X formatting templates, so that everything is set up automatically. Papers may include figures and tables, and, when they do, they will require cross-referencing, and, yes, you guessed it, L<sup>A</sup>T<sub>E</sub>X has tools to handle this as well. And many papers require revisions, both before and after they are submitted. L<sup>A</sup>T<sub>E</sub>X will renumber everything automatically — references, cross-references, section numbers — everything.

When you want to make presentations, L<sup>A</sup>T<sub>E</sub>X has add-on packages which can help create presentations that look good, and can include many options for overlays and navigating through a presentation. Finally, if you decide to write a book, L<sup>A</sup>T<sub>E</sub>X can handle books of any length, and can produce books of real beauty (for one example, take a look at [3]).

This article is organized as follows: in section 2, I cover some of the basics of using L<sup>A</sup>T<sub>E</sub>X. In section 3, I give more detail on the helpful things L<sup>A</sup>T<sub>E</sub>X can do that I listed above. In section 4, I show why some of the bad things you may have heard about L<sup>A</sup>T<sub>E</sub>X aren't true. Finally, in section 5, I give some resources for those who want to learn more.

## 2 The very basics

L<sup>A</sup>T<sub>E</sub>X is not a word processor. It's a document preparation system. Rather than type words and then format them using drop-down menus, in L<sup>A</sup>T<sub>E</sub>X the formatting is part of the text, all of which is written in ASCII characters. At first, this seems bizarre, but after a while (not too long a while) you begin to appreciate it.

L<sup>A</sup>T<sub>E</sub>X does *exactly* what you tell it. You can *see* what you are telling it; if something goes wrong (and it will) you can try to find the problem yourself, and, if you can't, you can show it to others. You can even e-mail it. Try doing that with something you did in a word processor; "Well, I was using version 9 and I clicked on this, and then on that, then the pull-down menu appeared and I clicked on the default ... then I entered 2". Sheesh. With L<sup>A</sup>T<sub>E</sub>X, you can e-mail your actual code to an expert, or to one of the help groups listed below. The L<sup>A</sup>T<sub>E</sub>X community is friendly, there are places to go to get help, probably right on your campus. In my experience, L<sup>A</sup>T<sub>E</sub>X experts welcome L<sup>A</sup>T<sub>E</sub>X novices. (Some suggestions of where to get help are in section 5.)

### 3 The good stuff

#### 3.1 Sectioning

So, you're writing a long article. It has sections. How to create them? An example:

```
\section*{Introduction}
In this article I prove that the key dependent
variable in my field is related to the
particular independent variables I have
available to me, and in just the ways I
thought it would be.
\section{Methods}
  \subsection{Subjects}
    A bunch of students who happened to be
    in my class the day I had a bad cold
    and couldn't give a lecture.
  \subsection{Analysis}
    I think I tried ANOVA.
\section{Results}
All my null hypotheses were rejected.
Now they feel bad.
\section{Discussion}
If this doesn't get me a big grant, I'm
history in this department. Maybe I can
find work as a \LaTeX{} compositor?
```

(Note that you can indent your source however you like). L<sup>A</sup>T<sub>E</sub>X will handle the numbering, the formatting, the spacing, and all that, leaving you free to do the writing and the thinking. And L<sup>A</sup>T<sub>E</sub>X won't try to guess what you're thinking, or start numbering sections whenever you type a number, or start indenting like crazy.

#### 3.2 Cross-referencing

At some points in your article you want to refer to other parts, or to figures, or tables. No problem. At the part you want to refer to you need a `\label` command, and at the point where you make your reference, you need a corresponding `\ref` command. Like this:

```
In subsection \ref{SS:section} I showed you how
to create sections and subsections.
```

This produces the following:

```
In subsection 3.1 I showed you how to create
sections and subsections.
```

#### 3.3 A simple table

OK, I admit it. It can be hard to create complicated tables in L<sup>A</sup>T<sub>E</sub>X.<sup>1</sup> It's hard to create good complicated tables in *any* program. But simple tables aren't so bad. Here's an example:

---

<sup>1</sup> Some L<sup>A</sup>T<sub>E</sub>X systems automatically put the basics of an environment in place for you, such as XEmacs, LyX, T<sub>E</sub>Xmacs, and others.

```
\begin{table}
\centering
\begin{tabular}{lrr}
Quantile & Male & Female\\\hline
0\%      & 59  & 44\\
50\%     & 69  & 64\\
100\%    & 77  & 71\\
\end{tabular}
\caption{Quantiles of male and female heights}
\label{tab:malefemale}
\end{table}
```

which produces Table 1:

Quantile	Male	Female
0%	59	44
50%	69	64
100%	77	71

**Table 1:** Quantiles of male and female heights

What does all this do? Well, there isn't space here to go into everything even for this table (see Section 5). But for a start:

- The `\begin{table}` and `\end{table}` define a *table* which can be placed anywhere in a document, and given a caption and a label. Every `\begin` must have an `\end`.
- The `\centering` command horizontally centers everything following until the `\end{table}`.
- The `\begin{tabular}` sets up a table, and the `{lrr}` makes it three columns with the first left aligned and the others right aligned.
- The ampersands (the `&` character) separate columns.
- A double backslash (`\\`) ends each row.
- The `\hline` adds the printed horizontal line.

#### 3.4 Graphics

L<sup>A</sup>T<sub>E</sub>X also provides extensive methods to deal with graphics. For social scientists, perhaps the most useful are ways to directly import `.pdf` and `.eps` files that are created by other programs, such as statistical packages. Although the full use of imported graphics can be complex (see Section 5) a basic example takes the following steps:

1. Create your graphics file (e.g., `diagram1.pdf`) and store it in the same directory with your L<sup>A</sup>T<sub>E</sub>X file for the article you are writing.
2. In the preamble include the following command: `\usepackage{graphicx}`.
3. At the spot where you want your diagram, put `\includegraphics{diagram1}`.

If you want L<sup>A</sup>T<sub>E</sub>X to move the figure to the closest position where it will fit in your document, give it a label and cross-reference inside a `figure`, similar to the previous `table` example. For instance:

```
\begin{figure}
  \centering
  \includegraphics{diagram1}
  \caption{This is an example of a figure.}
  \label{fig:example}
\end{figure}
```

### 3.5 BibT<sub>E</sub>X

Although space here does not permit a full discussion of bibliography creation in L<sup>A</sup>T<sub>E</sub>X, you should know that there is a package called BIBT<sub>E</sub>X which allows you to create extensive bibliographies, enter the information for each citation in a natural way, and then never have to reenter the information again. There are methods for formatting the citations to match a wide variety of styles.

## 4 The (supposedly) bad stuff

### 4.1 L<sup>A</sup>T<sub>E</sub>X is hard to learn

OK, this is *partially* true, in that, if you want to or need to, there is a *lot* you can do with L<sup>A</sup>T<sub>E</sub>X. You can create complicated diagrams, write long books with complex and beautiful formatting, create multiple indexes, and multiple lists, and on and on. But the *basics* of L<sup>A</sup>T<sub>E</sub>X are *not* so hard; in fact, you're well on your way with what you've seen here.

### 4.2 L<sup>A</sup>T<sub>E</sub>X can't be annotated

This one is simply incorrect. There are several ways to insert editorial comments into L<sup>A</sup>T<sub>E</sub>X files. One is to use the `\textcolor` command to insert comments in a different color. Another is to use `\marginpar` to insert comments in the margin.

### 4.3 You can't share L<sup>A</sup>T<sub>E</sub>X files with people who use Word

There are some free programs which attempt to convert L<sup>A</sup>T<sub>E</sub>X to Word, for example, `latex2rtf` (<http://tug.org/utilities/texconv/latex2rtf.html>). I haven't tried these extensively. For Windows, I have found the commercial program TeX2Word to be quite useful; see <http://www.chikrii.com/> and Dave Walden's articles [4] for more information on this software.

### 4.4 You can't see the output while you type

While technically true, typesetting a file and previewing the result is typically a matter of a sin-

gle keystroke or mouse click, and typesetting is extremely fast.

## 5 Where to go from here

There is a huge variety of materials to help you learn more about L<sup>A</sup>T<sub>E</sub>X:

- CTAN (The Comprehensive T<sub>E</sub>X Archive Network) is a repository of T<sub>E</sub>X macros, packages, formats, utilities, and other goodies, and has lots of material, some of it for beginners. Two that I found useful are
  - For more on graphics, see <http://www.ctan.org/tex-archive/info/epslatex>.
  - For a thorough introduction to L<sup>A</sup>T<sub>E</sub>X, see <http://www.ctan.org/tex-archive/info/beginlatex>.

There are numerous other introductory materials there, as well—the above are just my own preferences.

- Books, including:
  1. *Guide to L<sup>A</sup>T<sub>E</sub>X* [2] which is an excellent introduction to L<sup>A</sup>T<sub>E</sub>X.
  2. *Math into L<sup>A</sup>T<sub>E</sub>X* [1], which is particularly useful if you have to type a lot of math.
  3. *The L<sup>A</sup>T<sub>E</sub>X Companion* [3]. This is a great reference, but not for beginners; it's also good for impressing people with the power of L<sup>A</sup>T<sub>E</sub>X. If you start using L<sup>A</sup>T<sub>E</sub>X a lot you will probably wind up wanting this one.
- The mailing list for general questions and discussion is <http://lists.tug.org/texhax>.
- The T<sub>E</sub>X newsgroup, `comp.text.tex`, is also for general questions and discussion.
- There are a number of FAQs and lists of tips and tricks; two I've used are
  1. <http://www.tex.ac.uk/faq>
  2. <http://www.texnik.de/>

## References

- [1] George Grätzer. *Math into L<sup>A</sup>T<sub>E</sub>X*. Birkhäuser, New York, 3<sup>rd</sup> edition, 2000.
- [2] Helmut Kopka and Patrick W. Daly. *Guide to L<sup>A</sup>T<sub>E</sub>X*. Addison Wesley, Boston, 4th edition, 2004.
- [3] Frank Mittelbach, Michel Goossens, et al. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison Wesley, Boston, 2nd edition, 2004.
- [4] David Walden. Travels in T<sub>E</sub>X Land. *The PracT<sub>E</sub>X Journal*, 2005 (issues 3 and 4). <http://tug.org/pracjourn/2005-3/walden-travels> and <http://tug.org/pracjourn/2005-4/walden-travels>.

# Hypertext capabilities with pdf $\LaTeX$

Federico Garcia

federook (at) gmail dot com

## 1 Introduction

With the standardization of electronic publishing and sharing of manuscripts, a wealth of new technical resources and possibilities is open to authors, resources that go beyond habitual on-paper typographic uses, to involve the new everyday tool of the reader: the mouse click.

The  $\TeX$  world, as usual, has been quick to take up the new possibilities. In particular, pdf $\TeX$  by Hàn Thế Thành et al., along with its companion pdf $\LaTeX$ , and the `hyperref`  $\LaTeX$  package by Sebastian Rahtz and Heiko Oberdiek — both available in standard distributions of  $\TeX$  and  $\LaTeX$  — are especially successful implementations of the possibilities of the PDF format — by far the most standardized, and by now ubiquitous, electronic format.

