TUGboat

Volume 22, Number 1/2 / March/June 2001

TEX Users Group

Memberships and Subscriptions
TUGboat (ISSN 0896-3207) is published quarterly by the TEX Users Group, 1466 NW Naito Parkway, Suite 3141, Portland, OR 97209-2820, U.S.A.

2001 dues for individual members are as follows:

- Ordinary members: $75.
- Students: $45.

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site: `http://www.tug.org`.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: $85 a year, including air mail delivery.

Periodical-class postage paid at Portland, OR, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TEX Users Group, 1466 NW Naito Parkway, Suite 3141, Portland, OR 97209-2820, U.S.A.

Institutional Membership
Institutional Membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group. For further information, contact the TUG office (`office@tug.org`).

Addresses
General correspondence, payments, etc.
TEX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Delivery services, parcels, visitors
TEX Users Group
1466 NW Naito Parkway
Suite 3141
Portland, OR 97209-2820
U.S.A.

Telephone
+1 503 223-9994

Fax
+1 503 223-3960

Electronic Mail
(Internet)

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

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

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

To contact the Board of Directors:
`board@tug.org`

World Wide Web
`http://www.tug.org/`
`http://www.tug.org/TUGboat/`

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

[T]hose who know only what words are *for* can hardly know what words *are*. I cannot find it within me to see them only as manipulable counters, though they are that; they seem, quite often, a parade of gorgeous animals muttering by, a caravan slouching off to Gutenberg or some equally imaginary place.

Paul West
The Secret Lives of Words (2000)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
EDITOR BARBARA BEETON

VOLUME 22, NUMBER 1/2 • MARCH/JUNE 2000
PORTLAND • OREGON • U.S.A.

TUGboat

For the remainder of the 2001 volume year, two issues will appear. The September 2001 issue (Vol. 22, No. 3) will contain the Proceedings of the 2001 TUG Annual Meeting. During 2002, the communications of the TeX Users Group will be published as one double issue (1/2) and two regular issues. The September issue (Vol. 23, No. 3) is expected to contain the Proceedings of the 2002 TUG Annual Meeting.

We are unfortunately not able to set a definitive schedule for the appearance of the next few issues. but are making every effort to be back on a reasonably normal schedule by the end of 2002.

TUGboat is distributed as a benefit of membership to all members.

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

Owing to the lateness of the present issue, and the scarcity of material submitted for future issues, suggestions will be accepted and processed as received.

Manuscripts should be submitted to a member of the *TUGboat* Editorial Board. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor-in-Chief, Barbara Beeton, to the Managing Editor, Robin Laakso, or to the Production Manager, Mimi Burbank (see addresses on p. 3).

The *TUGboat* "style files", for use with either `plain` TeX or LaTeX, are available from CTAN. For authors who have no network access (browser or FTP), they will be sent on request; please specify which is preferred. Send e-mail to `TUGboat@tug.org`, or write or call the TUG office.

This is also the preferred address for submitting contributions via electronic mail.

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to `TUGboat@tug.org` or to the Editor, Barbara Beeton (see address on p. 3).

Other TUG Publications

TUG is interested in considering additional manuscripts for publication. These might include manuals, instructional materials, documentation, or works on any other topic that might be useful to the TeX community in general. Provision can be made for including macro packages or software in computer-readable form. If you have any such items or know of any that you would like considered for publication, send the information to the attention of the Publications Committee at `tug-pub@tug.org` or in care of the TUG office.

TUGboat Advertising and Mailing Lists

For information about advertising rates, publication schedules or the purchase of TUG mailing lists, write or call the TUG office.

Trademarks

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

METAFONT is a trademark of Addison-Wesley Inc.
PostScript is a trademark of Adobe Systems, Inc.
TeX and AMS-TeX are trademarks of the American Mathematical Society.
UNIX is a registered trademark of X/Open Co. Ltd.

Addresses

TEX Users Group Office
Robin Laakso
1466 NW Naito Parkway, Suite 3141
Portland, OR 97209-2820 U.S.A.
+1 503 223-9994
Fax: +1 503-223-3960
office@tug.org

William Adams
75 Utley Drive, Ste. 110
Mechanicsburg, PA 17055
U.S.A.
willadams@aol.com

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 U.S.A.
+1 401 455-4014
bnb@ams.org, tugboat@tug.org

Karl Berry
685 Larry Ave. N
Keizer, OR 97303 U.S.A.
karl@tug.org

Victor Bos
Software Construction Laboratory
Turku Centre for Computer Science
Lemminkäisenkatu 14 A
FIN-20520, Turku
Finland
v.bos@abo.fi

Mimi R. Burbank
CSIT, 408 Dirac Science Library
Florida State University
Tallahassee, FL 32306-4130 U.S.A.
+1 850 644-2440
mimi@csit.fsu.edu

Kaja Christiansen
Dept. of Computer Science
Arhus Univ., Ny Munkegade
Bldg 540
DK-8000 Aarhus C, Denmark
kaja@daimi.aau.dk

Donald DeLand
Integre Technical Publishing Co.
4015 Carlisle NE, Suite A
Albuquerque, NM 87107 U.S.A.
don.deland@tug.org

Susan DeMeritt
IDA/CCR La Jolla
4320 Westerra Court
San Diego, CA 92121 U.S.A.
+1 619 622-5455
sue@ccrwest.org

Victor Eijkhout
Computer Science Department
111 Ayres Hall
University of Tennessee
Knoxville, TN 37996-1301 U.S.A.
victor@eijkhout.net

Robin Fairbairns
32 Lilac Court
Cherryhinton Rd.
Cambridge, CB1 4AY, U.K.
Robin.Fairbairns@cl.cam.ac.uk

Peter Flynn
Computer Centre
University College
Cork, Ireland
+353 21 902609
pf@ucc.ie

George Grätzer
Department of Mathematics
University of Manitoba
Winnipeg MN, R3T 2N2
Canada
gratzer@cc.umanitoba.ca

Hans Hagen
Pragma ADE
Ridderstraat 27
8061 GH Hasselt, The Netherlands
pragma@wxs.nl
http://www.pragma-ade.nl

Jim Hefferon
Department of Mathematics
Saint Michael's College
Colchester, VT 05439, U.S.A.
tex@joshua.smcvt.edu
http://joshua.smcvt.edu/hefferon.html

Lars Hellström
Matematiska institutionen
Umeå universitet
S-901 87 Umeå
Sweden
Lars.Hellstrom@math.umu.se

John D. Hobby
Bell Laboratories
Room 2C-458
700 Mountain Ave.
Murray Hill, NJ 07974-0636
hobby@research.bell-labs.com

Alan Hoenig
17 Bay Avenue
Huntington, NY 11743 U.S.A.
+1 516 385-0736
ajhjj@cunyvm.cuny.edu
ahoenig@suffolk.lib.ny.us

Stephanie Hogue
AlphaSimplex Group
One Cambridge Center
9th Floor
Cambridge, MA 01242 U.S.A.
shogue@typewright.com

Laura Elizabeth Jackson
Raleigh, North Carolina U.S.A.
lejacks2@unity.ncsu.edu
http://www.educat.hu-berlin.de/~voss/

Mimi Jett
Institute for Advanced Learning
IBM Research
(use TUG Office address)
+1 503 578-2366
jett@us.ibm.com

Judy Johnson
jannejohnson@yahoo.com

Donald E. Knuth
Department of Computer Science
Stanford University
Stanford, CA 94305 U.S.A.

Adam H/ Lewenberg
211 Paddock Drive East
Savoy, Illinois 61874
U.S.A.
adam@macrotex.net

Sjouke Mauw
Computing Science Department
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB, Eindhoven
The Netherlands
sjouke@win.tue.nl

Wendy McKay
Control and Dynamical Systems
107-81
California Institute of Technology
Pasadena, CA 91125, U.S.A.
wgm@cds.caltech.edu

Frank Mittelbach
LATEX3 Project Team
latex-l@relay.urz.uni-heidelberg.de

Patricia Monohon
University of California San Francisco
Dill Research Lab
3333 California Street, #415
San Francisco, CA 94118 U.S.A.
+1 415 502-2839
pmonohon@zimm.ucsf.edu

Ross Moore
Macquarie University
NSW 2109, Australia
ross@maths.mq.edu.au

Arthur Ogawa
40453 Cherokee Oaks Drive
Three Rivers, CA 93271 U.S.A.
+1 209 561-4585; Fax: +1 209 561-4584
`ogawa@teleport.com`

Cheryl Ponchin
Center for Communications Research
Institute for Defense Analyses
29 Thanet Road
Princeton NJ 08540-3699
`cheryl@ccr-p.ida.org`

Roy Preston
4 Avon Wharf
Bridge Street
Christchurch
Dorset BH23 1DY
England UK
`preston@lds.co.uk`
`http://www.lds.co.uk/preston/`

Denis Roegel
LORIA
Campus scientifique
BP 239
54506 Vandœuvre-lès-Nancy cedex
France
`roegel@loria.fr`
`http://www.loria.fr/~roegel/`

Kristoffer Høgsbro Rose
IBM
T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532 U.S.A.
`krisrose@us.ibm.com`

Michael Sofka
C&CT, VCC 309
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180-3590 U.S.A.
`sofkam@rpi.edu`

Philip Taylor
The Computer Centre,
Royal Holloway and Bedford
New College,
University of London,
Egham Hill
Egham, Surrey TW20 0EX, U.K.
`P.Taylor@vax.rhbnc.ac.uk`

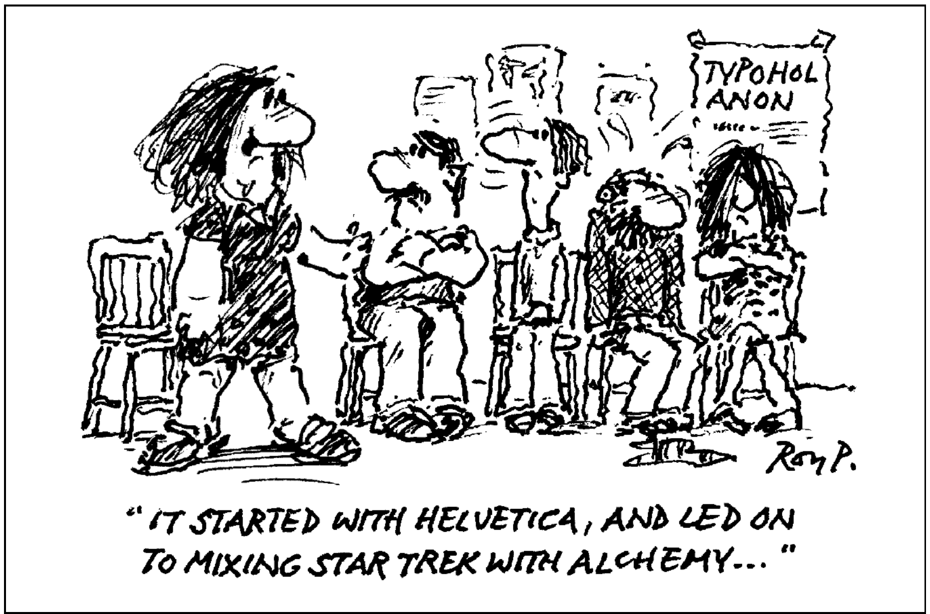
Christina Thiele
15 Wiltshire Circle
Nepean K2J 4K9, Ontario Canada
`cthiele@ccs.carleton.ca`

Herbert Voß
Berlin, Germany
`voss@lyx.org`

Hermann Zapf
Seitersweg 35
D-64287 Darmstadt, Germany

Cartoon

by Roy Preston


TYPOHOL ANON
Roy P.
"IT STARTED WITH HELVETICA, AND LED ON TO MIXING STAR TREK WITH ALCHEMY..."

General Delivery

From the President

Mimi Jett

Greetings TUG Members,

This issue represents some major improvements in the way we create and produce *TUGboat*. Starting with the organization of the TUG Editorial Board, formed at the Annual Meeting in Newark, Delaware, last August, and creating the position of Managing Editor, we are anticipating a higher level of participation from a greater pool of volunteers.

For many years, the entire set of tasks needed to create and produce one issue of the journal has been handled by a small set of dedicated members: Barbara Beeton, Mimi Burbank, Christina Thiele, Sebastian Rahtz and Robin Fairbairns. I know many others have helped, but these five people have really given life to *TUGboat*. Thank you all, for many years of service and devotion to the community.

Our objective in creating the Editorial Board and Managing Editor position was to copy the model of most professional scholarly publishers: a managing editor is responsible to recruit content and deliver content to the Editor-in-Chief (EIC) for vetting and to the production manager for formatting, and printing/mailing. The managing editor also keeps the production schedule and coordinates the activities of the editorial board. TUG is lucky to have Robin Laakso, our excellent Office Manager, also skilled in the organizational and communication requirements of a Managing Editor. Robin has embraced the challenge. We are very pleased to report that this double issue is the result of our new strategy. The goal is to continue to involve more people, maintain a high quality publication and get *TUGboat* back on schedule. We plan at least one other double issue in 2002.

Involving more people with *TUGboat* takes not only organizational tools, but also you, new and renewing members — people who have an ongoing interest in TeX and TeX-related research and products. You may be interested to know that the total number of TUG members has remained steady at approximately 2000 for the last 4-5 years. Member categories represented include individual, student, senior, subscriber, institutional and joint memberships worldwide. TUG members are located in 54, 58, 66, 65, and 59 (and counting) countries respectively, from 1998 to 2002. Each year new user groups are linked to the TUG website allowing people in all corners of the world to access TeX users locally, or in nearby countries.

The relative stability of membership numbers is a welcome trend; however, we think we can do better. Another forward-moving decision made at the 2001 Annual Meeting in Delaware was to re-apply for the 501(C)(3) non-profit status. For those of you unfamiliar with US tax codes or if you were not a member of TUG 15 years ago when we first applied for the 501(C)(3) non-profit status, let me explain. After TUG, the association, ceased to operate under the umbrella of the American Mathematical Society in 1987, the organization incorporated. With the corporate filing came a set of by-laws and decisions to be made about taxation. Hence TUG applied for the 501(C)(3) non-profit status. Unfortunately, after a great legal effort, TUG was denied the request and given a 501(C)(6) "trade association" non-profit status instead. Fifteen years ago we were living in a world much different than today. Indeed, the concept of "open source" used in the production of "computerized typesetting software" was very difficult to understand let alone compare to other not-for-profit organizations operating at that time.

The importance of the 501(C)(3) status cannot be underestimated. The (C)(3) status offers US members a tax deduction for their membership (minus the cost of materials) and contributions. For US members this means dues and contributions are fully tax-deductible to the amount allowed by law. It also means a potential increase in membership as many institutions require the (C)(3) status to join a not-for-profit membership-based organization. The (C)(3) status would make TUG eligible for a low cost non-profit mailing permit. Currently, under the (C)(6) status, TUG is not eligible for the lowest bulk mailing rates. The TUG Board of Directors believes that TUG qualifies for the (C)(3) status; that is why we recently hired an attorney to file a new application. If awarded (the IRS review can take 6–12 months), TeX User Groups worldwide will potentially benefit from grant monies, equipment donations, and the lowest shipping costs available from the US.

The TUG Board of Directors would like to know about TeX-related projects that you or someone you know is working on — and encourage you to submit an article about those projects to *TUGboat*. We cannot, after all, recognize the efforts of those in our community if we are unaware of your work!

As always, we welcome your comments and thank you for your support.

◇ Mimi Jett
IBM Research
Institute for Advanced Learning
`jett@us.ibm.com`

Editorial Comments

Barbara Beeton

We're late ...

This issue, nominally dated March/June 2001, is a year late. There are many causes, but one of the principal reasons is the difficulty of obtaining material and bringing it together to make up the issue.

There has for a long time been a decrease in the number of articles submitted for publication. Requests for new material, as for volunteers, have gone for the most part unanswered. It is hard not to reach the conclusion that most people using TEX in their daily work feel that what they are doing isn't "news", or that their experience wouldn't really be of interest to other TEXies.

Another factor in our delay is the increase of the duties associated with the "real jobs" of the members of the production team, to the point where they find it difficult to find time to beat the bushes for new material, or even to review the publications of other TEX groups for material they think would be interesting to the wider TUG audience.

The decision to designate this as a double issue was a difficult one, especially without the quantity of material that designation would ordinarily imply. The overriding consideration was our desire to try to get the publication back on schedule.

We've been considering some alternatives to the present format, to adjust to a smaller amount of material. One possibility is a smaller trim size, perhaps 6×9 or 7×10 inches, both common sizes for scholarly journals. However, this could preclude publication of items such as Hàn Thế Thành's dissertation (**21**:4) or the proposed new LATEX algorithm for handling floats (**21**:3, pages 278-290); these require the large page size. It would also be very difficult to prepare articles based on the LATEX `doc` format, which depends on a wide left margin to display macro and environment names. Another possibility is to reduce the number of issues from four to three per year; however, this would result in the loss of our periodical mailing permit, which requires four issues per year. The jury is still out; your opinions are welcome — send them to us at `TUGboat@tug.org`.

As mentioned by Mimi Jett in her president's message, a new position has been created, that of Managing Editor. This function has been added to the duties of TUG's office manager, Robin Laakso. Robin has been diligent in trying to create a schedule we can meet, and in her nagging of the editor and production team, and she is eager to receive new material from prospective authors that she can add to the production queue.

We hope that you will be understanding of our predicament, and that you will make suggestions — and submit articles — to help bring *TUGboat* back to its former standing.

CTAN and "The Treasure Chest"

In 1998, Christina Thiele was inspired to create "The Treasure Chest", a regular column summarizing the activity on CTAN — new postings, updates to existing material, and sometimes the adoption of an orphaned package by another maintainer.

This issue marks the transition of "The Treasure Chest" from Christina's tender care to the hands of William Adams, a longtime TEX user.

I would like to take this opportunity to thank Christina, who has worn so many hats on TUG's behalf, not only for her work on this and other *TUGboat* efforts, but also for her friendship and good will over many years. We haven't heard the last from Christina (she remains on the *TUGboat* production team).

To prepare for the switchover, I compiled a collection of CTAN announcements for William. Looking back on these messages, it struck me how many people have dedicated themselves to the care and feeding of this fine archive over the years. This seems a fine time to express thanks to them as well: Rainer Schöpf and Reinhard Zierke in Germany, Robin Fairbairns in the U.K., and Jim Hefferon in the U.S. are the current maintenance crew. Great job, guys, well done!

`ctan-ann`, the mailing list for CTAN announcements, has moved to the Dante server; if you want to subscribe, send a message to

```
majordomo@dante.de
```

containing one line:

```
subscribe ctan-ann
```

TEX Mexico User Group

A new TEX group was announced near the end of 2000: TEX México — El Grupo de Usuarios TEX México.

The TEX México User Group is committed to promote the use of the digital typesetting system TEX and METAFONT created by Donald Knuth, adapt the TEX family to the development and enhancement of new tools to help writing and typesetting antique texts of New Spain, and the use of the system for glyph creation of ancient mesoamerican languages for their use, study and preservation in the digital era.

Visit the web site at `http://ciencia.dcc.umich.mx/tex/`.

Goodbye to Father Larguier

Father Everett Larguier, s.j., possibly the oldest member of TUG, died peacefully on September 22, 2000, in Mobile, Alabama. According to a brief article in *TUGboat* **20**:2 (pages 89–90), he was born in 1910, and started using LaTeX in the mid-1980s as a way "to keep old man Alzheimer from my door", and to work on a book on topology. He was still using LaTeX on a Linux box up to August 2000, writing books.

Mimi Burbank corresponded with Fr. Larguier for several years, and provided his TeX support. She reports that she will miss his gentle and cheerful exchanges.

Some places to learn more about books and printing

Rare Book School at the University of Virginia is an independent institute supporting the study of the history of books and printing and related subjects. Week-long courses are offered throughout the year, as are public lectures. The Book Arts Press is the publishing arm of Rare Book School. The catalog contains a number of publications about book and manuscript history, bookselling, and bibliography, including many issues of the last six years of *Fine Print*. Of special interest are several videotapes about type and books. The prices are quite reasonable; many of these items, if available elsewhere, are priced much higher.

Curiously, the Rare Book School mascot, like that of TeX, is a lion. He is seen throughout the RBS web site engaged in various activities. Visit the site if only to make his acquaintance:
`http://www.virginia.edu/oldbooks`

The San Francisco Center for the Book is dedicated to exploring the arts of the book and the visible word. A busy schedule of letterpress and other classes is posted on their web site; you don't even have to live in the San Francisco area — some classes last for just one day or a weekend. The web site also lists their ongoing exhibition program. Visit `http://www.sfcb.org/` to see what's available.

5000 years of the written word

An item from the BBC News announced an international conference hosted by Iraq "to mark the 5000th anniversary of the written word." According to the organizers, writing was born in the ancient city of Uruk, now in southern Iraq, in the third millennium BC.

Although a precise date and location are difficult to pin down at this distance, there is abundant archaeological evidence for believing that the culture of the Tigris-Euphrates delta "formed the basis for what is almost certainly the world's oldest literary culture."

The article can be viewed in its entirety at `http://news.bbc.co.uk/hi/english/world/middle_east/newsid_1230000/1230835.stm`.

The Gutenberg Bible online

Scanned images of several different copies of the Gutenberg Bible have been posted online. The availability of different copies makes it possible to compare both the printed text and the decorations, which were added later. Two copies, one printed on vellum and the other on paper, are in the collection of the British Library (`http://portico.bl.uk/`), and one copy, the original on vellum, is from the Göttingen State and University Library (`http://www.gutenbergdigital.de/gudi/eframes/`). Both sites contain other items of interest as well.

An article describing the British Library project can be found at `http://news.bbc.co.uk/hi/english/uk/newsid_1035000/1035014.stm`.

Xy-pic home moved to TUG

The home site for the Xy-pic package (by Kristoffer Rose and Ross Moore) has moved to the TUG web site. It can be found from a link on the page `http://tug.org/applications/`.

The page also points to TUG home pages for a growing number of other major packages.

Legibility study online

It's not always easy to find documented studies concerning the legibility or readability of fonts, so every online reference is useful.

An addition to this corpus is "Legibility and Readability of Small Print: Effects of Font, Observer Age and Spatial Vision" by G. Kevin Connolly. This paper is a thesis submitted in 1998 toward an M.S. degree from the Department of Psychology of the University of Calgary, Alberta, Canada: `http://www.psych.ucalgary.ca/PACE/VA-Lab/gkconnol/Thesis.html`

In addition to statistical analysis of the study results, the paper contains an extensive bibliography; unfortunately, none of the references have online links.

◇ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 USA
bnb@ams.org

Why TEX?

Jim Hefferon
St Michael's College
`http://joshua.smcvt.edu/hefferon.html`

From time to time someone may ask you for a list of TEX's strengths. They may want to explain to an administrator why to install it, they may have been advised to use it, or they may simply have found it on their system and want to know what it does. Starting on the next page is a list that you can give to them (it starts on a fresh page to make neater copies). It is understandable to anyone with experience in computing systems or programming.

This article grew out of a Usenet discussion[1] started by Fabrice Popineau. I'd like to thank the contributers to that discussion, who are too numerous to name singly. I would like also to thank Lynne Hefferon and Peter Flynn for their help.

[1] Thread: *10 best reasons to use TeX ?* on `comp.text.tex`, Nov 2001.

Why TEX?

TEX is a system for computer typesetting — for placing text on a page. It is well known in the area of typesetting mathematics and other technical material.

But 'well known' is a relative term. Most computer users who are not scientists do not know TEX. This document is for you if you have heard a little and want an overview of its strengths.

1 Introduction

You no doubt already use other tools so we can start by comparing TEX to the two most common ways of placing text on a computer.

1.1 Compared to word processors

Most people arrange material on a page with a word processor.

Word processors are easy to begin with. To get a blank line between two paragraphs you enter it in. To make a reference to the bibliography you type it into the text in the style that you need. Seems simple. You know what you want and you just do it.

But as the document gets to be a bigger and tougher job, laying it out yourself becomes a problem. Seeing that there is the same amount of vertical space between all of the paragraphs in a twenty-page article is error-prone work. So is making sure that all of the bibliographic entries follow the requirements. And, very few authors have the knowledge and aesthetic eye to correctly place and size symbols in an equation.

In contrast, TEX authors find it easy to have systematic formatting, even when they have complicating elements such as mathematics or tables.

So TEX is like a word processor in that both put text on a page. But it is different in many ways, one of which is that it automates much of the job.

1.2 Compared to HTML

In HTML you might emphasize a point with italics by typing '`it's <i>hot</i> here`' and only later, as the material is run through a browser, is it actually formatted. TEX works in the same way: you might type '`it's \emph{hot} here`' and later run that file through the program to do the typesetting.

Thus TEX is like HTML in that the step of typing the material is separate from the step of setting the material. Unlike HTML, however, TEX can be used as a programming language. You can branch with '`if`' constructs, use variables, etc. So, while becoming a TEX expert takes longer than becoming an HTML expert, the gain is that TEX gives you the power to do more.

This power has allowed people in the TEX community to accomplish great things with it and, even if you never become a wizard yourself, you can use their wizardry in your work. Browse around the Comprehensive TEX Archive Network[2] and you will see that there are solutions available, usually freely available, for a very wide range of needs.

1.3 TEXing

An installation of TEX has many components. But for a first impression, the figure on the next page gives the core flow of information. The material that you have entered in `filename.tex` is processed (technically, 'expanded by the macro package') into a form that is understandable to the main TEX engine, which performs the typesetting. Then you convert the result to the output format that you want. We'll discuss parts of this flow in the next section.

2 Ten Reasons

These are the reasons most often cited for using TEX, grouped into four areas: Output Quality, Superior Engineering, Freedom, and Popularity.

Output Quality

You write documents to be read and understood. Your first concern should be: how good is the output? Is it as readable and as useful as possible? Is it, even, beautiful?

(*i*) TEX has the best output What you end with, the symbols on the page, is as usable — and beautiful — as a non-professional can produce.