This article is intended as a quick guide to the most immediately usable functions of these tools. It is not intended to replace the available documentation, which is much more precise and complete. In fact, my intention is to steer away from completeness. For example, the file `options.pdf` in the `hyperref` distribution lists all the options to the package. This list, containing no explanations or running text, is enough to fill more than two pages — there's well over 60 options. The main fact about them, however, is that most of them are completely meaningless to the average user, who probably won't understand them anyway (at least in my case).

In other words, only a small portion of the universe of possibilities of on-screen documents is directly relevant for the average author. But the documentation of these features (the relevant ones) is all too often obscured by the rest of them. The whole business appears more overwhelming than it actually is. Hopefully, this article will help this perceived situation. It is an exposition of the (relevant) extended possibilities of pdf $\LaTeX$  in terms of the  $\LaTeX$  we all know. In some cases this will actually imply a lie, a white one. A footnote will state as much in such cases, without going into further details.

## 2 Immediate special effects of `hyperref`

When `hyperref` is loaded (for useful options, and tips on loading the package, see [Hypersetup and options](#))

in an otherwise normal  $\LaTeX$  document, a number of things happen without further intervention:

- The items in the table of contents, the list of figures, and the list of tables, will be links: when the reader clicks on them, the cursor will jump to the corresponding target.
- The superscript that calls for a footnote will be a link to the footnote itself.
- Bibliographical references through `\cite` will create links to the entries in the final bibliography list.
- All pairs of `\label`-`\ref` will also produce links (the result of a `\ref` leading by mouse click to the corresponding `\label`).

The last point merits a few notes. First of all, this is true of *all* elements capable of cross referencing in  $\LaTeX$ : not only chapters and sections, but also to `enumerate` items, equations, and footnotes.

For example, after

```
\footnote{This is the footnote.}\label{note}
\begin{enumerate}
\item \label{item1} This is the first item.
\item \label{item2} This, the second.
\end{enumerate}
```

a `\ref` to any of these keys will produce the corresponding number as a link.

The same is true for `\pageref`, which prints the number of the *page* where the referenced element appears (rather than the number of the element itself). With `hyperref`, this page number will be a link.

For chapters and sections, in addition, `hyperref` offers a third command in the family: `\nameref`. Instead of printing the chapter or section number, `\nameref` will typeset *the name* of the chapter/section. This is much better and elegant than using the number if the document is intended to be read on the screen.

## 3 Arbitrary cross references

`\label` and `\ref` function in connection with  $\LaTeX$  counters. But on-screen reading is not limited to refer to things that have a number. Thus, `hyperref` offers commands for the creation of cross references that are independent from counters. These are `\hypertarget` and `\hyperlink`, exact analogues of `\label` and `\ref` respectively.

Both of these commands, however, have a second argument:

```
\hypertarget{<key>}{<text>} (like \label)
\hyperlink{<key>}{<text>} (like \ref)
```

For `\hypertarget`, `<text>` is the destination of the user's click; for `\hyperlink`, it is the text of the link itself. The `<text>`, in both cases, is any L<sup>A</sup>T<sub>E</sub>X box.

As an example, the following code creates a picture (from the file `logo.png`) that is the destination of a link:

```
\hypertarget{ref1}{\includegraphics{logo.png}}
```

The link itself, with the text ‘see the logo’, would be created with

```
\hyperlink{ref1}{see the logo}
```

## 4 External links

### 4.1 Cross referencing between files

The links made by `\label`-`\ref` and `\hypertarget`-`\hyperlink` work within a single file. Cross referencing *between* files is possible thanks to a third pair of commands provided by `hyperref`, described next.

This time, however, the commands are less than analogous to the usual `\label`-`\ref`. They require not one, but two ‘keys’, called `<key>` and `<category>` in the syntax below). Why this is so has to do with PDF syntax, but the L<sup>A</sup>T<sub>E</sub>X user can think of `<category>` as a second key.

```
\hyperdef{<category>}{<key>}
\hyperref{<file>}{<category>}{<key>}{<text>}
```

For example, the following line sets up a destination in the present file (note the space at the end of the line):

```
\hyperdef{xmpl}{dest}_%
```

An external file would link to this destination (in the present file) with

```
\hyperref{hypertext.pdf}{xmpl}{dest}
{to the destination}
```

### 4.2 Links to the web

The other kind of common external link is to a webpage. The command `\url` takes one argument — the destination's URL address — and creates a link to it (a click on it opens the system's browser on the requested page). For example, `\url{www.tug.org}` leads to the TUG web site.

A related kind of link is the ‘mailto’ link: with a click on it, the system opens the local email program and creates a new message to the indicated email address. This is done through the command:

```
\href{mailto:<email address>}{<link text>}
```

## 5 Bookmarks

The `hyperref` package handles virtually everything related to bookmarks (sometimes called “outlines”) that the average user comes across. If instructed (see [Hypersetup and options](#)), the package will automatically compile the bookmark panel from the items in the table of contents. This includes sections, chapters, etc., and also the usual L<sup>A</sup>T<sub>E</sub>X results of `\addtocontents` and related commands.

There are two things that are not so direct: first, how to get a bookmark that does *not* correspond to an item in the TOC. For this, `hyperref` offers `\pdfbookmark[<level>]{<text>}{<key>}`.

The `<level>` is 0 for chapters, 1 for sections, etc., and `-1` for `\parts`. The `<text>` is the text of the bookmark itself. The `<key>` doesn't really matter to us (it's another of those PDF-format-related requirements), except for the requirement that it be unique.

The destination of a bookmark created by such a `\pdfbookmark` command is the exact place where the command is issued. The bookmark itself will be appended, in the bookmark panel, in the position where the command is issued *with respect to other chapters, sections, etc.*

For example, the present article has a “Dummy Bookmark”, which is created right here with

```
\pdfbookmark[2]{“Dummy Bookmark”}{“bmkey”}
```

As a result, it appears (in the bookmark panel) right below the one for this section (“Bookmarks”) and before the one for the next (“Text and T<sub>E</sub>X”). The fact that it appears as a subitem of “Bookmarks” is due to the [2] argument (the `<level>`).

So, how to get bookmarks appended to *other* places in the panel? For example, how to create a bookmark whose destination comes after a section heading, but with the bookmark itself being placed before the one for that section in the bookmark panel?

This question is not simply a puzzle, but has a potentially useful application: the mapping of the list of figures or the list of tables (as well as the table of contents) in the bookmark panel. This is the second thing that is not so direct with `hyperref` (or `pdfLATEX`, for that matter).

In fact, it is a relatively hard thing to achieve. I found a solution when writing `pittetd`, the electronic dissertations class at the University of Pittsburgh. I have a project of abstracting this part of the `pittetd` code and uploading it to CTAN as a small, independent package, but for the moment interested readers can consult the documentation of the solution in `pittetd.dtx`.

## 6 Text and T<sub>E</sub>Xt

Bookmarks are the main environment of another issue PDF writers should be aware of: they are made only of plain text, and cannot support what L<sup>A</sup>T<sub>E</sub>X can put in—for example—section titles. For example, the next subsection:

### 6.1 $\alpha$ -expressions and H<sub>2</sub>O

This heading looks right in the text, but the corresponding bookmark is wrong.

The alternative takes care of the bookmark:

### 6.2 Alpha-expressions and water

But the heading in the text is not satisfactory.

### 6.3 $\alpha$ -expressions and H<sub>2</sub>O

The solution is to use another command from `hyperref`, this one named `\texorpdfstring`:

```
\texorpdfstring{\TeX text}{\plain text}
```

The result is a flexible expression that behaves like T<sub>E</sub>X text (T<sub>E</sub>Xt) in a L<sup>A</sup>T<sub>E</sub>X context, and like plain text in PDF-related strings.

For example, the present subsection was created with:

```
\subsection{%
  \texorpdfstring{\$alpha}{Alpha}-expressions
  and \texorpdfstring{H$_2$O}{water}}
```

This procedure is generally not needed with accents and commands with immediate expansion, including common logos and symbols. But `hyperref` cannot convert more advanced stuff—notably math mode—without help, and gives a warning. It is in these cases that `\texorpdfstring` is useful.

## 7 Post-it notes

On page 130 above there is an expandable ‘post-it’ note. In my opinion, this easily overlooked resource of PDF offers an excellent alternative to the footnote when a document is intended to be read on the screen. `hyperref` does not include support for these notes, but their creation (through pdfT<sub>E</sub>X primitives) is not hard:

```
\pdfannot width w height h depth d
  { /Subtype /Text /Contents (text) }
```

The three dimensions `w`, `h`, and `d` are all L<sup>A</sup>T<sub>E</sub>X dimensions. But the one that is important is `h` (the height), because it determines where the note appears in relation to the text baseline. It is a good idea use a `\quad` after the command.

One thing to keep in mind with post-it notes is that their exact behavior (color, size, when it opens, how it closes, etc.) is not very standardized, and tends to change from version to version of Ac-

robat Reader. Nevertheless, the note can be given a different color if, between the braces of `\pdfannot`, `/C [r g b]` is appended; a title for the note is determined with `/T (title)`; and the note can be open by default with `/Open true`.

If post-it notes are used at all, the document has to be typeset by pdfT<sub>E</sub>X (rather than converted from DVI), since otherwise the primitive `\pdfannot` is not available. See the next section.

## 8 Hypersetup and options

The `hyperref` package has so many options that it provides a separate command `\hypersetup` for configuration. Thus, options to the package can be specified either in the usual way

```
\usepackage[options]{hyperref}
```

or in a separate line:

```
\hypersetup{option, option, ...}
```

Some of the many options of `hyperref` are of particular interest.

---

```
colorlinks
linkcolor=red
urlcolor=blue
citecolor=green
```

---

```
bookmarks
bookmarksopen
bookmarksnumbered=false
```

---

```
pdftitle="Hypertext capabilities with pdfTeX"
pdfauthor="Federico Garcia"
pdfkeywords="Bookmarks, links, PDF, LaTeX"
```

---

The last three are useful to set up the fields of the ‘Document Properties’ information. (Again, the behavior of this is not reliably standard from version to version of Acrobat Reader.)

Another thing to keep in mind is that `hyperref` has to know the way the PDF file is produced: whether pdfT<sub>E</sub>X is run on the document, or a separate program will convert the DVI—in which case `hyperref` needs to load the corresponding driver. This information is given to the package as an option (`‘pdftex’` for pdfT<sub>E</sub>X, `‘dvi2pdfm’` if this program is used, etc.). Since this information is also used by other packages—notably `graphicx`—it is customary to indicate this option as a general option *to the document class*, which will then pass it to any package that needs it.

## 9 A tip

`hyperref` changes the internal mechanism of cross references. The change is truly magical, in the sense that the user, most of the time, has and needs to have no idea that a change occurred.

However, there is one case in which the change

becomes relevant: when a document is ‘converted’ from plain to hyperlinked, or the reverse; in other words, when the `\usepackage{hyperref}` line is added or removed. The presence of auxiliary files created by one mechanism and used by the other *will* create quite unintelligible error messages.