This especially holds for complex documents, such as ones with mathematics; see Appendix A. It also holds for documents that are complex in other ways: with many tables, or many cross references or hyper-links, or just with many pages.

The usual way of working with a word processor, clicking the material in, is not suited to complex documents so we cannot fairly compare such output with TEX. However, even on simple documents TEX has advantages. Compare the two samples in Appendix B. These are short and the typographic differences are subtle but even a non-expert may see that the TEX page looks "more right". The word processor's page has some lines with wide gaps between words and some lines with too many words stuffed in (contrast the second paragraph's second line with its third). TEX's output is more readable.

[2] try `http://www.ctan.org/tex-archive/macros/latex/contrib/supported`

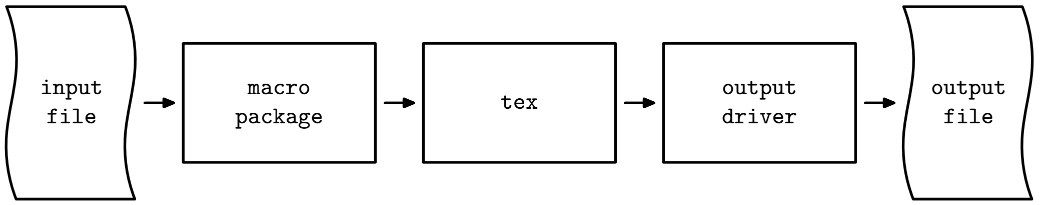


Figure 1: The basic information flow of TeX

(*ii*) **TeX knows typesetting** Expertise is coded into TeX.

Appendix B is an example. TeX's more even lines are a consequence of its more sophisticated algorithms for making paragraphs and for hyphenating.

Another way that this expertise gives better output comes in setting technical material. TeX moves the task, as much as possible, into the software. For instance, it automatically classifies each mathematical symbol as a variable, or a relation, etc., and sets them with appropriate amounts of surrounding space. It also sizes superscripts, and many other things. The result is that, because your document follows the conventions of professional typesetting, your readers will know exactly what you mean. You almost never have to fret with the formulas. They just come out right.

The quality of output that you get is the single best reason to use TeX.

Superior Engineering

Everyone has been frustrated with software that is slow, fat, buggy, or that undergoes frequent incompatible version changes. TeX will not give you those troubles; from a Computer Science standpoint, TeX is very impressive.

(*iii*) **TeX is fast** TeX was written by D Knuth, one of the world's leading experts in the design of algorithms. It ran quickly when it was developed in 1978 and so on today's machines it is very fast. It is easy on your computer's memory and disk space, too.

(*iv*) **TeX is stable** TeX is in wide use, with a long history. It has been tested by millions of users on demanding input. It will never eat your document. Never.

But there is more here than just that the program is reliable. TeX's designer has frozen the central engine, the actual `tex` program. Documents that run today will still run in ten years, or fifty. So 'stable' means more than that it actually works; it means that it will continue to work, forever.

(*v*) **Stable but not rigid** A system locked into 1978's technology would today have gaps. That's why TeX is extendable, so that innovations can be added on — layered over the underlying engine.

An example is the LaTeX macro package, which is the most popular way to use TeX today. It is a front end to the engine, affecting the way authors input their work. It adds conveniences such as automatic cross references, indexing, a table of contents, automatic numbering of chapters, sections, theorems, etc., in a variety of styles, and a straightforward but powerful way to make tables.

LaTeX also adds a philosophy of encouraging authors to structure their document by meaning rather than by appearance. For instance, the way that most word processor users make a section heading is by typing the title, highlighting it with the mouse, clicking in a menu to select boldface, clicking in another menu for the point size, and then adding the white space above and below. LaTeX authors type `\section{`*title*`}`. This has two advantages. First, since it is a computer language command, it makes the type style, size, and vertical spacing uniform throughout your document. (True, working with a computer language makes changing the default trickier. But on the other hand, if you have put in two dozen or more section headings by hand then chances are that you've erred in at least one.) The second advantage of LaTeX's approach is that it is self documenting. You can, for instance, automate producing a list of sections. (Some word processors can do logical structuring, although few authors use it.)

And, LaTeX itself is extendable. There are thousands of "style files," which do everything from adapting the basics to the needs of the American Math Society,[3] to making cross-references into hyper-references,[4] all the way to allowing you to

[3] `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`

[4] `http://www.ctan.org/tex-archive/macros/latex/contrib/supported/hyperref/doc/manual.pdf`

add epigraphs,[5] the short quotations that sometimes decorate the start or end of a chapter.

Just because LaTeX is the most popular macro package doesn't mean that it is the best one for you. Many others are available; see the the TeX Users Group's interest page.[6]

So TeX has been, and is being, developed and extended in many ways.

(*vi*) The input is plain text TeX's source files are portable to any computing platform. They are also easy to produce automatically, for example when you want to write a report from material in a database. They are compact (all of the files for my 450-page textbook[7] and 125-page answer supplement fit easily on one floppy). And, they integrate with other tools such as search utilities.

Use of this type of input file stems from an overall mindset. TeX arose in the world of science and engineering where there is a tradition of cooperating closely with fellow workers. A binary input format, especially a proprietary one, is bad for cooperation: probably you have had to go through the trouble of upgrading because coworkers upgraded and you could no longer read their files. With TeX systems that experience is quite rare (the last time that there was a LaTeX release that lost some backward compatibility was in 1995).

There are even ways to run TeX directly from XML input, which many people think is the standard input format of the future. So, with the TeX formatting engine in the middle, the input front end may be adjusted to meet your needs, and changing times.

(*vii*) The output can be anything As with inputting, TeX's outputting step is separate from its typesetting. The TeX engine's results can be converted to a printer language such as PostScript, or to a web language such as PDF or HTML, or, probably, to whatever will appear in the future. And, the typesetting (line breaks, etc.) will be the same no matter where your output appears.

See also the example section below.

Freedom

Most computer users have heard about Free and Open Sourced software and know that, as with the GNU programs, Linux, Apache, Perl, etc., this style of development can yield software that is first class. TeX, along with associated materials such as index makers or style files, falls into this category.

(*viii*) TeX is free The source of the main `tex` engine is open. (In large part because of this it is thoroughly debugged. Knuth offers a reward for finding errors and no significant ones have been found in a very long time, despite many smart folks looking for them). All of the other main components are open, also.

(*ix*) TeX runs anywhere Whatever meets your needs — Windows, Macintosh, a variety of Unix, or almost any other system — you can get TeX, either freely distributed or in a commercial implementation.[8]

So although the core of TeX was written some time ago, it fits well with today's trends.

Popularity

Using the same system as many other people has advantages. You can get answers to your questions. Your problems might well have already been solved. And, because of this large user base, your system is sure to be around for years.

(*x*) TeX is the standard Most scientists, especially academic scientists, know TeX. As a result, many publishers of technical material are set up to work with it. In particular, TeX is the system preferred by the American Math Society.[9]

Because it is the standard, TeX's support by other technical software is the best. For example, there are editing modes to make input convenient, such as AUCTeX for Emacs. Another example is that most computer algebra systems, such as Maple and Mathematica, will give output in TeX. And no doubt technical software developed in the future will support TeX, also.

In addition, TeX is used by many people outside of the sciences, for all of the reasons given in this document. For instance, there is a way to produce beautiful critical edition texts.[10]

You wouldn't want to use a bad system simply because it is popular. TeX has earned its user base for sound reasons, some of them given above. Nonetheless, the existence of such a base is itself one reason to adopt a software package.

In summary, TeX was designed by one of the world's foremost computer scientists. That design makes it especially shine in areas where the system familiar to most computer users, word processors, falls short. Briefly, that is, it was designed well.

[5] `http://www.ctan.org/tex-archive/macros/latex/contrib/supported/epigraph`

[6] `http://www.tug.org/interest.html`

[7] `http://joshua.smcvt.edu/linalg.html`

[8] `http://www.tug.org/interest.html`

[9] `http://www.ams.org/tex`

[10] `http://www.ctan.org/tex-archive/macros/plain/contrib/edmac`

3 An Example

Anyone can see from the two appendices (and this document) that TEX's output quality is first-rate. However, some of the other points above might be less familiar. This section may help to make them more concrete. Someone asked on the discussion group `comp.text.tex`[11] whether TEX would be suitable for a large job where the text is generated from a database. Here is my reply.

```
> I'm contemplating using tex (or latex)
> to perform a mail merge. .. anywhere
> from 1,800 to 25,000 documents at a
> time .. What performance issues .. ?
I doubt TeX would slow you down. I just
wrote this Perl script
......................................
#!/usr/bin/perl
$file_text="\\documentclass{letter}
\\begin{document}
\\begin{letter}{Addressee \\\\ Address}
\\opening{Dear Sir} Hi. \\closing{Thanks}
\\end{letter}
\\end{document}";
for ($i=0; $i<100; $i++) {
    $fn="test".$i;
    open(OUTFILE,">$fn.tex");
    print OUTFILE $file_text;
    close OUTFILE;
    system("latex $fn");
    system("dvips -Pwww -o$fn.ps $fn");
    system("rm $fn.aux");
    system("rm $fn.log");
}
......................................
which writes 100 LaTeX letters, then
LaTeX's them, then converts to PostScript
for printing (and deletes a couple of
log files).  On my laptop (P3; I don't
know the MHz) execution took 22 secs.
So I think your bottleneck is more likely
to be your printing device.
```

This illustrates many of the points above. First, TEX's input is plain text, so I was able to generate it easily in a program. Second, TEX is fast, so I got the one hundred letters in no time. Third, I didn't have to ask the person what platform they were using because TEX runs anywhere. For that matter — fourth — I didn't have to ask what software vendor they had licenses with, because TEX is free.

That is, it illustrates that TEX is a practical professional tool. TEX helps solve problems.

[11] `http://groups.google.com/groups?q=comp.text.tex`

4 When Not to Use TEX

Despite my enthusiasm for TEX, my children write their school reports with a word processor. That's because TEX has a steeper learning curve and for their material the word processor is just fine.

A word processor suits your needs if your documents are brief, structurally simple, and entered by hand. If you will only ever write straightforward text, in short to medium-sized documents, and where good-enough typography is good enough, then stick with a word processor.

The opposite extreme, a document such as a brochure or an advertisement that is dominated by graphics, font changes, and color, is best tackled in a layout tool like Quark or Framemaker.

5 For More Information

The TEX Users Group[12] has much more information and many links, including more of an introduction[13] and a list of available distributions.[14] A good way to get started if you already have TEX installed is *The (Not So) Short Introduction to LATEX 2ε*.[15]

A A Sample of Mathematics

The first of the two following pages is an excerpt from *Theory of Recursive Functions and Effective Computability* by Rogers.

B A Sample of Plain Text

The other page has the first two paragraphs of *Zen in the Art of Archery* by Herrigel, on the left done in Microsoft's *Word* and on the right done in LATEX.

For each, I just selected the 12 point Times Roman font that came with my system and in other areas used the defaults, except that I made the line width be 3.5 inches. This is the layout of the edition of the book that sits on my shelf, and also lets you compare outputs here side by side.

Jim Hefferon
St Michael's College
2002-Mar-04

[12] `http://www.tug.org`
[13] `http://www.tug.org/whatis.html`
[14] `http://www.tug.org/interest.html`
[15] `http://www.ctan.org/tex-archive/info/lshort`

Recursive definitions are familiar in mathematics. For instance, the function f defined by

$$\begin{aligned} f(0) &= 1, \\ f(1) &= 1, \\ f(x+2) &= f(x+1) + f(x), \end{aligned}$$

gives the Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, ... (The study of *difference equations* concerns the problem of going from recursive definitions to algebraic definitions. The Fibonacci sequence is give by the algebraic definition

$$f(x) = \frac{\sqrt{5}}{5}\left(\frac{1+\sqrt{5}}{2}\right)^{x+1} - \frac{\sqrt{5}}{5}\left(\frac{1-\sqrt{5}}{2}\right)^{x+1}.)$$

The *primitive recursive functions* are an example of a broad and interesting class of functions that cam be obtained by such a formal characterization.

Definition The class of *primitive recursive functions* is the smallest class $\mathcal{C}$ (i.e., intersection of all classes $\mathcal{C}$) of functions such that

i. All *constant functions*, $\lambda x_1 x_2 \cdots x_k[m]$ are in $\mathcal{C}$, $1 \le k$, $0 \le m$;

ii. The *successor function*, $\lambda x[x+1]$, is in $\mathcal{C}$;

iii. All *identity functions*, $\lambda x_1 \cdots x_k[x_i]$ are in $\mathcal{C}$, $1 \le i \le k$;

iv. If f is a function of k variables in $\mathcal{C}$, and g_1, g_2, ... , g_k are (each) functions of m variables in $\mathcal{C}$, then the function $\lambda x_1 \cdots x_m[f(g_1(x_1, \ldots , x_m), \ldots , g_k(x_1, \ldots , x_m))]$ is in $\mathcal{C}$, $1 \le k, m$;

v. If h is a function of $k+1$ variables in $\mathcal{C}$, and g is a function of $k-1$ variables in $\mathcal{C}$, then the unique function f of k variables satisfying

$$\begin{aligned} f(0, x_2, \ldots , x_k) &= g(x_2, \ldots , x_k), \\ f(y+1, x_2, \ldots , x_k) &= h(y, f(y, x_2, \ldots , x_k), x_2, \ldots , x_k) \end{aligned}$$

is in $\mathcal{C}$, $1 \le k$. (For (v), "function of zero variables in $\mathcal{C}$" is taken to mean a fixed integer.)