Therefore, a tip: make sure to delete all auxiliary files (`.aux`, `.toc`, `.lot`, `.lof`, ...) when you will run a document for the first time with `hyperref` (or, later, for the first time without it).

## 10 Material not covered

Two main things are not covered in this document: ‘hyper-bibliography’, in which the entries in the bibliography list can link back to the citations in the text; and ‘hyper-index’, in which the page numbers in the index are links to the corresponding pages.

These topics are not so easy to deal with in a general manner. In the main, support for these tends to be fragile, because of the myriad of styles for both bibliography and index. As a result, dealing with them essentially requires proficiency with `LATEX` and *MakeIndex*. Given this proficiency, the file `hyperref.dtx` is a source of information on how to get things done.

In the case of bibliographies, a more immediate guide is the file `backref.pdf`, actually the documentation to the sub-package of the same name (by David Carlisle and Sebastian Rahtz). The file is part of the `doc` directory of `hyperref`.

## 11 Documentation of hyperref

This is a description of the PDF files in the `doc` directory of `hyperref`:

**manual** by Sebastian Rahtz is a reference guide to the package. Not very useful for beginners, it contains often crucial information on details, important when you are doing something extraordinary.

**paper** by Heiko Oberdiek is a precise and complete explanation of many of the topics treated here, from the point of view of ‘how does it work?’ (rather than ‘how do I use it?’).

**slides** is the set of slides used by Heiko in his presentation of ‘paper’. It is an illustration of the possibilities of using `hyperref`, `pdfTEX`, and the package `thumbpdf`.

# Removing vertical stretch — mimicking traditional typesetting with $\text{\TeX}$

Kaveh Bazargan and CV Radhakrishnan  
River Valley Technologies  
[www.river-valley.com](http://www.river-valley.com)

## Abstract

One of  $\text{\TeX}$ 's advantages over traditional typesetting systems is the mechanism to stretch horizontal and vertical glue as needed, in order to aid paragraph building and pagination. But all  $\text{\TeX}$  operators involved in day-to-day page make-up know that this inbuilt intelligence is often 'too clever' and frustrating for the user.  $\text{\TeX}$  will not do what you want it to do, and only over many years can operators gain the knowledge that allows them to make just the right change to the source code in order to coerce  $\text{\TeX}$  to produce the desired result.

Recently we have been experimenting with removing vertical stretchability, with promising results. Our approach is to round off the height of all vertical material, including floats and displayed equations, to be an integral number of the leading of the main text. One advantage is that this allows true 'grid' setting in double column text.

## 1 Some things we have always wanted

It is useful to look at some things we have always wanted to do in  $\text{\TeX}$  but found difficult.

### 1.1 More control over glue

Anyone involved with 'real world' pagination of  $\text{\TeX}$  and  $\text{\LaTeX}$  files is aware of the frustration that  $\text{\TeX}$ 's 'glue' can generate.  $\text{\TeX}$ ies have long learned to accept this limitation, and developed tricks to work efficiently. However, some apparently simple tasks are still mysteriously complex to the non- $\text{\TeX}$ ie. For example, in figure 1, suppose that two lines have to move from the first to the second column. Logic would imply that two lines from the second column would have to move to the next page. But in  $\text{\TeX}$  this rarely happens; for instance, the glue around the displayed equation might shrink or stretch instead.

### 1.2 Grid setting

One of the most frequent complaints about  $\text{\TeX}$  when setting double column text is that, normally, the lines of text are not set on a grid. In other words, the baselines of one column do not align with those of the other. This is another consequence of the inherent stretchability of  $\text{\TeX}$ 's glue mechanism.

### 1.3 Precise control over positioning of graphics

This is another related problem. Suppose we want to move a floating graphic slightly higher or lower, without affecting pagination. For example we might

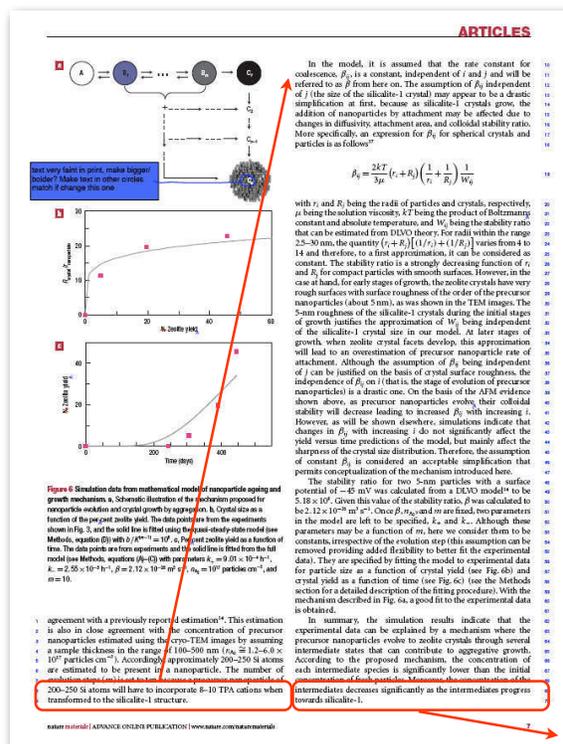
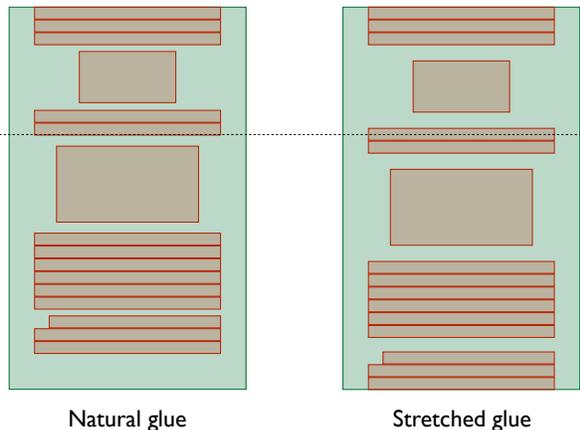


Figure 1: In  $\text{\TeX}$  documents, taking two lines over from the first column does not necessarily result in two lines from the second column moving over.



**Figure 2:** Stretching of glue to fit text to a page. The positions of the baselines are hard to predict when glue has stretchability.

want the top of a graphic appearing at the beginning of a column to be moved up a fraction in order to align with the top of the text in the next column. With the usual  $\LaTeX$  commands this is not easy. It turns out that our macros for controlling glue allow us to control the exact positioning of graphics too.

## 2 How $\TeX$ makes up pages

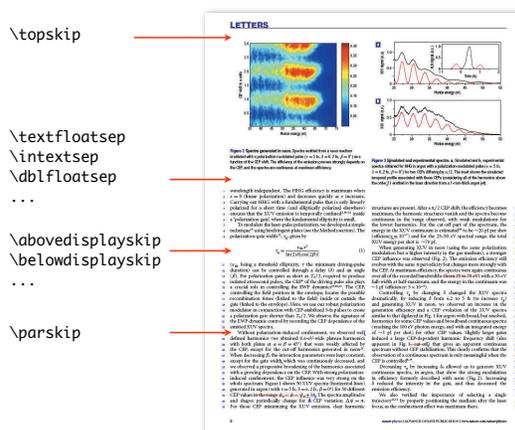
Figure 2 shows  $\TeX$ 's normal mechanism for setting a page. First of all the boxes are arranged in the vertical list, spaced out by the natural height of the glues assigned. Then,  $\TeX$  stretches or shrinks the glues so as to fit the boxes and make the last baseline align with the bottom of the page. As is evident, it is difficult to control the positions of the baselines using this method.

## 3 Our approach to the solution

At River Valley, we have thought about and discussed extensively the methods we might use to effect grid setting. These include testing each box in the vertical list on the fly, and tweaking the vertical position. We tried to do this, both using  $\TeX$  macros, and also doing it at compiler level, using  $\text{pdf}\TeX$ . Unfortunately this approach did not work.

The more successful strategy was to try and make sure that all items in the vertical list had heights which were integral multiples of the value of  $\backslash\text{baselineskip}$ . Examples of these items are:

- Floating elements (e.g. figures and tables)
- Displayed equations
- Section headers
- All skips between paragraphs, etc.



**Figure 3:** Some of the many vertical glues that can affect pagination. In general, baselines of the two columns are not aligned on a grid.

Figure 3 shows some of the many glue parameters that are inserted in the vertical list and which must conform to this rule.

Let's look in more detail at how we dealt with specific issues.

### 3.1 Glue at the top of each column

When  $\TeX$  starts typesetting a page, it inserts a glue called  $\backslash\text{topskip}$  at the top of the column. The value of this glue is derived from a complex formula involving the elements in the first line. We set  $\backslash\text{topskip}$  to the value of  $\backslash\text{baselineskip}$  which simplified matters considerably and did not produce any problems.

### 3.2 Stretching baseline glue

$\backslash\text{lineskip}$  and  $\backslash\text{lineskiplimit}$  are two more parameters that can cause big headaches in grid setting. Here is the logic:  $\TeX$  sets paragraphs by putting one line above another, normally spacing out lines by the value of  $\backslash\text{baselineskip}$ . The exception comes when  $\TeX$  thinks two adjacent lines might clash. In particular,  $\TeX$  examines the depth of each line of text and the height of the succeeding line. If, when placing the usual  $\backslash\text{baselineskip}$ ,  $\TeX$  finds that the boundaries of the boxes containing the adjacent lines is closer than the value of  $\backslash\text{lineskiplimit}$ , then the normal procedure is aborted, and a glue equal to the value of  $\backslash\text{lineskip}$  is inserted. As we can see from figure 4, this can result in variable line spacing, making grid setting impossible.

We looked at the instances where  $\backslash\text{lineskip}$  had been applied. In most cases, the normal leading would have sufficed, as the oversized elements were

Here is some text to show what happens when we have a large mathematical construct in a line of text. Here is some text to show what happens when we have a large mathematical construct in a line of text. Here is some text to show what happens when we have a large mathematical construct in a line of text.  $\int_0^{\frac{\pi}{6}} \frac{1}{4} + 2 + 3$ . Here is some text to show what happens when  $\int_0^{\frac{\pi}{6}} \frac{1}{4} + 2 + 3$  we have a large mathematical construct in a line of text. Here is some text to show what happens when we have a large mathematical construct in a line of text.

**Figure 4:** Variation of leading when large inline maths is present.

not aligned with each other.  $\TeX$  does not know the horizontal position of the offending boxes, so in general there is no clash of text. Even when they were aligned, in most cases we could avoid the clash by reformatting the paragraph. The publications we were considering for grid setting contained only light mathematics, so we decided that we would do away with using `\lineskip` and just keep an eye out for clashing items, and deal with them on a case by case basis. So we chose the following values:

```
\lineskiplimit = -10pt
\lineskip = 0pt
```

This negative value for `\lineskiplimit` instructs  $\TeX$  not to apply `\lineskip` unless there is an overlap of more than 10pt between two adjacent lines. The value given to `\lineskip` is unimportant. Of course this might give very ugly overlapping lines, but we would pick these up while checking proofs, and we would deal with them manually. For seriously overlapping maths, we would change from inline to display math.

### 3.3 Dealing with floating elements

Floating elements, such as figures or tables, generally consist of the main float, namely the graphic or the table, a caption, and three glue items, one above the complete float, one below, and one separating the main element from the caption. In order to maintain grid setting, we need to control the vertical size of all these five elements. The three glue elements are set such that the total natural height is equal to an integral number of `\baselineskips`. The main element and the caption are rounded up or down, so that their heights are also an integral number of `\baselineskips`. This is done through a ‘`\roundoff`’ macro that is executed at run time.

Our general macro for floats is as follows:

```
\begin{figure}
\centering
```

```
\Xfigure{figure1}
\caption{Caption of figure.}
\label{fig1}
\end{figure}
```

which is similar to the normal  $\LaTeX$  float macro. We have made the control of spacing more useful by using `keyval.sty`, and adding the following options:

```
[beforegr = ...pt]
[aftergr = ...pt]
[beforecap = ...pt]
[aftercap = ...pt]
[line = ...]
```

These options allow the main element or the caption to be moved up or down. This is done before the `\roundoff` macro is executed, so the final height of each element will still be an integral number of `\baselineskips`. The final option is a negative or positive integer, and adjusts the three glue parameters such that the complete height of the float is an integral number of lines larger or smaller. This is useful in solving pagination problems.

### 3.4 Displayed equations

For equations that do not break across pages, we again use `\roundoff` to make them fit the grid. The display glues such as `\abovedisplayskip` are set to fixed amounts, as before.

### 3.5 The one problem: breaking displays

There is one problem we have not solved yet, namely that of automatically breaking displayed equations, and maintaining grid setting. The problem is that `\roundoff` produces one  $\TeX$  box, so  $\TeX$ ’s normal page breaking mechanism cannot be applied. For manuscripts with light mathematics, this is not a major problem, as the few such occurrences can be fixed manually, but it would be good to have an automated solution.

## 4 Results

Figures 5 and 6 show a double column page set on a grid. The floating figure and the mathematics are all rounded off so that the text is set on a grid.

We have found that the time taken in pagination has reduced considerably after using the grid macros. When we need to move some lines from one column to the next, it results in exactly the same number of lines being taken over from the second column. The `keyval` options in floats allows us to deal easily with widows and orphans, by making a float one line longer or shorter.

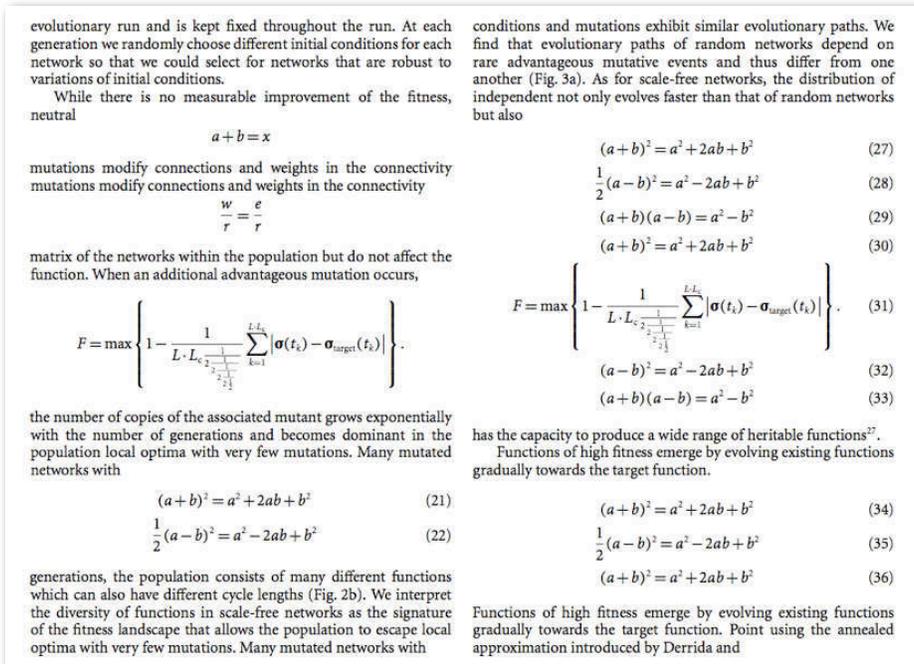


Figure 5: Example of a double column page with heavy maths, set on a grid.

## 5 Availability

We intend to release these macros under a free and open license. Our intention is to include them in a style file.

## 6 Acknowledgments

Hàn Th́ Thành did a lot of preliminary work in determining which route to take. CV Rajagopal helped in writing the macros. Jagath and Rishi did the refinements and testing.

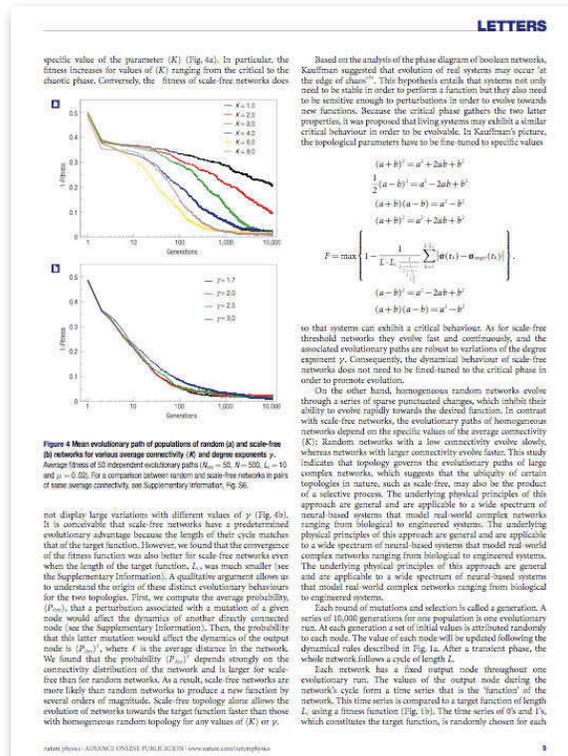


Figure 6: Example of a double column page set on a grid.

## Abstracts

(We hope that full papers will be published in a future issue. *Ed.*)

### **TypeSpec v.2: Typesetting font specimens**

William Adams

Stephen Moye's plain T<sub>E</sub>X TypeSpec macros converted to work as a L<sup>A</sup>T<sub>E</sub>X document class and organized into macros will be shown, including a number of new layouts. Additional ideas for specimen layouts will be solicited, and some will be constructed interactively during the presentation.

### **L<sup>A</sup>T<sub>E</sub>X and the different bibliography styles**

Federico Garcia

The myriad of different styles for bibliography and reference layouts can be, in the main, classified into three main families: the 'label' family, where references are denoted with a label, usually in [square brackets]; the 'author-year' family (Adorno, 1978); and the 'footnote' family. Although there are arguments for each of those families, the choice between them is, in the last analysis, decided by tradition: different disciplines have come to adopt different styles, and new generations of authors will naturally follow the uses of their predecessors (which are also enforced institutionally, for example with journal guidelines).

L<sup>A</sup>T<sub>E</sub>X itself (with BIB<sub>T</sub>E<sub>X</sub>) is designed toward the label family. Some packages, like cite, provide extra functionality in that family. The other two families are reflected in the L<sup>A</sup>T<sub>E</sub>X world by special packages. The author-year family is well illustrated by packages such as harvard, natbib, and achicago. Footnote-style references are implemented by the package opcit. A basic description of these packages follows. I will spend relatively longer with opcit, which is comparatively recent (2002) and the one I know best.

### **Creation of a PostScript Type 1 logo font with MetaType1**

Klaus Höppner

MetaType1 is a tool created by Bogusław Jackowski, Janusz Nowacki, and Piotr Strzelczyk for creation of PostScript Type 1 fonts. It uses MetaPost, t<sub>l</sub>utils and some awk scripts to create a MetaPost font source with some special macros.

MetaPost was used to create the Latin Modern fonts, which are derived from Computer Modern fonts but include many additional characters, especially accented ones. It is part of most modern T<sub>E</sub>X distributions. Some original fonts, notably Iwona and Kurier, have also been created by the developers of MetaType1.

I came into touch with MetaPost when I wanted to convert an existing logo font from METAFONT to PostScript Type 1. Unfortunately there exists no tutorial or cookbook for using MetaType1. So I started to play

with the example fonts supplied as part of MetaType1 and to read the comments in the source. This tutorial gives a simple example and the lessons I learned.

### **Common macro pitfalls and how to avoid them**

Ned W. Hummel

In the process of learning L<sup>A</sup>T<sub>E</sub>X there are a number of common pitfalls that many of us fall into at some point. Most of us encounter these pitfalls when writing macros for the first time. One of the great advantages of L<sup>A</sup>T<sub>E</sub>X is the ability to logically markup our document. Unfortunately, a number of us tend not to apply that same logical markup philosophy when writing macros.

We will consider several examples and discuss ways to re-write them using a logical markup philosophy.

### **A wayward wayfarer's way to T<sub>E</sub>X**

Stephen Moye

The amusing recollections of one particular humanities T<sub>E</sub>X user's adventures in T<sub>E</sub>XLand.

### **Fonts, typefaces, glyphs & sorts**

Steve Peter

This presentation focuses on the general characteristics and usages of typefaces, without specific reference to T<sub>E</sub>X. I will begin by covering the history of printing technologies and offering an overview of some useful classification schemes for typefaces. I then turn to a practical discussion of selecting the right typeface for the right job, with a nod toward using T<sub>E</sub>X to its fullest.

### **Introduction to memoir**

Steve Peter

This presentation serves as a gentle introduction to Peter Wilson's memoir class, an alternative to the standard L<sup>A</sup>T<sub>E</sub>X classes. Memoir is quite flexible, and makes it easy to create beautiful book, article, and report designs, without having to search for, install, and load numerous third-party packages.

### **T<sub>E</sub>X and after dinner speaking**

Alan Wetmore

I will discuss a somewhat novel use for T<sub>E</sub>X, preparing an after dinner speech for a scientific conference. My experience some years ago required me to prepare, at quite short notice when a scheduled dignitary was forced to cancel, an entertaining diversion for the attendees of a conference banquet. Inspired by the then-current popularity of *Who wants to be a Millionaire?*, and *The Weakest Link*, I produced a domain-specific trivia "contest" based on some frenzied Internet sleuthing. Formatted using pdfscreen and pdfL<sup>A</sup>T<sub>E</sub>X I produced an attractive presentation for the audience. In the process I learned a little about various T<sub>E</sub>X's presentation capabilities. Some examples will be given.