At first sight it must seem intolerably degrading for Zen — however the reader may understand this word — to be associated with anything so mundane as archery. Even if he were willing to make a big concession, and to find archery distinguished as an "art," he would scarcely feel inclined to look behind this art for anything more than a decidedly sporting form of prowess. He therefore expects to be told something about the amazing feats of Japanese trick-artists who have the advantage of being able to rely on a time-honored and unbroken tradition in the use of bow and arrow. For in the Far East it is only a few generations since the old means of combat were replaced by modern weapons, and familiarity in the handling of them by no means fell into disuse, but went on propagating itself, and has since been cultivated in ever widening circles. Might one not expect, therefore, a description of the special ways in which archery is pursued today as a national sport in Japan?

Nothing could be more mistaken than this expectation. By archery in the traditional sense, which he esteems as an art and honors as a national heritage, the Japanese does not understand a sport but, strange as this may sound at first, a religious ritual. And consequently, by the "art" of archery he does not mean the ability of the sportsman, which can be controlled, more or less, by bodily exercises, but an ability whose origin is to be sought in spiritual exercises and whose aim consists in hitting a spiritual goal, so that fundamentally the marksman aims at himself and may even succeed in hitting himself.

At first sight it must seem intolerably degrading for Zen — however the reader may understand this word — to be associated with anything so mundane as archery. Even if he were willing to make the big concession, and to find archery distinguished as an "art," he would scarcely feel inclined to look behind this art for anything more than a decidedly sporting form of prowess. He therefore expects to be told something about the amazing feats of Japanese trick-artists who have the advantage of being able to rely on a time-honored and unbroken tradition in the use of bow and arrow. For in the Far East it is only a few generations since the old means of combat were replaced by modern methods, and familiarity in the handling of them by no means fell into disuse, but went on propagating itself, and has since been cultivated in ever widening circles. Might one not expect, therefore, a description of the special ways in which archery is pursued today as a national sport in Japan?

Nothing could be more mistaken than this expectation. By archery in the traditional sense, which he esteems as an art and honors as a national heritage, the Japanese does not understand a sport but, strange as this may sound at first, a religious ritual. And consequently, by the "art" of archery he does not mean the ability of the sportsman, which can be controlled, more or less, by bodily exercises, but an ability whose origin is to be sought in spiritual exercises and whose aim consists in hitting a spiritual goal, so that fundamentally the marksman aims at himself and may even succeed in hitting himself.

U.K. TUG, Oxford, Sunday, 12 September 1999 Question & Answer Session with Donald Knuth

Philip Taylor: I'm sure everybody here by now knows Don Knuth. Don has very kindly agreed to do a question and answer session this evening. In order to record both the questions and the answers, we have two microphones. One for Don's use and a radio microphone so we can capture your question on the tape, as well as the answer. Many thanks. Thanks Don.

Donald Knuth (DEK): O.K., I hope you can hear me.

Audience: Use the microphone.

DEK: No, I can't use the microphone, it doesn't work for me. It's only for the tape, but I will try to project. Since I have been retired for a few years and my voice isn't what it used to be, I'll do my best.

A part of the rules are that when you ask me a question, you give your name first. Another rule is that I get to ask questions too. Occasionally.

I did prepare one small thing, because, as coming to Oxford I re-read one of my favourite novels, a Dorothy L. Sayers mystery called *Gaudy Night*, which is all about Oxford. I wanted to read you just part of it, which has to do with typesetting. [*Laughter*] In Chapter 3 it mentions a Miss Lydgate, who had prepared her magnum opus, about — I don't know how to pronounce it — prosody, the study of meters in poetry. It said her handwriting was difficult to read, her experience in dealings with printers was limited but she had invented a novel and complicated system of notation that involved the use of twelve different varieties of type; and then she had all kinds of sheets in page proofs and so on. And she said 'Don't prick your fingers on that bit of manuscript that's pinned on, I'm afraid it's rather full of marginal balloons and interlineations, but I suddenly realised I could work out a big improvement in my notation, so I've had to alter it all the way through.' Then Harriet said comfortingly, 'Well, the Oxford University Press is no doubt accustomed to deciphering the manuscripts of scholars.' [*Laughter*] Now this work of Miss Lydgate appears to play a kind of minor role in the entire book and I have a question for you here, because it said her system of scansion required five alphabets with a series of pothooks for its expression, and this is a term I don't know. What are pothooks? If any of you can tell me . . .

Audience: Spell it out.

DEK: It's spelt like pot hooks, P-O-T-H-O-O-K-S.

Audience: Pothooks are hooks for hanging pots.

Male Voice: Yeah, but in typography it's an 'S' like an 'S'.

DEK: Is it some kind of special symbol? O.K. So she had these twelve fonts plus pothooks, which she used to typeset. So anyway you have some idea. Does this mike actually magnify? Can you hear me a little better? O.K. Well hello. I was reading from one of my favourite books. . . , but that's enough. I have one more thing prepared then I will go for questions. This was for Phil, I wanted to show you.

Laughter & Clapping: (Don shows he's wearing a T-shirt under his shirt that celebrates the Aston UK TEX Archive, 'oldest and best')

DEK: I always try to wear the appropriate T-shirt for the day. O.K., your turn now.

Dave Pawson: My name is Dave Pawson and I have an answer to your question. I was brought up in the 1940s and 50s in the north of England and one of the houses that I moved into was built in the mid 18th century, and around the coal burning stove and above it and beneath the mantelpiece there were a number of hooks and they were the 'pothooks' for hanging the pots on. [*The OED discusses this usage.*] Does that answer your question?

DEK: O.K. So this must be a similar shape that you would use for the symbols. I think that someone said that shorthand uses pothooks — some systems of shorthand. Thank you.

Dave Pawson: I think it's what you get when you type a left angle bracket into TEX and you are expecting a straight translation, and one of your assumptions in the big blue book comes unstuck, because it comes out like a pothook. [*Laughter*] Is that true?

DEK: Might be. The other thing I wanted to say is thank you very much to Phil Taylor for arranging that we could have the 10th anniversary celebration during the brief window of time that I could be in the U.K. I don't get to travel very often, and so now I get to remember not only the 10th anniversary celebration of TEX Users Group in America but also the one from here. At the 10th celebration in America we had the president of TUG dressed as

Thanks are due to the following:
Philip Taylor, for tape recording the Q&A session;
Jonathan Fine, for managing the transcription;
Pam George, transcribing from tape and making corrections;
Don Knuth, for diligently and promptly correcting proofs.

META the Lion — that was Bart Childs — and now here we have Mrs. TEX as well so now my delight is complete.

Sebastian Rahtz: On your recommendation last year I bought a copy of *Life: A User's Manual* by Georges Perec, which I am still trying to read. I wonder while I'm still trying to read it whether you would like to recommend a film or a piece of music which has equal meaning to you as Georges Perec's book.

DEK: I guess when I made my home pages, a few years ago, one of the pages listed books that I was recommending to read. There's this incredibly different book by Georges Perec called *Life: A User's Manual*, which is a combination of many different kinds of artistry. It has a mathematical basis, but still becomes, I think, a great work of fiction. He developed this book with 99 chapters. (There should really be 100 chapters, but each chapter was based on a lot of mathematical constraints and one of his rules was that you had to break one of the rules, so naturally it has only 99 chapters instead.) It is the story of the people who live in an apartment block in Paris. There are 10 floors and 10 apartments on each floor, and you go through the apartments — actually some of the apartments have several rooms — but you go through the apartments in the order of a knight's tour. Eventually you find out about the lives of all these people, and there are many other very interesting constraints that he put into the book. Each chapter is a little short story, kind of independent of the others. Now you are asking if there's something else analogous, in the domain of music or ...

Sebastian Rahtz: Film.

DEK: ... or film. O.K. The closest thing in film is this new movie *Run Lola Run*, from Germany. If you were to do it the way Perec did it you would have many more chapters, but *Run Lola Run* gives you a story three times. The first time ends in disaster and so Lola says 'No take me back. Let's do it again.' So we start over and she does something slightly different in the first scene and then we go through the whole story again, but everything happens five seconds later, so certain accidents don't occur in the streets and the whole plot changes. At the end of the second telling of the story it's another disaster, not for her, but for her boyfriend, and that's too terrible to accept. The third version of the story leads to a happier fate.

In music I suppose I think of a theme that had been proposed to Bach, I think by one of the noblemen of his time. He supposedly improvised a theme on that melody spontaneously, but then he was fascinated by it afterwards. During the last year of his life he prepared a manuscript that he left unfinished at his death, called the *Art of the Fugue*. That work is analogous to Perec's, because the idea is to make a thing of beauty while working within tight constraints.

Sebastian Rahtz: Good, thank you. [*Pause*]

DEK: Well, if there are no more questions ... [*Laughter*]

Elizabeth Gilliart: What are you going to write next?

DEK: Yes, I tend to be writing about a page a day and so there's nobody alive that has read everything I've written, except perhaps me. But I'm just finishing now a book that is incredibly specialised. It will be in the Springer Lecture Series, Lecture Notes on Computer Science, and it's called *MMIXware*. It's a set of computer programs to simulate the new RISC computer that I designed this year. MMIX is a computer for the new millennium. The first 'M' is for millennium and it replaces a computer called MIX that I used in my books on computer programming. I had the privilege of working with the designer of the DEC Alpha chip, Dick Sites, who is one of my students; John Hennessy, the designer of the MIPS chip, also was a participant in this design, and a few other people in Silicon Valley. We came up with something I think would be a fairly good machine to build in about ten years. It tries to be the cleanest computer design, and easy to learn, fairly nice to look at, and to make theories about; there is a group of people that are helping me use this new computer to rewrite the algorithms that I had written for the old computer. *MMIXware* is a set of software programs that make MMIX live even though it hasn't been built yet. The exciting thing to me is the pipeline simulator, which is a meta simulator — which means that it can simulate millions and millions of different kinds of possible implementations, or even implementations that nobody knows how to build. You can say for instance how much memory it has, what kind of caches it has, what kind of strategies the caches use to remember the recent information; a pipelined computer has different functional units, you can say how many multipliers you have and you can define a functional unit to handle any subset of the 256 operation codes; you can have any number of functional units, you can issue any number of instructions simultaneously, you can look ahead different ways and control all kinds of things. And then you can find out how fast your programs run. Now if we had that machine we could put TEX onto MMIX and see how fast it works with different kinds of caches. This program for the meta simulator is probably the most difficult computer program I ever wrote.

To me it was most interesting because it depends on the idea of literate programming that I used when I worked on TEX; for me perhaps the greatest spin-off of TEX was this idea of literate programming, which has helped me write computer programs of all kinds. I don't think I could ever have written the MMIX meta simulator without literate programming; it would have been too mind boggling. I would not have been able to get the whole thing together and debug it; it would have been too much of a mental strain, overburdening my head. In the past, literate programming has often helped me write better programs, but here for the first time it was crucial, or I couldn't have written the program at all. I don't think *MMIXware* would have been possible without a good documentation language to help me understand what I was doing as I went along. So this will be a book about 400–500 pages, but it's mostly just the typeset version of these literate programs for that new computer.

Then also I'll be finishing a sort of sequel to *Digital Typography*. *Digital Typography* is volume three of a set of collected papers; all the scientific papers that I've written are being divided into eight categories. The first book contained the things I wrote about *Literate Programming*. The second book was called *Selected Papers on Computer Science*; those were the papers I wrote for audiences that weren't primarily computer scientists; it collects the general works. And then the third volume was *Digital Typography*, and I told Phil I'd come here tonight because I'll do anything to sell copies of that book. The fourth volume will be called *Selected Papers on Analysis of Algorithms*, and those are my mathematical papers for what I think is my main unique life's work — the study of computer methods in a quantitative way: How good are particular computer methods, from a definite exact mathematical point of view? That book is estimated to be about 750 pages, and it will contain, I think 37, 38 papers on that subject. The material has all been scanned and put into TEX form. But I have to spend a few months in spare moments going through dotting the i's and crossing the t's and making the index, putting the bibliographies in a consistent format, finding out people's middle names for the index, things like that. And I also go through every paper and put it into the form in which I would like it to be remembered. So if a paper was written in the 70s and I used sexist pronouns, I change that; I try to rework it so that instead of saying 'he did it', I'll say 'they did it' or something. Also I change 'which' to 'that' a lot. It's an American thing.

Male Voice: Why?

DEK: It's actually because of an Englishman named Fowler who wrote *The King's English* and gave rules for 'which' versus 'that'. The people in America believed him and they started teaching English classes, and it was taken over then by the *New Yorker* and other well-edited journals until finally the copy editors who used Fowler's rules — Fowler's book came out in the 20s I think[1] — the copy editors who used Fowler's ...