# Calendar

## 2007

- Mar 7–9 DANTE 2007, 36<sup>th</sup> meeting, Westfälische Wilhelms-Universität, Münster, Germany. For information, visit <http://www.dante.de/dante2007>.
- Feb 9– Mar 18 Guild of Book Workers 100th Anniversary Exhibition: A traveling juried exhibition of books by members of the Guild of Book Workers. Utah Museum of Fine Arts, Salt Lake City, Utah. Sites and dates are listed at <http://palimpsest.stanford.edu/byorg/gbw>.
- Mar 24–25 First International ConT<sub>E</sub>Xt User Meeting, Epen, The Netherlands. For information, visit <http://context.aanhet.net/epen2007>.
- Apr 9– May 20 Guild of Book Workers 100th Anniversary Exhibition: A traveling juried exhibition of books by members of the Guild of Book Workers. Branford P. Millar Library, Portland State University, Oregon. Sites and dates are listed at <http://palimpsest.stanford.edu/byorg/gbw>.
- Apr 23 **TUG 2007** (July 17–20), abstracts due. For information, visit <http://www.tug.org/tug2007/>.
- Apr 28– May 2 17<sup>th</sup> EuroT<sub>E</sub>X Conference + 15<sup>th</sup> BachoT<sub>E</sub>X Conference = EuroBachoT<sub>E</sub>X 2007, Bachotek, Poland. For information, visit <http://www.gust.org.pl/BachoTeX/EuroBachoTeX2007>.
- May 31– Jun 1 Sixth Annual St. Bride Conference, “Great British Design?”, London, England. For information, visit <http://stbride.org/friends/conference>.
- Jun 4– Aug 3 Rare Book School, University of Virginia, Charlottesville, Virginia. Many one-week courses on type, bookmaking, printing, and related topics. For information, visit <http://www.virginia.edu/oldbooks>.

---

## TUG 2007 Practicing T<sub>E</sub>X, San Diego, California.

- Jul 17 Workshops (free for attendees).
- Jul 18–20 The 28<sup>th</sup> annual meeting of the T<sub>E</sub>X Users Group. For information, visit <http://www.tug.org/tug2007>.
- 
- Aug 1–5 TypeCon 2007, Seattle, Washington. For information, visit <http://www.typecon.com/>.
- Aug 5–9 SIGGRAPH 2007, San Diego, California. For information, visit <http://www.siggraph.org/s2007/>.
- Aug 6–10 *Extreme* Markup Languages 2007, Montréal, Québec. For information, visit <http://www.extrememarkup.com/extreme/>.
- Aug 28–31 ACM Symposium on Document Engineering, University of Manitoba, Winnipeg, Canada. For information, visit <http://www.documentengineering.org/>.
- Sep 12–16 Association Typographique Internationale (ATypI) annual conference, Brighton, UK. For information, visit <http://www.atypi.org/>.
- Sep 18–19 Conference on “Non-Latin typeface Design”, St. Bride Library, London, and the Department of Typography, University of Reading, UK. For information, visit [http://stbride.org/events\\_education/events/](http://stbride.org/events_education/events/).
- Oct 11–13 American Printing History Association 2007 annual conference, “Transformations: The persistence of Aldus Manutius”, University of California at Los Angeles. For information, visit <http://www.printinghistory.org/>.

*Status as of 1 March 2007*

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>.

# TUG 2007: Practicing T<sub>E</sub>X

Workshops and presentations on

L<sup>A</sup>T<sub>E</sub>X, T<sub>E</sub>X, MetaPost,  
ConT<sub>E</sub>Xt, LuaT<sub>E</sub>X,  
and more

July 17–20, 2007

San Diego State University  
San Diego, California, USA

<http://tug.org/tug2007>  
[tug2007@tug.org](mailto:tug2007@tug.org)

**Keynote address:** *Peter Wilson,*  
The Herries Press

- April 23, 2007 — presentation proposal deadline
- May 18, 2007 — early bird registration deadline
- July 17–20, 2007 — workshop and conference



## Further information

Conference attendees will enjoy an opening night reception and an (optional) banquet one evening. Coffee and lunch will be served each day of the meeting. Located on the campus of San Diego State University, an easy trolley ride from downtown San Diego. Inexpensive campus housing is available.

Conference fee, hotel, and other information is available on the web site.

## Sponsorship

If you'd like to support the conference, promote T<sub>E</sub>X products and services, or otherwise provide sponsorship, see the web site for donation and advertising options.

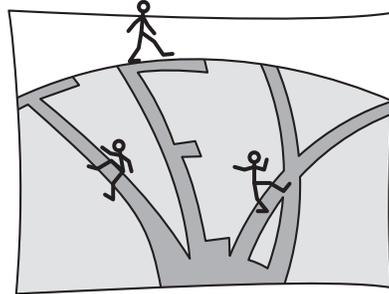
We thank the present sponsors: the German-speaking T<sub>E</sub>X users group DANTE e.V., von Hoerner & Sulger GmbH, and Adobe Systems Inc. have provided generous support; San Diego State University is our host; and special thanks to the many individual contributors.

Hope to see you there!

*Sponsored by the T<sub>E</sub>X Users Group*

## Invitation to EuroT<sub>E</sub>X 2007

The XVII European T<sub>E</sub>X Conference, EuroT<sub>E</sub>X 2007, April 28th until May 2nd, 2007, is organized jointly by CSTUG, the Czechoslovak T<sub>E</sub>X Users Group and GUST, the Polish T<sub>E</sub>X Users Group, at Bachotek, near Brodnica, in the north-east of Poland. This is the place where the annual GUST BachoT<sub>E</sub>X conferences are organized yearly since 1993. EuroT<sub>E</sub>X 2007 will also be the XV BachoT<sub>E</sub>X, hence it is also called EuroBachoT<sub>E</sub>X 2007.



The conference motto is

*T<sub>E</sub>X: Paths to the Future*

Looking into the future of T<sub>E</sub>X seems to be justified by some recent developments around our beloved system. In no particular order:

- new pdfT<sub>E</sub>X release,
- METAPOST v. 1.1,
- a batch of new font families from the new project called T<sub>E</sub>X Gyre,
- a working version of Omega 2,
- LuaT<sub>E</sub>X,
- X<sub>Y</sub>T<sub>E</sub>X, and
- many more.

To where will they lead us? What potential do they have? Will they attract more users? Will they converge and if so will we get a new quality? Will we be able to typeset documents better? Or perhaps easier? These and more questions will and should be asked and discussed during the conference.

Watch the conference site:

<http://www.gust.org.pl/BachoTeX/EuroBachoTeX2007>

It contains all the necessary information and is regularly updated as new information becomes available.

Please contribute papers to make EuroBachoT<sub>E</sub>X 2007 even more interesting. Dates for abstracts and paper submissions are on the conference web page.

Help advertise EuroBachoT<sub>E</sub>X! You can freely use its poster

<http://www.gust.org.pl/BachoTeX/EuroBachoTeX2007/poster.html>

And, of course: put this event into your calendar and then come and join the T<sub>E</sub>Xies from around Europe and the world. You are indispensable here!

Jerzy Ludwichowski  
(for the Organizing Committee)

## TUG Business

### TUG 2007 election report

Steve Peter  
for the Elections Committee

Nominations for TUG President and the Board of Directors in 2007 have been received and validated. Because there is a single nomination for the office of President, and fewer nominations for Board of Directors than there are open seats, there will be no requirement for a ballot this election.

For President, Karl Berry was nominated. As there were no other nominees, he is duly elected and will serve for another two years.

For the Board of Directors, the following individuals were nominated: Barbara Beeton, Jon Breitenbucher, Kaja Christiansen, Sue DeMeritt, Ross Moore, Cheryl Ponchin, and Philip Taylor. As there were fewer nominations than open positions, all the nominees are duly elected.

Terms for both President and members of the Board of Directors will begin with the 2007 Annual Meeting in San Diego. Congratulations to all.

Sam Rhoads has decided to step down at the end of his term this year. On behalf of the Board, I wish to thank him for his service. The bonds with the TUG Board are not entirely loosed just yet, though, as he will continue to chair the Bursary Committee.

Continuing board members, with terms ending in 2009, are: Steve Grathwohl, Jim Hefferon, Klaus Höppner, Arthur Ogawa, Steve Peter and Dave Walden. Also, Dick Koch has agreed to an appointment to the Board starting with the 2007 Annual Meeting; his term will also expire in 2009.

Statements for all the candidates, both for President and for the Board, are appended (in alphabetical order). They are also available online at <http://www.tug.org/election>, along with announcements and results of previous elections.

◇ Steve Peter  
for the Elections Committee

### Barbara Beeton



#### Biography:

For the T<sub>E</sub>X Users Group:

- charter member of the T<sub>E</sub>X Users Group; charter member of the TUG Board of Directors;
- *TUGboat* production staff since 1980, Editor since 1983;
- committees: publications, bylaws, elections;
- chair, Technical Working Group on Extended Math Font Encoding;
- liaison from Board to Knuth Scholarship Committee 1991–1992.

Employed by American Mathematical Society:

- Staff Specialist for Composition Systems; involved with typesetting of mathematical texts since 1973; assisted in initial installation of T<sub>E</sub>X at AMS in 1979; implemented the first AMS document styles; created the map and ligature structure for AMS cyrillic fonts.
- Standards organizations: active 1986–1997 in: ANSI X3V1 (Text processing: Office & publishing systems), ISO/IEC JTC1/SC18/WG8 (Document description and processing languages); developing the standard ISO/IEC 9541:1991 Information technology–Font information interchange.
- AFII (Association for Font Information Interchange): Board of Directors, Secretary 1988–1996.
- STIX representative to the Unicode Technical Committee for adoption of additional math symbols.

#### Personal statement:

Four years ago, I expected to be retiring from the AMS about now, but I've decided I'm not ready yet. And the ability to participate in the decisions that help direct TUG is still important to me too.

I recently read about a new position: “Corporate Memory Officer”. I think that even without the title, that's the function I perform—letting other members of the Board know whether something has been tried before, and with what results, to help inform decisions, without (I hope) standing in the way.

With support from the members of this wonderful community, I'd like to continue for four more years.

**Karl Berry**

## Biography:

I have served as TUG president since 2003 and was a board member for two terms prior to that. During my term as president, we've enacted new initiatives, notably expanding the scope of the special member and institutional memberships. We've also partnered with Addison-Wesley for online book sales, with Bigelow&Holmes for making the Lucida fonts available through TUG and with Adobe making the Utopia typeface family freely available.

As president, I also coordinate the formal and informal meetings of the Board, provide direction and oversight to the Executive Director, and monitor TUG's financial transactions. I also serve on the conference committee, and thus have been one of the organizers for all TUG-sponsored conferences since 2004, both the annual meetings and the Practical  $\TeX$  conferences, including web site and program creation, coordination of publicity, and so forth.

I have been on the TUG technical council for many years. I co-sponsored the creation of the  $\TeX$  Development Fund in 2002, and am one of the primary system administrators and webmasters for the TUG servers. I'm also one of the production staff for the *TUGboat* journal and have driven the successful effort to get it back on schedule.

On the  $\TeX$  development side, I'm currently editor of  $\TeX$  Live, the largest free software  $\TeX$  distribution, and thus coordinate with many other  $\TeX$  projects around the world, such as pdf $\TeX$  and MetaPost. I developed and still work on Web2c (Unix  $\TeX$ ) and Kpathsea, a freely redistributable library for path searching, Eplain (a macro package extending plain  $\TeX$ ), GNU Texinfo, and many other projects. I was also a co-author of  *$\TeX$  for the Impatient*, an early comprehensive book on  $\TeX$ , which is now freely available. I first encountered and installed  $\TeX$  in 1982, as a college undergraduate.