Audience: Roberts' rules!

DEK: Whose rules?

Audience: Just after Roberts' rules!

DEK: I never heard of Roberts' rules of English, only *Roberts' Rules of Order*[2] — and I think your comment is out of order. Anyway, Fowler essentially gave an algorithm that pretty much boils down to this: Look at what comes before the word 'which'. If it's a preposition or a comma, then it's fine; 'which' can also be a pronoun. But otherwise you change it to 'that'. Well, it got to the point where almost all of the magazines in America and most of the newspapers by the end of the 70s were using 'which' versus 'that' in Fowler's recommended style. This was because the copy editors had risen through the ranks and won the battle. I had written the first draft of *The TEXbook* in the old way; but Guy Steele who was visiting Stanford from MIT, marked it all up, everywhere I had a wicked 'which'. Simultaneously my copy editor from Addison-Wesley was revising volume 2 of *The Art of Computer Programing*, which I had in TEX form before the 2nd edition came out. I was doing the typesetting in 1979 and 80, and that copy editor was also a member of this new generation. So that's when I learned the algorithm for wicked 'whiches'; I could do a search with a text editor, checking to see if it's following a preposition or a comma.

Soon I became very sensitive to this, as were a lot of Americans. I started to be irritated when people would quote a sentence from my earlier papers where I used a wicked 'which'. It was even hard for me to read the New English Bible because I would silently have to translate 'which' to 'that'. I go into

[1] H. W. Fowler *The King's English* (Oxford 1907). H. W. Fowler *A Dictionary of Modern English Usage* (Oxford 1924)

[2] Somebody named Paul Roberts wrote *Understanding Grammar* in 1954, but I don't think it was terrifically influential.

more detail in my book *Mathematical Writing*, published by the Math Association of America. The British perspective I understand is different still, but in America it's very much a stylistic lapse that calls attention to itself if you don't follow the convention. You can go through now say 100% of the magazines in the last 10 or 15 years and you very rarely see a wicked 'which'.

James Foster: You said that the MMIXware book was for Springer; did they ask you to use their LaTeX 2.09 style files? [*Laughter*]

DEK: I take it that LaTeX 2.09 style files are a joke because that's an old version of LaTeX?

James Foster: We had to use them recently.

DEK: I was given a free hand here. But I'm supplying camera copy so we'll which is a special version of CWEAVE that makes little indexes on the right-hand page. One of the chapters in *Digital Typography* talks about this: 'Mini-Indexes for Literate Programs' is the title of that chapter. Basically when you are reading a computer program every name of a variable on the page that you are reading is either defined on that page or you can find it in the mini-index. And the mini-index will tell you that the variable is, say, defined in section 5, and it is a procedure name or a constant and so on. It gives you a quick reference. It's an idea that I picked up from some textbooks on languages, where people would be learning French or Russian. Given a short story in some other language, every word in the vocabulary that you might have to look up in a dictionary appears in an index on that page. With CTWILL I have a bunch of macros that check the uses and definitions of everything on a page. The macros automatically prepare mini-indexes; there is a separate pass to sort the words, but the first pass does the layout. Some of the more difficult — probably the most difficult — TeX macros that I ever wrote are involved in that. I call this program TWILL because it's sort of a double WEAVE. Anyway the CTWILL program gives me the output that I have to send to Springer.

So no, I won't be using their style files. I did one other book in their series but that was some years ago, before they had learnt to insist on those things I suppose. For some purposes of course these extra format restrictions are not only for your creativity but also for electronic archiving. It's not the kind of restrictions that Perec would have added, but I imagine the publisher sees that it's to their advantage to have as many things in that style as possible. It depends on what you are doing, because that can also stifle what you want to say. One of the things uppermost in my mind from the beginning of TeX was that I would have the freedom to introduce a new notation if it was the right thing for the subject I was writing about. If I wanted to make up a new notation I wouldn't have to go through any middlemen who wouldn't understand the notation. I would be able to typeset it and I would know that it was getting through in the way I wanted it.

Philip Taylor: I have some bad news: They really do want us out of here by half past ten, so there's time for just a couple more questions.

Jonathan Fine: I hope it's not too late to ask this question. Why did you introduce category codes?

DEK: Well, if you look at this book *Digital Typography* it shows the original draft of TeX that I made the first night when I stayed up late typing a proposed design. The feature was even more awful in those days, it handled not only category codes but all kinds of penalty amounts and other parameters; anything that I didn't know how to fix as a constant in TeX I just left for later, so that I could change it. But you know originally I didn't think of having many users; TeX was a system for me and my secretary, so I wasn't thinking of things in much generality. So the same mechanism that I had originally for category codes was also used for things like how many characters before breaking for a hyphen, what was the penalty for widows, et cetera, all done with the same mechanism 'chpar'. C-H-P-A-R it was called: Change Parameter. I had no idea of generality in the first place.

Then later on, as I saw the applications Math Reviews were making, I realized that I'd better have some way to allow more flexibility... Jonathan, you'd be interested to know that the very first implementation of proto TeX was done without recognizing control sequences as words; each character was read in as a character by itself and then the 'hash table' lookup would go on afterwards. The first implementation, which was done by my students in the summer of 77 while I was in China, was very much like Active TeX in that sense. We found out that we could make the program run a lot faster in its inner loop if we distinguished control sequences from ordinary text, but originally every character was active in the very first draft. In this book *Digital Typography* I resurrected the computer files that I had used when first getting my thoughts in order.

Dominik Wujastyk: Don, you made a change in TeX when you made it able to read an eight-bit character, and that was very important for European users and others. I completely respect and like the idea that TeX is fixed, that you've finished with it as it were for the moment, and that it is a fixed point and it's not going to change anymore. However it does look as if TeX will have to go Unicode. There is of course Omega already; I've never used Omega, I haven't had the courage to take it on yet, but I think it's there in my future, and of course other people are using it. I just wonder about whether it's going to just always be Omega or something like it and TeX, and they're just going to become separate things and go their own ways; or could you give a Unicode version of TeX some imprimatur or some special blessing, so that it becomes the one that everyone uses. I wonder whether all the development efforts can somehow be brought together again.

DEK: Yeah! It seems so difficult. I'm a great fan of Unicode, but I also know enough about it to know that it's incredibly complicated, and that I would never have a system based on Unicode that I would be able to say has no bugs in it because of the extra complexity. Well, I like TeX, I like having a program that is, if not 100% reliable, it's 99.9999 — it's as reliable as anything. So solid that we can build on it. This means however that I'm not supporting all of the important languages that Unicode supports. Still the job of doing all of those in one system is so incredibly hard that I don't know where the expertise is going to come from to get it to such a well debugged level. For me to put my imprimatur on something would require so much work, I would have to check that the thing had been done right, and there is so much involved in getting it done right. As you know, Unicode 3.0 which will be coming out early next year, really covers almost all the languages of everyone alive today; they have filled in the last gaps, they've got Burmese and the Maldive Islands, Sri Lanka, the places where the political difficulties were; they have several thousand extra Vietnamese and Chinese characters and so on. And Yi, and Mongolian, and native American — various Inuit and Algonquian languages — are all there now.

But each of these languages has special difficulties involved in the typesetting. It's not just a matter of getting the symbols; all kinds of ligatures and things must go in, and rewriting of characters. Many of the languages have no spaces between words, and special hyphenation conventions, and a total user community of a few thousand. Moreover, all the people who do use some of these languages are educated enough that writing in English gives them more job security; thus the more they do to make it possible for everybody else in their group to use the system, the less chance they have of being uniquely able to do anything. So it's going to be hard to support this commercially; it's surely going to be a volunteer effort. The effort is not only an order of magnitude more difficult than what I had to do, but it also has to be done pretty much as a labour of love, as I did it. So it looks to be a while before it could converge like that — not that it's impossible, but I myself wouldn't be in a position to bless it. All I can do is provide an example of one of the world's nearly bug-free programs so that other people can try to emulate the good points and correct the bad points.

Well, thank you very much. [*Applause*]

Philip Taylor: Don, I'd like to thank you very much indeed, on behalf, not only of the Committee of the U.K. TeX Users Group, but of every one of the members here who, I'm sure, are absolutely delighted that you have spoken to them. Thank you very much indeed for your time, for joining us. Jill, thank you very much indeed as well for coming along. It's been a great pleasure to have you both in our midst. We wish you a very safe and happy stay in the U.K. for the rest of your trip. Thank you very much indeed.

— — ∗ — —

Editor's note: Earlier interviews and Q&A sessions with Don can be found in these issues of *TUGboat*:

- **7** No. 2 (1986): The "coming out party" for *Computers & Typesetting*, May 1986
- **13** No. 4 (1992): Q&A with the Nordic group and conversation with Roswitha Graham, November 1991
- **17** No. 1 (1996): TUG '95 Q&A
- **17** No. 4 (1996): Q&A Amsterdam (NTG) and Prague (CSTUG), March 1996
- **21** No. 2 (2000): Interview by "Advogato", January 2000

Several of these sessions are already posted on the TUG web pages, and the rest will be posted when time permits.

How (LA)TEX changed the face of Mathematics: An E-interview with Leslie Lamport, the author of LATEX*

A great deal of mathematics, including this journal, is typeset with TEX or LATEX; this has made a lasting change on the face of (published) mathematics, and has also permanently revolutionized mathematics publishing. Many mathematicians typeset their own mathematics with these systems, and this has also changed mathematical thinking, so that in a casual conversation one might write `\sqrt{2}` *instead of* $\sqrt{2}$ *on the tablecloth... We take "ca. 20 years of TEX" as the occasion to ask Leslie Lamport, the author of LATEX, some questions.* (GMZ)

— — * — —

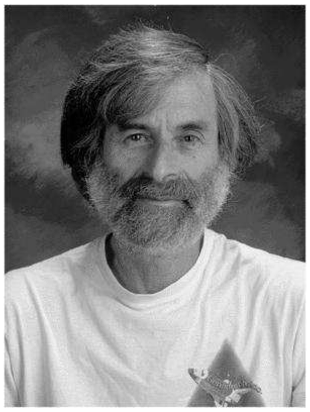
GMZ: *How were your own first papers produced? Did you start out on a typewriter? On roff/troff/nroff?*

LL: Typically, when writing a paper, I would write a first draft in pen, then go to typewritten drafts. I would edit each typed draft with pencil or pen until it became unreadable, and would then type the next draft. I think I usually had two typewritten drafts. I would then have a secretary produce a nicely typed "final" version, which would usually be subject only to minor changes. I went on-line around 1977, using TVedit and a primitive text-formatting system that I believe was called Pub. I switched to Scribe when it became available (maybe 1978?) and switched to TEX perhaps a year later. I first used Unix when I moved to DEC in 1985, so I was never a *roff user.

GMZ: *Could you tell us about the pre-history: Don Knuth wrote TEX in the seventies. It was working but hard to use. People tried, some wrote macros, ... What was the situation when you "got started"?*

LL: When Don was creating TEX80(?), the second version of TEX, the popular macro package at the time was one written by Max Diaz — I've forgotten its name.[1] I was in the process of starting to write a book, and I found Diaz's macros inadequate. So, I needed to write a set of macros for the book. I figured that, with a little extra effort, I could make a macro package that could be used by other people as well. That was the origin of LATEX.

GMZ: *Was this always meant to be "free software"? Did you ever try to "get rich" with it? Do you regret that you didn't?*



LL: At the time, it never really occurred to me that people would pay money for software. I certainly didn't think that people would pay money for a book about software. Fortunately, Peter Gordon at Addison-Wesley convinced me to turn the LATEX manual into a book. In retrospect, I think I made more money by giving the software away and selling the book than I would have by trying to sell the software. I don't think TEX and LATEX would have become popular had they not been free. Indeed, I think most users would have been happier with Scribe. Had Scribe been free and had it continued to be supported, I suspect it would have won out over TEX. On the other hand, I think it would have been supplanted more quickly by Word than TEX has been.

GMZ: *Tell us about your "comic/tragic experiences trying to get computer scientists and computer science journals to enter the computer age".*

LL: People will go to great lengths to avoid having to change what they do. In the early days of LATEX, my colleagues at SRI would always tell me that they would write their next paper in LATEX. A few years ago I got fed up with the fact that computer science journals were still sending around paper manuscripts for review. I circulated a message saying that computer scientists should refuse to review paper manuscripts — except in unusual circumstances, such as submissions from third-world countries. One editor complained that she was handling so many papers that the cost of disk storage for all of them would have been prohibitive. A simple calculation showed that, with disk prices at the time, the storage would have cost about $250 — less than the cost of the filing cabinet she was then using. (Now, of course, it would be about $2.50.)

In the late 80's, I proposed to the ACM that they should create standard document styles or macro packages for what were then the three major

*This interview was originally published in the *Mitteilungen der Deutschen Mathematiker-Vereinigung* 1/2000, pages 49–51. Reprinted with permission.
The interview was conducted by Günter M. Ziegler

[1] Editor's note: Fácil TEX

formatting programs, TEX/LATEX, troff, and Scribe. While their journals would accept paper submissions as usual, authors who submitted papers electronically in one of those styles would have the benefit of electronic transmission speeds. An editor at ACM dismissed the idea because it was unfair to force people who didn't have access to computers to submit their papers electronically. (I can assure you that I'm not making this up; my imagination isn't that fertile.)

People will switch to something new only if they're forced to by circumstances. People started using TEX because pencil and paper became untenable as a way to produce mathematical documents. Journals started accepting electronic submissions when it became impossible to ignore the Internet any longer.

GMZ: *Is LATEX hard to use?*

LL: It's easy to use—if you're one of the 2 % of the population who thinks logically and can read an instruction manual. The other 98 % of the population would find it very hard or impossible to use.

GMZ: *Why is there no high/same-quality WYSIWYG system available?*

LL: The entrance barrier is too high. To have any chance of success, a system would have to do everything that TEX does. That makes it too much work for any individual. A company like Microsoft could do it; I presume they don't because the market is too small. I occasionally think of going over to the Dark Side and proposing to Microsoft that they hire me and put me in charge of a group to develop such a system. Fortunately, I have other things to do that keep me out of trouble.

The speed of modern computers has removed some of the allure of WYSIWYG. TEX can process a 10-page paper in a couple of seconds. I have a simple Emacs macro that, with a single keystroke, processes and redisplays the paper I'm working on. So, when I'm writing a paper, I just have to type TEX source, I don't have to read it.

GMZ: *It's nearly frightening to what extent LATEX has now "solved all the problems" and seems to be without any (?) competitors?*

LL: It doesn't have any competitors in the technical sense of competition—that is, there's no other system that can do what it does. In the Darwinian sense, its competition is much too strong for it to survive. Kids these days use Word. As I already said, people are extremely reluctant to learn something new. When those kids grow up, they're not going to want to learn a new, arcane system. So, I expect the use of TEX and LATEX to die out. However, a mathematician just assured me that there is no alternative for math and physics, and he expects TEX to survive the 100 years that Don predicted. We'll see.

GMZ: *You say that people/kids won't "want to learn a new, arcane system." Couldn't it be fun (!) to learn that certain things don't work, exactly because one had made a logical error? LATEX as a computer game?*

LL: It's naive to expect something like LATEX, that's at best going to be used only by professional mathematicians and scientists, to filter down to the grade-school level. Even if there were some point to teaching kids such an esoteric system, it couldn't be done for the same reason that it's been impossible to raise the level of math and science education in this country—namely, kids can't learn from teachers who don't know the subject well, and people who are good in math and science don't become grade-school teachers.

GMZ: *Here is a recent email dialogue I had with a colleague in Toronto:*

```
>
> "Guenter M. Ziegler" wrote:
>
> >
> > Charming: people [CS professors!!]
> > still use troff! Weren't they forced by
> > law at some point to adopt TeX?
>
> Can't help it, I prefer to type,   .NH 3 than
> /subsubsection etc.
> But then I love unix's two letter commands also!
>
> By the way I can type , eg., .NH 6, does latex
> use /subsubsubsubsubsection ?
>
```

Please comment.

LL: The use of `\subsubsection` instead of `\sss` was a deliberate choice—inspired by Scribe—to make command names understandable instead of short. I think that was a good choice. The user who hates to type can always define `\sss` to mean `\subsubsection`. However, a technical writer typically spends many hours per page writing a document, and the time spent actually typing text is a negligible part of the work. That's probably why neither I nor anyone I know bothers defining shorter synonyms for commands.

One can argue that the use of `\subsubsection` etc. instead of `\section{3}` was a mistake. However, rather than `\section{3}`, a more logical approach would be a `\heading` command that creates a section heading at the current level, and commands to increase and decrease the current heading level. My feeling now is that the intuitive simplicity of the current system outweighs the advantages of the logical approach; but others might certainly disagree.

One thing along those lines that definitely was a mistake was the use of `\small`, `\large`, etc. instead of a `\size{n}` command along with commands to increase or decrease the size. I'm afraid I just copied the size-changing commands from Scribe without thinking.

GMZ: *Any regrets about things you should have done better when you "did it"? Lessons to be learned from that? (Knuth has published parts of his log books ...)*

LL: There are lots of mistakes that I made — such as the size-changing commands. But those are inevitable. You can find many of them by looking at the differences between LaTeX2.09 and LaTeX 2_ε. But the biggest mistake I made was not in how I designed LaTeX, but in how I didn't design TeX. When Don was writing TeX80, he announced that it would be a reimplementation of TeX78, but he was not going to add new features. I took him seriously and asked for almost no changes to TeX itself. The only change I can remember strongly urging involved page breaking. People who used TeX78 will remember that, when TeX couldn't find a good page break, it would very often produce a horrible one — a page containing two or three lines. I felt that this would be a real show-stopper — much worse than words extending to the right of the margin — so I lobbied hard for the change. However, there were many other improvements that I could have suggested but didn't. In the end, Don wound up making very big changes to TeX78. But they were all incremental, and there was never a point where he admitted that he was willing to make major changes. Had I known at the beginning how many changes he would be making, I would have tried to participate in the redesign. Don had a small group of helpers — mostly students — with whom he met regularly. I could have joined that group and perhaps have had some influence on the design. Who knows, maybe I could have persuaded him to replace TeX's macro-expansion language with something better. A macro-expansion language is good for a quick-and-dirty solution, so it was appropriate for TeX78. But it's not good for serious programming because you always have to fight to get things expanded at the right time.

GMZ: *Three LaTeX mistakes that people should stop making?*

LL: 1. Worrying too much about formatting and not enough about content. 2. Worrying too much about formatting and not enough about content. 3. Worrying too much about formatting and not enough about content.

GMZ: *What's your view on mathematical typesetting in the future? Quantum leaps ahead?*

LL: I'm pessimistic about software in general. When computers were the province of the technically sophisticated, people wrote software for technically sophisticated users. Now, technically sophisticated users are an insignificant niche market. Standards are being driven by the marketplace, which cares only about the masses. So, mathematicians have no place in the brave new world of computing. They will have to make do with the same flashy but technically impoverished tools that the little old lady in Peoria uses. So, you can display video animations on the web, but there's still no good way to display a mathematical equation.

The future of technical communication is the World Wide Web and the CD-ROM. There may soon be a window of opportunity for two products: one for "typesetting" math for the Web, and the other for creating CD-ROM textbooks. The proposed standard for adding math features to html will, if adopted, make it possible to produce poorly formatted but readable math html documents.

Computers make possible all sorts of new forms of communication. For example, one can have a sort of permanent workshop which consists of a set of technical presentations combined with a chat room. Based on the chat-room discussions, participants can continually refine the technical presentations. It could be something like a "living Bourbaki" for a subject.

However, mathematicians, like all people, are extremely conservative. For example, they still write proofs essentially the same way they've been doing it for centuries. I believe I've demonstrated in

```
AUTHOR  = "Leslie Lamport",
TITLE   = "How to Write a Proof",
JOURNAL = "American Mathematical Monthly",
VOLUME  = 102,
NUMBER  = 7,
YEAR    = 1995,
Month   = "August-September",
Pages   = "600--608"
```

that there's a better way. But they are just as reluctant to try it as they are to try anything new. Their excuses make no more sense than the ones I heard 15 years ago to explain why they weren't switching to (LA)TeX.

Typography

Typographers' Inn

Peter Flynn

1 CAP*itali*ZaTiOn

The first time I used BibTeX I was horrified to find it had capitalized — by itself — all the titles which I had so carefully typed with a single initial capital and subsequent continuous lowercase. Fortunately it didn't take much reading of the fine manual and the FAQ[1] to discover that enclosing the title in an additional pair of curly braces would stop this until such time as I could identify a suitable bibliographic style that didn't use capitalization (or learn how to hack BibTeX styles myself).

Recently there have been discussions on Usenet about this topic (in `alt.usage.english` as well as `comp.text.tex` and elsewhere) which led Markus Kuhn to ask if there were any good standards or rational preferences.

Standards there certainly are: the FAQ mentioned above refers to the *Chicago Manual of Style* as BibTeX's source for the behavior of its default styles, and Markus himself referred to the various ISO standards which recommend a different approach.

But it's hard to be rational about preferences in style which are essentially æsthetic. While there may be a separate rationality in the approach each standard takes, the choice between them is largely a matter of convention and history. As in other areas of typography, there seems to be a clear Atlantic Divide between the Pros on the North American side preferring the capitalization of each word except pronouns, prepositions, articles, and conjunctions; and the Antis on the European side going for capitalization only of the first word and any proper nouns.

The first method is perhaps more formal than the second, but tends to look slightly dated nowadays (it always reminds me of the chapter summaries in books of an earlier period, *In which the Author Betakes himself to London to Procure a Sufficiency of Cutlery against Dining with Friends*), but it has the advantage that it stands out as a title without the need for bold or sans-serif type: just a larger size is enough. Where the title is in a sans or a contrasting face, the second method looks less strained.

[1] The present UKTUG FAQ has a good answer at `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=capbibtex`.

The overriding problem for BibTeX users, however, is not the style itself, but the automation of capitalization. When all the titles of a bibliography are in common English, without technical phrases, foreign words, scientific terms, or quoted titles, automated capitalization works very well most of the time. But technical, research, and academic bibliographies typically use many words in a special sense, or draw them from a specialist vocabulary, making automation less reliable. This, and not any inherent dislike of capitalization itself, seems to be what triggers the user to declare, "I don't like|want|need capitalized titles."

Perhaps less well documented (and certainly mentioned only rarely in training) is that if the user is writing for a publisher, using the publisher's styles, it's not the user's choice whether or not the titles are capitalized: it's dictated by the class or package.

2 Compliments

At TUG'2001 in Delaware I showed a draft brochure I had put together to publicise LaTeX in my area.[2] This document includes the font sampler I mentioned last time, albeit as an illustration at a much reduced size. Regrettably, the PDF version used in the PDF leaflet has been badly blurred by Acrobat in conversion from the original PostScript as it contains several bitmap fonts, but there is also a `.ps.gz` version at the location given, which is much clearer. It appears that if an EPS file is converted to PDF by Acrobat, and it contains at least one bitmap font, *all* the fonts in the EPS get blurred.

The comments I made in my presentation at the time are in the *Proceedings* issue (*TUGboat* 22.3), but I ended with an appeal which I repeat here:

If you know or encounter users who are pleased and happy with LaTeX, especially about what it can do typographically, get them to write a sentence or a paragraph saying why, and send it to me (email address at end) for use in publicity.[3]

We don't blow LaTeX's trumpet often enough, and we need quotable quotes from people with demonstrable experience.

◇ Peter Flynn
Computer Centre, University College, Cork, Ireland
`pf@ucc.ie`

[2] The document is still available online at `http://www.silmaril.ie/documents/latex-brochure/leaflet.pdf`. US users should note this version is formatted to print on A3 paper.

[3] But please make sure they agree to being quoted in public!

Font Forum

Laudatio for Professor Hermann Zapf

Frank Mittelbach

Honored Professor Zapf,
Ladies and Gentlemen.

A Laudatio, according to the dictionary[1], is a "celebratory speech within the framework of a ceremony in which someone's accomplishments and services are honored." The dictionary further informs me, as someone who has never learned Latin, that there is a relationship to laudare (which means praise) and to laudes (which means the singing of praises).

Now, I doubt that my singing would be appreciated, so I will restrict myself to praise. It gives pleasure to praise as it also gives pleasure to be rightly praised, and this right Professor Zapf has earned in all areas in which he has worked.

Now, many proverbs advise that one should avoid too much praising and in view of the fact that in a concert hall one is not allowed to cough, and during a eulogy one is not allowed to yawn, I am well advised not to bore anyone here with too many details.

Speaking of praise, I should perhaps begin with DANTE, the German TEX users organization, or more exactly with its members, or even more exactly with the great majority of TEX users in the entire world. Their affinity and love for typography is the reason that brings us together here today. It is their wish to honor Professor Zapf for his considerable contribution to twentieth century typography, and particularly for his influence on digital typography.

The species "TEXie" is a strange beast. In the age of Microsoft Word and Co, its members steadfastly refuse to deal with the computer as they should, cobbling together text with mouse-clicks and pull-down menus. Rather they garnish their text with strange, useless signs like the backslash, fancy brackets, and incomprehensible commands. And all that to avoid widows, orphans[2] and other obscure things.

These ladies and gentlemen speak of badly kerned fonts, of high-quality composition, throwing around words such as quad, leaving the rest of the world bewildered: "What do these people want? With my word processor, everything is much faster and simpler, and I can see right away what it looks like!"

"Yes," answers the devoted TEXie, "that's exactly why it looks like *that.*"

Fanatics? Crackpots? People that time has passed by? — After all, TEX was created more than twenty years ago. Can such a dinosaur of the computer age still have any relevance? Can one still take its champions seriously in the age of WYSIWYG[3] and multimedia?

One can and one should. The rules of typography also hold true for text that has not been set in lead. Promoting these rules is even more important nowadays than in the past, because with 'desktop publishing for all', the knowledge of both the compositor and the typographer is under threat unless software takes over some portion of these tasks.

Even though TEX is now middle-aged, its roots are firmly anchored in the foundations of typographic quality. Many may be astounded — but even after such a long time there is hardly any other computer program whose typesetting quality approaches that of TEX, let alone surpasses it.