## Personal statement:

I believe TUG can best serve its members and the general  $\TeX$  community by working in partnership with the other  $\TeX$  user groups worldwide, and sponsoring projects and conferences that will increase interest in and use of  $\TeX$ . I've been fortunate enough to be able to work essentially full time, pro bono, on TUG and  $\TeX$  activities the past several years, and plan to continue doing so if re-elected. It would be an honor to serve another term.

**Jon Breitenbucher**

## Biography:

I am currently an Adjunct Professor and Instructional Technology Specialist at the College of Wooster. I began using  $\TeX$  and  $\LaTeX$  in 1992 while a graduate student at The Ohio State University. I helped customize the thesis class while at OSU and since then have written one for Wooster's Independent Study Thesis (IS). I have also created templates for homework solutions and papers while at Wooster.

## Personal statement:

If elected my desire is to help introduce  $\LaTeX$  to undergraduates and to encourage undergraduates to join TUG. Over the past five years I have helped over 70 undergraduates learn  $\LaTeX$  for their IS projects. I have also formed a self sustained community at Wooster. I would like to find ways in which TUG can help others who are doing the same things at their institutions or in their communities. I would also like to see TUG develop a mechanism to turn these  $\TeX$  initiates into members.

**Kaja Christiansen**

## Biography:

I live in the city of Århus, Denmark and work at the University of Aarhus. My job at the Department of Computer Science involves system administration and software support, including the responsibility for all aspects of  $\TeX$  & friends: local styles, in-house classes, (very) frequent user support and maintainance. The department has about 550 students, 80 employees, a large number of active research groups and close ties to the BRICS Research Centre.

I heard about  $\TeX$  for the first time in fall of 1979. In Palo Alto at the time, I wanted to audit courses at Stanford; my top priority was lectures by Prof. Donald Knuth but that, I was told, was not possible as Prof. Knuth was on leave due to work on *a text processing project* . . . This project was  $\TeX$ ! Back home, it didn't take long till we had a runnable

system and thus introduced an early version of  $\text{\TeX}$  in Denmark.

Personal statement:

I have served as the chair of TUG's Technical Council since 1997 and co-sponsored the creation of  $\text{\TeX}$  Development Fund. I share system administrator's responsibilities for the TUG server (which access to the Internet is currently facilitated by my Department). In my role as a member of the board, my special interests have been projects of immediate value to the  $\text{\TeX}$  community:  $\text{\TeX}$  Live, *TUGboat* and TUG's web site. Since September 2002, I have also served as the president of the Danish  $\text{\TeX}$  Users Group (DK-TUG).

### Susan Demeritt



My name is Susan DeMeritt, I live in Lakeside, California (just outside San Diego).

I am employed by the Center for Communications Research, La Jolla, in San Diego, California for almost 18 years doing technical typing in the Publications Department. I started the position learning  $\text{\TeX}$  and working up to  $\text{\LaTeX} 2_{\epsilon}$ . I enjoy using  $\text{\LaTeX} 2_{\epsilon}$  to typeset mathematical and scientific papers.

I have been a member of the  $\text{\TeX}$  Users Group since 1989. I have been a member of the Board of Directors since March of 1998, and Secretary since 2001. I really enjoy being part of the Board of Directors of the  $\text{\TeX}$  Users Group and I hope my participation has been helpful.

I have successfully taught (along with Cheryl Ponchin) three  $\text{\LaTeX}$  classes, one at Rutgers University, one at Duke University, and one at the University of Delaware.

### Richard Koch



After degrees from Harvard and Princeton, I taught mathematics at the University of Oregon from 1966 to 2002, working on pseudogroups and filtered Lie algebras. For the last fifteen of these years I was director of the Undergraduate Program in Mathematics, and won the University's Ersted and Hermann teaching awards.

I began using  $\text{\TeX}$  on a NeXt cube, running Tomas Rokicki's  $\text{\TeX}$ View. After Apple bought NeXt, the Unix  $\text{\TeX}$  binaries were ported to OS X, but predictions from campus Apple representatives that a  $\text{\TeX}$  front end would follow didn't materialize. So I wrote  $\text{\TeX}$ Shop, a  $\text{\TeX}$  front end, releasing it while OS X was still in beta. In those days, Apple's pdf display engine didn't understand non-native fonts and  $\text{\TeX}$ Shop users had to use the Times Roman font and avoid mathematical symbols.

Work on  $\text{\TeX}$ Shop continues. Recently I have been involved in maintaining the Mac $\text{\TeX}$  install package for OS X. This package was written by Jonathan Kew, and originally based on Gerben Wierda's redistribution of  $\text{\TeX}$ . This year's version is based on the full  $\text{\TeX}$  Live 2007.

From  $\text{\TeX}$  meetings I know that these are just tips of the iceberg, and enjoy hearing about the exciting developments underway from a host of people which affect  $\text{\TeX}$  on all platforms.

### Ross Moore



I am an academic mathematician, living in Sydney, Australia.

Since the mid-80s I have been a user of  $\text{\TeX}$  and  $\text{\LaTeX}$ , mostly for mathematical applications, such as journal articles, books and Proceedings volumes. The need for special content layouts has led to my involvement in the development of several packages and other software, most notably  $\text{\Xy-pic}$  and  $\text{\LaTeX} 2_{\text{HTML}}$ , both of which I have presented at TUG annual meetings.

My first TUG meeting, in 1997, saw me joining the TUG Board of Directors where I have served ever since, and wish to continue to serve for at least

another term. For TUG I've worked on the Technical Council, the Publications Committee, assisted with organising annual meetings, been the contact person for the technical working groups TFAA and MacTeX (though the important work is done by others), and administer email discussion groups ( $\LaTeX$ 2HTML, Xy-pic, X<sub>Y</sub>TeX). Frequently I answer queries and give advice on the 'TeX on Mac OS X' mailing list, for Macintosh users of TeX.

Currently I am working with Chris Rowley, Will Robertson and others, preparing  $\LaTeX$  support for mathematics in the new world of Unicode and STIX fonts, which is soon to be upon us. This is in addition to working on ways to generate webs of inter-linked mathematical documents, as web-pages and in PDF format, for online journals, conference abstracts, and encyclopaedic collections, all generated from  $(\LaTeX)$  sources.

For the TUG board, I feel that my experience as both a TeX programmer, as well as a mathematical author and editor, provides a detailed understanding of how TeX and  $\LaTeX$  have been used in the past, as well as insight into new ways that the TeX family of programs may be used in coming years.

### Cheryl Ponchin



My name is Cheryl Ponchin, I live in Plainsboro, New Jersey, and I am employed by the Center for Communications Research in Princeton. I have been a technical typist for more than 20 years. I started with TeX and I am now using  $\LaTeX 2_{\epsilon}$  as well as many of the different packages available. I enjoy using this software to typeset mathematical and scientific papers.

I have been a member of the TeX Users Group since 1989. I have been a member of the TUG Board of Directors since March of 1998. I really enjoy being part of the TUG Group.

I have taught  $\LaTeX$  classes for TUG on my own and with Sue DeMeritt, as well as other classes for Princeton University. I was also asked my opinion on *A Guide to  $\LaTeX 2_{\epsilon}$* , which was very interesting and rewarding for me.

### Philip Taylor



Philip Taylor has been involved with TeX for approximately 20 years, ever since seeing an example of typeset copy produced using TeX by a friend at British Aerospace. Despite regular attempts to understand why anyone might use  $\LaTeX$ , he remains completely baffled and continues to believe that Plain TeX (or, even better, IniTeX) and a few home-grown macros are all that anyone could ever need. On a more serious note, he is completely convinced by the arguments in favour of semantic markup, and believes that TeX should adopt the HTML/CSS model, with one language used for document markup and a second, quite different, language used to ascribe appearance to semantics. He is also convinced that, had it not been for Hàn Thê Thành's invention and development of PdfTeX, the future for TeX might well have been very bleak indeed.

As a long-serving TUG Board member (a classic example of poacher-turned-gamekeeper), Phil is rarely willing to accept the *status quo*, and regularly argues that TUG should be more receptive to innovative suggestions from its members, no matter how much these ideas might challenge the current received wisdom.

---

## TUG financial statements for 2006

Dave Walden, TUG treasurer

This financial report for 2006 has been reviewed by the TUG board but has not been audited. It may change slightly when the final 2006 tax return is filed. TUG's tax returns are publicly available on our web site: <http://www.tug.org/tax-exempt>.

### Revenue highlights

Revenue increased 21 percent for 2006 compared to 2005. Total membership dues were \$102.6K at the end of 2006, compared to \$91.1K in 2005; this resulted from essentially flat membership from 2005 to 2006 combined with the first dues increase in many years. We ended 2006 with 1492 paid members. The auto-renewal option initiated in 2006 was chosen by 176 members, and 114 members selected the new electronic-only option.

TUG had \$31.6K in income in 2006 from other sources than membership fees. Three areas of particular note:

- Contribution income from generous TUG members and individuals worldwide increased by about \$1,000 from 2005 to 2006 (this included a *one-time* contribution of \$5K from [LinuxFund.org](http://www.linuxfund.org)).
- TUG store revenue of \$11.6K included significant sales of:
  - the Lucida font collection through our arrangement with Bigelow & Holmes
  - T<sub>E</sub>X CDs and DVDs
  - discounted T<sub>E</sub>Xnical books, through our arrangement with the Pearson Publishing Group (which includes Addison-Wesley)
- Interest income was up from 2005, on account of increased interest rates and the above increases in income and thus cash on hand.

### Cost of Goods Sold and Expenses highlights

Payroll and office expenses, *TUGboat* production and mailing, and software 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) by phasing out use of temporary office help.

Software production and mailing was budgeted and accrued in 2006 although the actual shipment is scheduled for early 2007.

*TUGboat* production and mailing (which included the EuroT<sub>E</sub>X 2005 proceedings and two normal issues of *TUGboat*) averaged over \$9,000 in 2006. This was up in 2006 from 2005 for two primary

reasons: (1) more pages in 2006 than in 2005, and (2) higher than average expenses for the EuroT<sub>E</sub>X proceedings.

A significant part of the Postage/Delivery—Members line item is individually mailing issues of *TUGboat* and software discs as members join throughout the year.

In 2006, TUG made the usual contributions of \$2,000 to the TUG Bursary and \$1,000 to EuroT<sub>E</sub>X. The 2006 contributions budget was less than the contributions for 2005 because of the Board's uncertainty about membership numbers in the face of the 2006 dues increase. The 2007 budget includes an increase in contributions.

### The bottom line

Netting the major line items of Revenue, Cost of Goods Sold, and Expenses, TUG had a gain of \$17,536 for the year, compared with a net ordinary income loss of almost \$3,000 in 2005.

There was a small prior year adjustment of \$−1,785, shown near the bottom of the Profit and Loss comparison. This resulted from an underestimate of the cost of publication in early 2006 of the last *TUGboat* issue of 2005 and unpaid invoices from 2005 which have been written off.

### Balance sheet highlights

The increased income mentioned above, combined with continued care with expenses, resulted in a significantly higher end-of-year assets level in 2006 compared with 2005.

The year-end accounts receivable is primarily for reimbursement of unused bursary funds. The final payment is due in February 2007.

The Committed Funds come to TUG specifically designated for the L<sup>A</sup>T<sub>E</sub>X project, the T<sub>E</sub>X Development fund, etc.; they have been allocated accordingly and are disbursed as the projects progress. TUG charges no overhead for administering these funds.

The payroll liabilities are for 2006 state and federal taxes due January 15, 2007.

### Summary

TUG was in better financial condition at the end of 2006 than at the end of 2005. We are hopeful this will continue in 2007. As announced elsewhere, there is no fee increase in 2007. Also, as mentioned above, the TUG board is planning to increase direct TUG contributions (fund more T<sub>E</sub>X development) in 2007. TUG continues to work closely with the local user groups and ad hoc committees on many activities to benefit T<sub>E</sub>X and its users.

## TUG 12/31/2006 (versus 2005) Balance Sheet

ASSETS	Jan - Dec 06	Jan - Dec 05
Current Assets		
Checking/Savings	\$133,790	\$115,994
Accounts Receivable	\$395	\$635
Other Current Assets		\$728
Total Current Assets	\$134,185	\$117,357
Fixed Assets	\$5,224	\$5,591
<b>TOTAL ASSETS</b>	<b>\$139,409</b>	<b>\$122,948</b>
LIABILITIES & EQUITY		
Liabilities		
Late TUGboat Accrual		\$7,000
Software Delay until 2007	\$6,500	
Committed Funds	\$9,322	\$7,005
Prepaid Member Income	\$1,710	
Payroll Liabilities	\$1,057	\$1,037
Deferred Conf Donations		\$1,794
Deferred Member Income		\$1,160
Total Liabilities	\$18,589	\$17,996
Equity		
Equity as of 1/1	\$104,972	\$117,722
Net Income	\$15,848	-\$12,770
Total Equity	\$120,820	\$104,952
<b>TOTAL LIABILITIES &amp; EQUITY</b>	<b>\$139,409</b>	<b>\$122,948</b>

## TUG 2006 (versus 2005) Revenue and Expenses

Ordinary Income/Expense	Jan - Dec 06	Jan - Dec 05
Income		
Membership Dues	101,669	91,173
Product Sales	11,776	7,410
Contributions Income	11,376	7,939
Practical TeX Conference	2,909	406
Conference Classes	965	
Annual Conference	-275	-204
Interest Income	4,589	3,672
Advertising Income	370	200
Total Income	133,379	110,596
Cost of Goods Sold		
TUGboat Prod/Mailing	28,998	18,626
Software Production/Mailing	6,500	8,092
Postage/Delivery - Members	2,702	4,874
Conf Expense, office + overhead	1,651	2,082
Copy/Printing for members	60	300
Total COGS	39,911	33,974
Gross Profit	93,468	76,622
Expense		
Contributions made by TUG	3,000	4,950
Office Overhead	12,229	13,411
Payroll Exp	58,622	59,066
Professional Fees	318	119
Depreciation Expense	1,667	2,041
Total Expense	75,836	79,587
Net Ordinary Income	17,632	-2,965
Other Income/Expense		
Prior year adjust	-1,785	-9,784
Total Other Income	-1,785	-9,784
Net Other Income	-1,785	-9,784
<b>Net Income</b>	<b>15,847</b>	<b>-12,749</b>

## Institutional Members

Aalborg University, Department of Mathematical Sciences, Aalborg, Denmark

American Mathematical Society, Providence, Rhode Island

Banca d'Italia, Roma, Italy

Center for Computing Sciences, 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

Marquette University, Department of Mathematics, Statistics and Computer Science, Milwaukee, Wisconsin

Masaryk University, Faculty of Informatics, Brno, Czech Republic

Moravian College, Department of Mathematics and Computer Science, Bethlehem, Pennsylvania

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

United States Environmental Protection Agency, Narragansett, Rhode Island

University College, Cork, Computer Centre, Cork, Ireland

University of Delaware, Computing and Network Services, Newark, Delaware

Université Laval, Ste-Foy, Québec, Canada

Universiti Tun Hussein Onn Malaysia, Pusat Teknologi Maklumat, Batu Pahat, Johor, Malaysia

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

Vanderbilt University, Nashville, Tennessee

# TeX Users Group Membership Form 2007



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 DeMeritt  
*Executive Director* Robin Laakso

TUG membership rates are listed below. Please check the appropriate boxes and mail the completed form with payment (in US dollars) to the mailing address at left. If paying by credit/debit card, you may alternatively fax the form to the number at left or join online at <http://tug.org/join.html>. The web page also provides more information than we have room for here.

Status (check one)  New member  Renewing member

Automatic membership renewal in future years

Using the same payment information; contact office to cancel.

	Rate	Amount
<input type="checkbox"/> <b>Early bird membership</b> for 2007 After May 31, dues are \$85.	\$75	_____
<input type="checkbox"/> <b>Special membership</b> for 2007 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 2007 (non-voting)	\$95	_____
<input type="checkbox"/> <b>Institutional membership</b> for 2007 Includes up to eight individual memberships.	\$500	_____
<input type="checkbox"/> <b>Don't ship any physical benefits</b> ( <i>TUGboat</i> , software) deduct \$20 <i>TUGboat</i> and software distributions are available electronically.		_____
<b>Receive software on CD</b> (always shipped on DVD)		
<input type="checkbox"/> Send TeX Live 2007 on CD	\$10	_____
<input type="checkbox"/> Send proTeXt 2007 on CD	\$10	_____
<input type="checkbox"/> Send CTAN 2007 on CD	\$15	_____
<b>Purchase last year's materials:</b>		
<input type="checkbox"/> <i>TUGboat</i> volume for 2006 (3 issues)	\$20	_____
<b>Voluntary donations</b> (or see <a href="https://www.tug.org/donate.html">https://www.tug.org/donate.html</a> )		
<input type="checkbox"/> General TUG contribution		_____
<input type="checkbox"/> Bursary Fund contribution		_____
<input type="checkbox"/> TeX Development Fund contribution		_____
<input type="checkbox"/> L <sup>A</sup> TeX 3 contribution		_____
<input type="checkbox"/> MacTeX contribution		_____
<input type="checkbox"/> CTAN 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.

Name \_\_\_\_\_

Department \_\_\_\_\_

Institution \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State/Province \_\_\_\_\_

Postal code \_\_\_\_\_ Country \_\_\_\_\_

Email address \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

Position \_\_\_\_\_ Affiliation \_\_\_\_\_

## T<sub>E</sub>X Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at <http://tug.org/consultants.html>. If you'd like to be listed, please fill out the form 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  
+1 561-966-8400

Email: [kinch \(at\) truetex.com](mailto:kinch(at)truetex.com)

Publishes TrueT<sub>E</sub>X, a commercial implementation of T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. Custom development for T<sub>E</sub>X-related software and fonts.

### **Martinez, Mercè Aicart**

Tarragona 102 4<sup>a</sup> 2<sup>a</sup>  
08015 Barcelona, Spain  
+34 932267827

Email: [m.aicart \(at\) menta.net](mailto:m.aicart(at)menta.net)

Web: [www.edilatex.com/](http://www.edilatex.com/)

We provide, at reasonable low cost, T<sub>E</sub>X 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. For more information visit our web site.

### **MicroPress Inc.**

68-30 Harrow Street  
Forest Hills, NY 11375  
+1 718-575-1816; fax: +1 718-575-8038

Email: [support \(at\) micropress-inc.com](mailto:support(at)micropress-inc.com)

Web: [www.micropress-inc.com](http://www.micropress-inc.com)

Makers of V<sub>T</sub>E<sub>X</sub>, fully integrated T<sub>E</sub>X system running on Windows. V<sub>T</sub>E<sub>X</sub> system is capable of one-pass output of PDF, PS, SVG and HTML; V<sub>T</sub>E<sub>X</sub> IDE includes Visual Tools for writing equations, function plots and other enhancements

and a large number of fonts, many not available elsewhere. Makers of many new and unique mathematical font families for use with T<sub>E</sub>X, see <http://www.micropress-inc.com/fonts>. Makers of microIMP, a fully WYSIWYG L<sup>A</sup>T<sub>E</sub>X-based Word Processor. microIMP supports T<sub>E</sub>X and AMST<sub>E</sub>X math, lists, tables, slides, trees, graphics inclusion, many languages and much else — all without any need for knowing T<sub>E</sub>X commands, see <http://www.microimp.com>. See our web page for other products and services. Serving T<sub>E</sub>X users since 1989.

### **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 T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, and ConT<sub>E</sub>Xt, 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  
+1 (703) 860-0013

Email: [borisv \(at\) lk.net](mailto:borisv(at)lk.net)

Web: <http://borisv.lk.net>

T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X consulting, training and seminars. Integration with databases, automated document preparation, custom L<sup>A</sup>T<sub>E</sub>X packages, conversions and much more. I have about twelve years of experience in T<sub>E</sub>X and twenty-five years of experience in teaching & training. I have authored several packages on CTAN and published papers in T<sub>E</sub>X related journals.



**Table of Contents** (ordered by difficulty)**Introductory**

- 3 *Barbara Beeton* / Editorial comments
  - typography and *TUGboat* news
- 29 *Barbara Beeton* / How to create a T<sub>E</sub>X Journal: A personal journey
  - a look back at *TUGboat* experiences since its founding
- 3 *Karl Berry* / From the President
  - some TUG activities and information for 2006
- 65 *Jon Breitenbucher* / L<sup>A</sup>T<sub>E</sub>X at a liberal arts college
  - experiences introducing L<sup>A</sup>T<sub>E</sub>X in an undergraduate liberal arts setting
- 61 *Elizabeth Dearborn* / T<sub>E</sub>X and medicine
  - experiences in self-publishing a medical transcription dictionary with T<sub>E</sub>X
- 126 *Peter Flom* / L<sup>A</sup>T<sub>E</sub>X for academics and researchers who (think they) don't need it
  - advocating L<sup>A</sup>T<sub>E</sub>X for scientists and dispelling myths
- 124 *Jim Hefferon* / L<sup>A</sup>T<sub>E</sub>X resources
  - a sampling of useful documentation, web sites, programs, and packages
- 84 *Troy Henderson* / A beginner's guide to MetaPost for creating high-quality graphics
  - introduction to MetaPost
- 24 *L<sup>A</sup>T<sub>E</sub>X Project Team* / L<sup>A</sup>T<sub>E</sub>X news, issue 17
  - latest L<sup>A</sup>T<sub>E</sub>X release notes: font encodings, graphics drivers, more
- 70 *Boris Veytsman* / Design of presentations: Notes on principles and L<sup>A</sup>T<sub>E</sub>X implementation
  - writing effective presentations and implementing them in L<sup>A</sup>T<sub>E</sub>X
- 49 *David Walden* / A lifetime as an amateur compositor
  - productivity with L<sup>A</sup>T<sub>E</sub>X and typesetting experiences along the way