There are many reasons for this. The most important is probably that Professor Knuth was able to build on the friendship, the knowledge and the help of such notables as Hermann Zapf, Charles Bigelow and Richard Southall. Their capacity for passing on their deep typographic knowledge, and ideas for its realization in the computer, have had a decisive influence on TEX as we know it today. Their cooperation helped to place TEXnow, after twenty years, among the best typesetting programs.

Donald Knuth started the TEX project with the goal of developing, in approximately half a year, a computer program with whose help his secretary would be able to typeset his books (in particular, of course, *The Art of Computer Programming*). This was his reaction to, in his opinion, the terrible deterioration of quality of his and other people's books through the increased use of computers in publishing.

As we know, that half-year became about ten years. As Don had to learn, typography can't be compressed into a computer program in half a year — actually, not even in ten years! But during this time, with knowledgeable help, a framework

This is a translation of the article "Laudatio auf Professor Hermann Zapf", which appeared in *Die TEXnische Komödie*, 1/2000, pages 31–36. Reprinted with permission.

[1] Duden, Deutsches Universalwörterbuch, Mannheim 1989

[2] Editor's note: The German terms, Schusterjungen (cobbler boys) and Hurenkinder (children of whores), are much more colorful than the English.

[3] What you see is what you get

could be built that still nourishes us today. During these years, Don changed from a computer specialist into an apprentice of the black arts, and I think we can justifiably thank his masters during that time, that he learned so well.

But I promised at the beginning to keep this speech short, so I should not digress any further from my Laudatio, but finally devote myself to the theme.

— — ∗ — —

There are many biographies about Hermann Zapf and I dare say that there is no contemporary book on the topic of typography of our time — at least none that one can take seriously — in which his name is not mentioned. That is not really surprising: With more than 170 different fonts to his credit, including Palatino, Saphir and Optima, he is not without reason considered the most important font designer of our century.

I therefore want to limit myself here to a few bibliographical highlights and then turn to some areas usually not mentioned in short biographies of Hermann Zapf.

Hermann Zapf was born in 1918 in Nürnberg (Nuremberg). According to his own autobiography[4], from early youth he was interested in technology and planned to become an electrical engineer, a career he could not take up because of political conditions.

Instead, he began his typographic career in 1934 as an apprentice to a photographic retoucher. His autodidactic study of typography led in 1938, after he finished his apprenticeship, to his first Fraktur font, "Gilgenart", which was designed for Stempel AG. After the turmoil of war, from 1948 to 1950 he worked as an assistant professor at the Werkskunstschule[5] in Offenbach, and from 1947 to 1956 as an artistic leader for Stempel AG, after that from 1956 to 1974 as a consultant to the Mergenthaler Linotype Company. During this time some of Hermann Zapf's best known fonts were created, for instance Palatino (1948) or Optima (1952).

In 1974 he was awarded the Gutenberg Prize of the City of Mainz. The Laudator, Giovanni Mardersteig, placed Hermann Zapf's accomplishments into the contemporary process of transition from lead type to electronic setting and film setting.

Right back in the early sixties, Hermann Zapf began to develop ideas for using the computer profitably in typography. But until the eighties these ideas fell on deaf ears, at least in Germany; even at the Technical University of Darmstadt, where he taught typography between 1972 and 1981, he could not interest anybody in research in this direction.

Research institutions in America were more open-minded. In 1976, Hermann Zapf was appointed Professor for Computer Typography at the Rochester Institute of Technology, where he taught until 1987.

In 1980, Hermann Zapf began his collaboration with Don Knuth on the Euler project at Stanford. The results of this project, a collection of beautiful scripts for mathematical typesetting, were made publicly available in 1985. Unfortunately, at least in my eyes, these beautiful fonts have not yet been distributed as widely as they deserve to be. But even if the direct success of that project seems small, the work still had wide-ranging implications, not least on the development of METAFONT84, which was created during that time.

Hermann Zapf made use of his experiences and results from the years of teaching in the United States, in his collaboration with URW in Hamburg, in the development of a suite of programs which collectively have became known under the name "hz-Program".[6] These programs expanded on the ideas that had been developed by Don Knuth and Michael Plass for the production of high-quality line breaking, adding new dimensions such as microtypographical changes to individual letters for evening out the spacing in a line or by using "Kerning on the Fly". As far as I know, these algorithms are now licensed for the program InDesign, which is perhaps, from the point of view of the TeX world, the first time a competitor for high-quality (automatic) computer typesetting needs to be taken seriously.

But the development of TeX is not terminated either. Work such as that of Professor Zapf stimulates others around the world to experiment with extensions of TeX, all with the aim of improving the quality of documents typeset by computer. I hope that this development will continue into the future in a positive manner; certainly the enthusiasm of those involved is a necessary prerequisite, but it also requires knowledge about the inherent values of typography and of people like Professor Zapf who have passed such knowledge on to us.

— — ∗ — —

[4] Select `http://www.fontexplorer.com/FontStore` and follow link to "Font Designers" and then select the "official homepages of Hermann Zapf".

[5] Art school

[6] Peter Karow: *hz-Programm*, Mikrotypographie für den anspruchsvollen Satz [Microtypography for fastidious typesetting], Gutenberg-Jahrbuch 1993, Mainz.

As the last task in *The TEXbook*, Don Knuth sent us TEXies on our way with the following: "Final exhortation: GO FORTH now and create *masterpieces of the publishing art!*"

For many of us this goal remains far in the future, but we have all learned to recognize and love good typography. And so, with an honorary membership in DANTE, we want to express our thanks to Professor Zapf for his priceless services to the art which to us is both precious and cherished.

◇ Frank Mittelbach
Mainz, Germany

My collaboration with Don Knuth and my font design work

Hermann Zapf

I shall report on my collaboration with Don Knuth and finish with some thoughts on typography. Since 1977 I have been teaching typographic computer programs at Rochester Institute of Technology. That was about the time when Steve Jobs and Stephen Wozniak were tinkering with their first Apple computer.

RIT was the first university to investigate typographic programming. The idea was to build a logical structure of typographic rules into repetitive functions of a program structure.

My collaboration with Don Knuth started in 1979, when he was working on his Computer Modern design. An extended correspondence preceded our first meeting. In 1980, I was invited to Stanford University, to work within the Computer Modern project.

Knuth was one of the first scientists to think about the appearance of mathematical text pages. The motive was one of his mathematical books which had been produced in England, with whose quality Knuth was not at all pleased.

That was the beginning of METAFONT,[1] a type font for scientific typesetting, on which he had been working since 1978. In a lecture on "Mathematical Typefaces" in 1979 he had reported on his ideas. This was published in the "Bulletin of the American Mathematical Society".[2]

Knuth's original intention was to base a new type shape on imitating electronically the strokes of a broad-nibbed pen. He even tried to simulate the effect of pressure on the nib in the resulting calligraphic shape. But our alphabet is not structured systematically; with *N*, *M* and *Z*, for example, the normal pen stroke breaks down at an angle of about 30 degrees. In 1980, with Stanford University students — among them David Siegel, whom I will mention again later — we employed METAFONT to develop Knuth's conception of Computer Modern. I stayed at Stanford for two weeks. It was extraordinary how fast Don Knuth grasped rather complicated details, as if he had previously worked on the design of letterforms.

In "Der Spiegel"[3] of 23 June 1980, there appeared an article about our research work with the obscure headline "Lieber Butter".[4] It is still a mystery to me how these people learned of the project and how they acquired the nice photograph of the two of us from the Stanford photographer. This article, with such a silly headline, has certainly never been read by any scientist; perhaps some farmers were interested. But one is simply amazed by the information sources and connections of this magazine. The headline "Lieber Butter" had been chosen according to "Der Spiegel" because Don Knuth had once said that he refused to eat margarine instead of butter.

About TEX, "Der Spiegel" wrote that it is a fundamental computer program to transform texts of arbitrary content or in any language into an optimal form — type size, a combination of different fonts, line spacing, appropriate hyphenation and distribution of words over complete paragraphs could all be controlled. End of citation.

Digital word processing originated with Dr. Rudolf Hell in Kiel.[5] He is regarded as the father of digital word processing. He started already in

This is a translation of the article "Meine Zusammenarbeit mit Don Knuth und meine Schriftentwürfe", which appeared in *Die TEXnische Komödie*, 1/2000, pages 37–44. Reprinted with permission. The translation was prepared by Dieter Glötzel. Footnotes and figures added in the translation.

[1] Editor's note: more likely Computer Modern, although work started on both at about the same time.

[2] Donald E. Knuth, "Mathematical Typography", *Bull. Amer. Math. Soc.* (*New Series*) **1** (1979), 337–372.

[3] a political weekly magazine in Germany

[4] this means "Butter preferred"

[5] a German town on the Baltic Sea

1925 with his so-called "Hellschreiber".[6] The letters were reduced to a set of small dots in order to transmit them. In 1964 he invented the "Digiset", the first electronic photocomposition machine, and thus began the era of digital resolution of type. In recognition of his pioneering work he was awarded the Gutenberg Prize of the International Gutenberg Society in Mainz in 1977. I designed the first digital alphabets for the Hell Digiset machine. These were "Marconi" in 1976 and "Edison" in 1978. Initially the resolution of the characters was relatively coarse, but after a just few years, the staircase-like structures along oblique lines had disappeared.

As a book designer one continues to try to find new ways to make production more rational and less costly. This led me to consider the idea of processing typographic information with computer programs.

At the TH Darmstadt,[7] where I have taught typography since 1973, nobody was keen on such ideas, and also the industry was not interested in typographical problems and tried instead to achieve higher typesetting speeds with their machines. So I talked about my intentions in the United States.

In 1964 I was invited to lecture on programmed typography at the Carpenter Center of Harvard University. A few years later I received a generous offer from the University of Texas. I was ready to move to Austin, but my wife would not, although everything looked rather attractive. During my stay in Austin I was appointed an "Honorary Citizen of the State of Texas" with all privileges. Presumably I would have been exempted from paying state taxes. But the dream was over.

In 1976 the Rochester Institute of Technology offered me the chance to be the successor to Professor Alexander Lawson, who had been teaching typography there since 1947. I would teach typographic computer programs for the first time. This was before MIT in Cambridge or Stanford in California started similar activities. We devised the most beautiful solutions, but in the end the realization failed at IBM and Xerox because of the huge amount of memory we would have needed for our programs.

In 1977, with some friends, I founded a company in New York which was to develop practical solutions with a menu-based user interface. We planned to develop programs with a simple structure in order to penetrate the American office market.

Back to Computer Modern. I regarded this type as a little too thin for a text font, but we could easily produce a stronger version from the METAFONT data later on. In particular, I foresaw problems with greatly reduced documents. As soon as we had digitized a structure of the type everything went rapidly.

The next extensive project together with Don Knuth was the type family "Euler" for the American Mathematical Society ($\mathcal{AMS}$) which started in 1979. It was the critical test for the first version of METAFONT. Knuth wanted to have alphabets fully adapted to mathematical typesetting which would fit nicely into a text. Several visits to Stanford and an extended exchange of letters accompanied the Euler project, which should have been finalized for the 200th anniversary of Euler's death. You will surely know Leonhard Euler. He was born in Basel[8] in 1707 and died in St. Petersburg[9] in 1783. But it took us until 1985 to finish the type family named after him. Based on the experience with the representation of type characters, Don Knuth developed an improved version of METAFONT which included also an outline method for producing complicated type shapes.

David Siegel together with other students of Knuth worked with us again on the digitization of the Euler type. In 1985 he wrote the report "The Euler Project at Stanford" for the Department of Computer Science of Stanford University.

A further documentation on the complete Euler type family, which included in addition to the Latin alphabet also a Greek, a Fraktur and an italic, was edited by Don Knuth and myself in 1989. It was published in Canada under the title "$\mathcal{AMS}$-Euler. A New Typeface for Mathematics".[10]

A quite unusual task was the Bible project "3:16", which I worked on with Don Knuth starting in 1989. It was an unusual effort finding all Bible verses starting with 3:16 and then interpreting these anew. He had been working on this project since 1977. We had a lot of fun, and it ended with publication of the book "Donald E. Knuth. 3:16 — Bible Texts Illuminated". When the book appeared, a series of exhibitions was organized, which showed the work of calligraphers from all over the world who had shaped calligraphically the Bible texts selected by Don Knuth.

It remains inexplicable how this man always finds enough time, besides his teaching profession and publishing comprehensive books on the "Art of Computer Programming" — one volume has been dedicated to me. Then to study the whole Bible

[6] "Hell-writer"; Editor's note: "hell" means bright, as light — a particularly apt name for a machine that does its work with a beam of light.

[7] Technical University of Darmstadt, Germany

[8] Switzerland

[9] Russia

[10] *Scholarly Publishing* **20** (1989), No. 3, 131–157.

and on top of that his frequent travels. At home he has an organ on which he loves to play to relax — if he finds time and is not busy working on a difficult problem. When he visited Germany we had always to find a beautiful organ for him to play. And in spite of all that he is always available for technical discussions.