**Intermediate**

- 133 *Kaveh Bazargan* / Removing vertical stretch — mimicking traditional typesetting with T<sub>E</sub>X
  - a method for typesetting on a grid, including double columns and math
- 129 *Federico Garcia* / Hypertext capabilities with pdfL<sup>A</sup>T<sub>E</sub>X
  - introduction to the hyperref package, links, bookmarks, and more
- 15 *Mark LaPlante* / The treasure chest
  - selected new CTAN packages in 2006
- 91 *Andrew Mertz and William Slough* / Graphics with PGF and TikZ
  - graduated examples of graphics within L<sup>A</sup>T<sub>E</sub>X using a MetaPost-like syntax
- 20 *Ignacio Llopis Tortosa and María José Castro Bleda* / paperT<sub>E</sub>X: Creating newspapers using L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>
  - automatically generating a PDF newspaper from external sources
- 77 *Boris Veytsman and Maria Shmilevich* / Automatic report generation with Web, T<sub>E</sub>X and SQL
  - creating high-quality project reports while minimizing overhead

**Intermediate Plus**

- 110 *Peter Flynn* / Rolling your own Document Class: Using L<sup>A</sup>T<sub>E</sub>X to keep away from the Dark Side
  - 5 *Oleg Parashchenko* / T<sub>E</sub>XML: Resurrecting T<sub>E</sub>X in the XML world
    - transforming XML input syntax through T<sub>E</sub>X to PDF
- 12 *Peter Wilson* / Glistings
  - stringing along; loops
- 100 *Boris Veytsman and Leila Akhmadeeva* / Drawing medical pedigree trees with T<sub>E</sub>X and PSTricks
  - typesetting medical pedigrees graphically

**Advanced**

- 4 *Donald Knuth* / T<sub>E</sub>X's infinite glue is projective
  - note about negative infinite glue
- 80 *Bob Neveln and Bob Alps* / Writing and checking complete proofs in T<sub>E</sub>X
  - verifying formal mathematical proofs with T<sub>E</sub>X and Python

**Reports and notices**

- 11 *Barbara Beeton and Idris Hamid* / Oriental T<sub>E</sub>X: A new direction in scholarly complex-script typesetting
- 26 *Practical T<sub>E</sub>X 2006 conference information*
- 137 *Abstracts* (Adams, Garcia, Höppner, Hummel, Moye, Peter, Wetmore)
- 138 *Calendar*
- 139 *TUG 2007 announcement*
- 140 *EuroBachOT<sub>E</sub>X 2007 announcement*
- 141 *Steve Peter* / TUG 2007 election report
- 145 *David Walden* / Financial statements for 2006
- 146 *Institutional members*
- 147 *TUG membership form*
- 148 *T<sub>E</sub>X consulting and production services*

# TUGboat

Volume 28, Number 1 / 2007  
Practical T<sub>E</sub>X 2006 Conference Proceedings

<b>General Delivery</b>	3	Karl Berry / <i>From the president</i>
	4	Barbara Beeton / <i>Editorial comments</i> Erratum: <i>TUGboat</i> 27:1 (EuroT <sub>E</sub> X proceedings); A new Korean T <sub>E</sub> X Society; L <sup>A</sup> T <sub>E</sub> X goes to the movies; Some <i>TUGboat</i> staff changes
<b>Warnings</b>	4	Donald E. Knuth / <i>T<sub>E</sub>X's infinite glue is projective</i>
<b>Software &amp; Tools</b>	5	Oleg Parashchenko / <i>T<sub>E</sub>XML: Resurrecting T<sub>E</sub>X in the XML world</i>
	11	Barbara Beeton and Idris Hamid / <i>Oriental T<sub>E</sub>X: A new direction in scholarly complex-script typesetting</i>
<b>Hints &amp; Tricks</b>	12	Peter Wilson / <i>Glistenings: stringing along; loops</i>
	15	Mark LaPlante / <i>The treasure chest</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	20	Ignacio Llopis Tortosa and María José Castro Bleda / <i>paperT<sub>E</sub>X: Creating newspapers using L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub></i>
	24	L <sup>A</sup> T <sub>E</sub> X Project Team / <i>L<sup>A</sup>T<sub>E</sub>X news, issue 17</i>
<b>Practical T<sub>E</sub>X 2006</b>	26	Conference program, delegates, and sponsors
<b>Keynote</b>	29	Barbara Beeton / <i>How to create a T<sub>E</sub>X journal: A personal journey</i>
<b>Publishing</b>	49	David Walden / <i>A lifetime as an amateur compositor</i>
	61	Elizabeth Dearborn / <i>T<sub>E</sub>X and medicine</i>
<b>Teaching &amp; Training</b>	65	Jon Breitenbucher / <i>L<sup>A</sup>T<sub>E</sub>X at a liberal arts college</i>
	70	Boris Veytsman / <i>Design of presentations: Notes on principles and L<sup>A</sup>T<sub>E</sub>X implementation</i>
<b>Software &amp; Tools</b>	77	Boris Veytsman and Maria Shmilevich / <i>Automatic report generation with Web, T<sub>E</sub>X and SQL</i>
	80	Bob Neveln and Bob Alps / <i>Writing and checking complete proofs in T<sub>E</sub>X</i>
<b>Graphics</b>	84	Troy Henderson / <i>A beginner's guide to MetaPost for creating high-quality graphics</i>
	91	Andrew Mertz and William Slough / <i>Graphics with PGF and TikZ</i>
	100	Boris Veytsman and Leila Akhmadeeva / <i>Drawing medical pedigree trees with T<sub>E</sub>X and PSTricks</i>
<b>Tutorials</b>	110	Peter Flynn / <i>Rolling your own Document Class: Using L<sup>A</sup>T<sub>E</sub>X to keep away from the Dark Side</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	124	Jim Hefferon / <i>L<sup>A</sup>T<sub>E</sub>X resources</i>
	126	Peter Flom / <i>L<sup>A</sup>T<sub>E</sub>X for academics and researchers who (think they) don't need it</i>
	129	Federico Garcia / <i>Hypertext capabilities with pdf L<sup>A</sup>T<sub>E</sub>X</i>
	133	Kaveh Bazargan and CV Radhakrishnan / <i>Removing vertical stretch — mimicking traditional typesetting with T<sub>E</sub>X</i>
<b>Abstracts</b>	137	Abstracts (Adams, Garcia, Höppner, Hummel, Moye, Peter, Wetmore)
<b>News</b>	138	Calendar
	139	TUG 2007 announcement
	140	EuroBachoT <sub>E</sub> X 2007 announcement
<b>TUG Business</b>	141	Steve Peter / <i>TUG 2007 election report</i>
	145	David Walden / <i>Financial statements for 2006</i>
	146	Institutional members
	147	TUG membership form
<b>Advertisements</b>	148	T <sub>E</sub> X consulting and production services

**Table of Contents** (ordered by difficulty)**Introductory**

- 3 *Barbara Beeton* / Editorial comments
  - typography and *TUGboat* news
- 29 *Barbara Beeton* / How to create a T<sub>E</sub>X Journal: A personal journey
  - a look back at *TUGboat* experiences since its founding
- 3 *Karl Berry* / From the President
  - some TUG activities and information for 2006
- 65 *Jon Breitenbucher* / L<sup>A</sup>T<sub>E</sub>X at a liberal arts college
  - experiences introducing L<sup>A</sup>T<sub>E</sub>X in an undergraduate liberal arts setting
- 61 *Elizabeth Dearborn* / T<sub>E</sub>X and medicine
  - experiences in self-publishing a medical transcription dictionary with T<sub>E</sub>X
- 126 *Peter Flom* / L<sup>A</sup>T<sub>E</sub>X for academics and researchers who (think they) don't need it
  - advocating L<sup>A</sup>T<sub>E</sub>X for scientists and dispelling myths
- 124 *Jim Hefferon* / L<sup>A</sup>T<sub>E</sub>X resources
  - a sampling of useful documentation, web sites, programs, and packages
- 84 *Troy Henderson* / A beginner's guide to MetaPost for creating high-quality graphics
  - introduction to MetaPost
- 24 *L<sup>A</sup>T<sub>E</sub>X Project Team* / L<sup>A</sup>T<sub>E</sub>X news, issue 17
  - latest L<sup>A</sup>T<sub>E</sub>X release notes: font encodings, graphics drivers, more
- 70 *Boris Veytsman* / Design of presentations: Notes on principles and L<sup>A</sup>T<sub>E</sub>X implementation
  - writing effective presentations and implementing them in L<sup>A</sup>T<sub>E</sub>X
- 49 *David Walden* / A lifetime as an amateur compositor
  - productivity with L<sup>A</sup>T<sub>E</sub>X and typesetting experiences along the way

**Intermediate**

- 133 *Kaveh Bazargan* / Removing vertical stretch — mimicking traditional typesetting with T<sub>E</sub>X
  - a method for typesetting on a grid, including double columns and math
- 129 *Federico Garcia* / Hypertext capabilities with pdfL<sup>A</sup>T<sub>E</sub>X
  - introduction to the hyperref package, links, bookmarks, and more
- 15 *Mark LaPlante* / The treasure chest
  - selected new CTAN packages in 2006
- 91 *Andrew Mertz and William Slough* / Graphics with PGF and TikZ
  - graduated examples of graphics within L<sup>A</sup>T<sub>E</sub>X using a MetaPost-like syntax
- 20 *Ignacio Llopis Tortosa and María José Castro Bleda* / paperT<sub>E</sub>X: Creating newspapers using L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>
  - automatically generating a PDF newspaper from external sources
- 77 *Boris Veytsman and Maria Shmylevich* / Automatic report generation with Web, T<sub>E</sub>X and SQL
  - creating high-quality project reports while minimizing overhead

**Intermediate Plus**

- 110 *Peter Flynn* / Rolling your own Document Class: Using L<sup>A</sup>T<sub>E</sub>X to keep away from the Dark Side
  - 5 *Oleg Parashchenko* / T<sub>E</sub>XML: Resurrecting T<sub>E</sub>X in the XML world
    - transforming XML input syntax through T<sub>E</sub>X to PDF
- 12 *Peter Wilson* / Glistings
  - stringing along; loops
- 100 *Boris Veytsman and Leila Akhmadeeva* / Drawing medical pedigree trees with T<sub>E</sub>X and PSTricks
  - typesetting medical pedigrees graphically

**Advanced**

- 4 *Donald Knuth* / T<sub>E</sub>X's infinite glue is projective
  - note about negative infinite glue
- 80 *Bob Neveln and Bob Alps* / Writing and checking complete proofs in T<sub>E</sub>X
  - verifying formal mathematical proofs with T<sub>E</sub>X and Python

**Reports and notices**

- 11 *Barbara Beeton and Idris Hamid* / Oriental T<sub>E</sub>X: A new direction in scholarly complex-script typesetting
- 26 *Practical T<sub>E</sub>X 2006 conference information*
- 137 *Abstracts* (Adams, Garcia, Höppner, Hummel, Moye, Peter, Wetmore)
- 138 *Calendar*
- 139 *TUG 2007 announcement*
- 140 *EuroBachOT<sub>E</sub>X 2007 announcement*
- 141 *Steve Peter* / TUG 2007 election report
- 145 *David Walden* / Financial statements for 2006
- 146 *Institutional members*
- 147 *TUG membership form*
- 148 *T<sub>E</sub>X consulting and production services*