You all know my work as a type designer. Some of my alphabets, such as Optima Antiqua and Palatino, are today part of the standard equipment of your PC or printer.

Maybe you've been annoyed with my Palatino Roman on your PC or Macintosh, because it does not provide enough special characters and accents. But be consoled, Microsoft will deliver an extended Palatino and then you will get 1200 characters for each of the Roman, Italic, Demibold and Demibold-Italic variants — a total of 4800 characters. Bill Gates was crazy enough to want all glyphs one could imagine including special variants of Latin letters as well as Greek and Cyrillic alphabets.

On a digital basis, today this is no longer a problem, because technically you can develop letters relatively fast on the computer screen. This also has a dark side. Unfortunately there are only a few authentic versions of Palatino. Many have been forged unscrupulously, and moreover, forged poorly. Palatino has the sad reputation to be the most often copied Roman font world wide. But that is a topic I don't want to discuss.

A few words on my new font, which has its own story. It is not intended for scholarly typesetting, but there are other areas for using type. It got the name Zapfino from an American who thought of the original idea. The history of this type started, as before, in Stanford with David Siegel. After he had finished his studies with Don Knuth, he wanted to start a business as a type designer based on his experience with METAFONT and Euler, and I should help him with his plans.

The project started in 1993. First he wanted to use some of my calligraphic work from 1944 to develop a novel script font. David Siegel developed for this purpose a kind of chaos program which he called Derrick. This was supposed to mix the different variants of letters within a word automatically. He wasn't able to find enough different instances of the individual letters to realize his idea. Theoretically everything worked fine, but how should an ordinary PC user manage such a technique, how should he select critical ligatures and insert these at the right position?

The complicated system soon became incomprehensible and, unfortunately, David Siegel suddenly lost all interest in our project when his girl friend ran away.

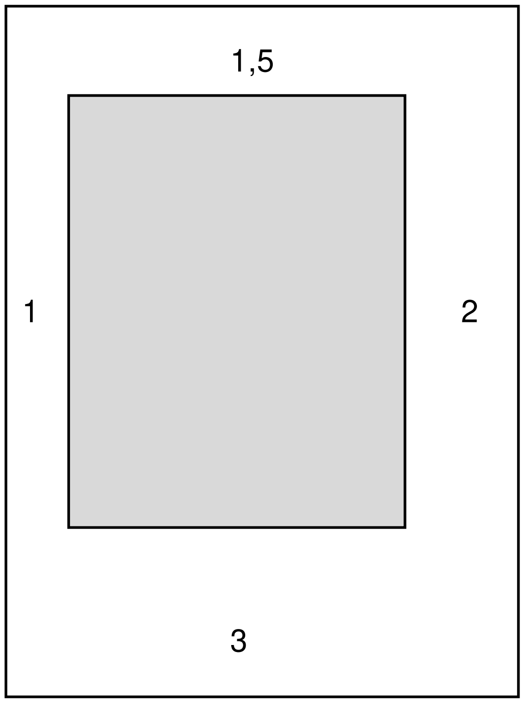


Figure 1: The classical proportions of the margins in a book: 1 - 1.5 - 2 - 3

The type then stayed for quite some time in my drawer until one day I showed it to Linotype Library GmbH in Bad Homburg.[11] We dared to continue to work on this unfinished program code. First we had to reduce the storage for characters to a reasonable size. We ended up with four alphabets with ligatures and extra large calligraphic flourishes, as well as 100 ornaments and special characters. By 1998 the work was finished, and it is now available on CD-ROM together with a movie about my work in calligraphy.

Typesetting with Zapfino is not so easy and needs a good feeling for the calligraphic features of the type. On the screen you have to check continually for overlapping characters because with the huge ascenders and descenders you easily get intersections of letters.

I was asked to present some thoughts on typography. There are enough typographic textbooks for sale from which to learn the basics. I will now talk about what you cannot learn from these books and give you some hints.

Do not invent a new type area — this has been developed with the experience of hundreds of years.

[11] a town close to Frankfurt

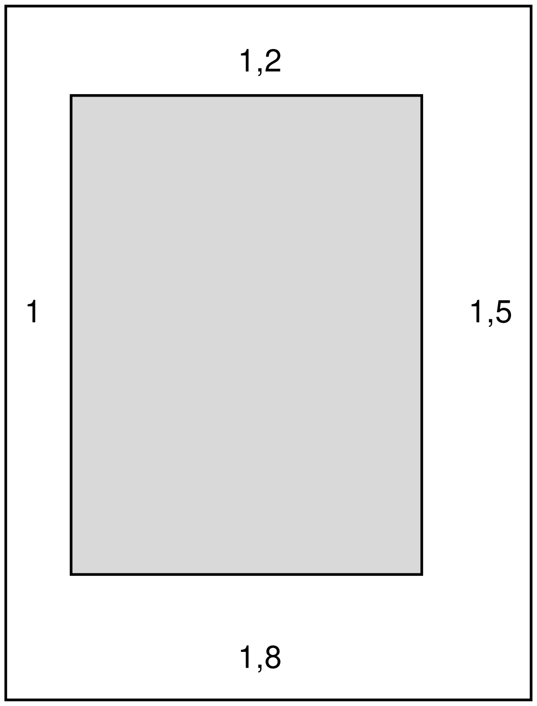


Figure 2: Proposal of the type area for a magazine: 1 - 1.2 - 1.5 - 1.8

If you are a mathematician, you may be tempted to position it in the geometric center. But you need functional margins of different widths. As a rule of thumb a progression of 1 – 2 – 3 and 4 is a good starting point. [See Figure 1 for the classical proportions of the margins in a book; Figure 2 proposes an adaptation for a magazine.] The inner margin for binding depends on the method of binding, whether you have a thread, wire or glue binding. With glue binding a wider inner margin is advantageous because you often need a flush cut. The outer margins also have a function. A wider outer margin is quite useful not only to be able to hold the book comfortably, but also to be able to make notes or annotations, and not just for controversial texts. In a scientific book at the Herzog-August-Bibliothek in Wolfenbüttel[12] I found marginal annotations with rather drastic terms, such as *Unfug* and *Quatsch*.[13] From the old historic shape of the letter *Q* one could determine that these were old annotations and had not been written by today's students.

Another point. When you have accumulated a thousand or more alphabets, you should not arrogantly try to display as many alphabets as possible in one document. Always seek the simplest and most significant form. A book is above all meant to be read and should not be a sandbox for typographic experiments.

An important typographic issue is the title on the spine of a book. You have to fit it onto the most narrow volume. How will you find your TEX periodical "Die TEXnische Komödie" on your bookshelf? [Figure 3 suggests an approach for narrow booklets. Figure 4 extends this to coping with loose sheets.] Whether the title on the spine runs from bottom to top, or from top to bottom is not that important, but whether a title is present on the spine at all, is. If you like, you can take part in the endless quarrel in Germany whether the title should run from the bottom up or the reverse.

But I ask you: Do you lay your book face down on your bedside table, when you put it away? This will answer any discussion whether the title should run up or down.

For "Die TEXnische Komödie" perhaps you could send out adhesive-back labels with the next issue; this would solve the problem.[14]

There remain many other things to be done in the digital world. Look closely at the characters on your computer screen. Most letters are corroded, without the fine details one needs to be able to recognize correctly the desired font. Here is an important job that remains to be done, although some technical prerequisites are needed to accomplish it. But when one can represent millions of colours, one should also in the future pay somewhat more attention to the presentation of the forms of alphabets.

If you want to learn more about the work of a font designer and what I have done in all these years, I recommend that you visit my permanent exhibition in the Herzog-August-Bibliothek in Wolfenbüttel. This proves that you do not have to die before you are awarded a permanent exhibition in a library.

◇ Hermann Zapf
Seitersweg 35
D-64287 Darmstadt
Germany

[12] a scientific library in Wolfenbüttel, Lower Saxony, Germany

[13] "nonsense, rubbish"

[14] Since 2001 DANTE has published "Die TEXnische Komödie" with a nicely printed title on the spine.

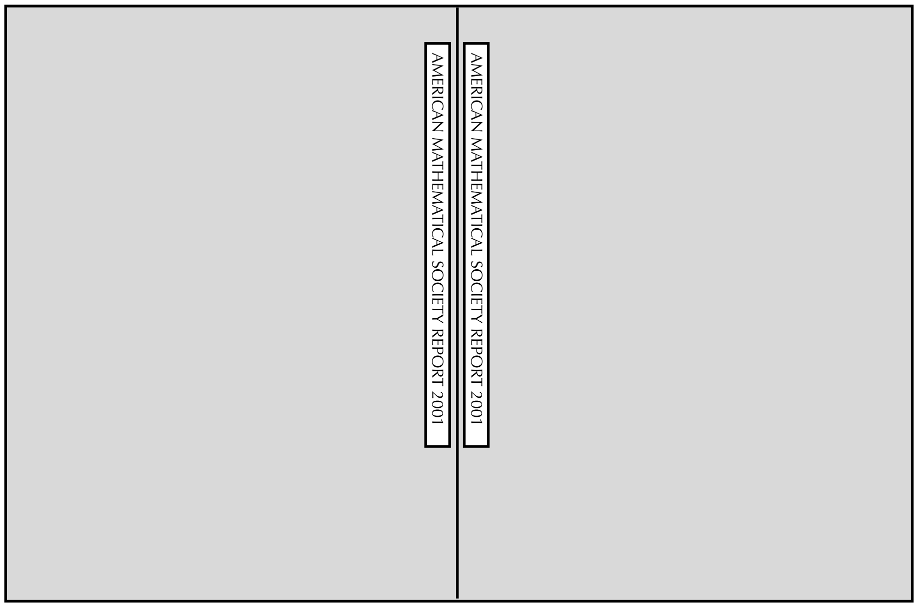

AMERICAN MATHEMATICAL SOCIETY REPORT 2001
AMERICAN MATHEMATICAL SOCIETY REPORT 2001

Figure 3: If no space for a title is on the spine of a booklet, paste a title on both sides of the fold. Put it onto the shelf of your library so that it stands out a quarter of an inch for easy finding.

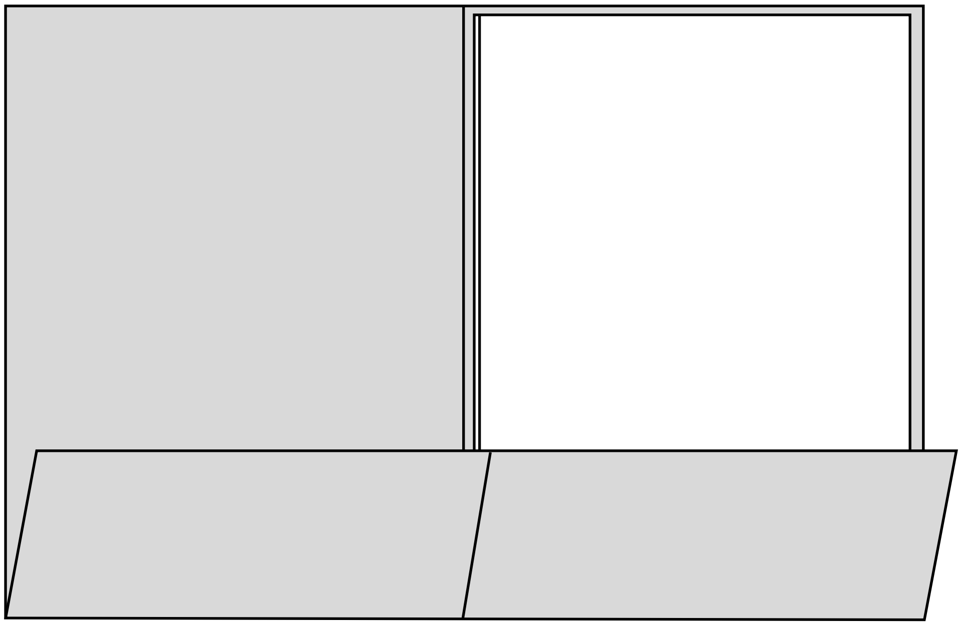


Figure 4: To keep loose sheets of cutouts from magazines, etc., put them in a folder of stiff colored paper and fold a pocket at the bottom. Paste titles as described above.

Software & Tools

Hyphenation Exception Log

Barbara Beeton

This is the periodic update of the list of words that TeX fails to hyphenate properly. The list last appeared in full in *TUGboat* 16, no. 1, starting on page 12, with an update in *TUGboat* 21, no. 2, pages 132–133. The present list contains only new items reported since then.

A copy of this article with the complete list can be found on the TUG Web pages, via a link in the on-line Table of Contents.

Owing to the length of the complete list, it has been subdivided into two parts: English words, and names and non-English words that occur in English texts. This update follows that division.

This list is specific to the hyphenation patterns that appear in the original `hyphen.tex`, that is, the patterns for U.S. English.

In the list below, the first column gives results from TeX's `\showhyphens{...}`; entries in the second column are suitable for inclusion in a `\hyphenation{...}` list.

In most instances, inflected forms are not shown for nouns and verbs; note that all forms must be specified in a `\hyphenation{...}` list if they occur in your document.

Thanks to all who have submitted entries to the list. Since some suggestions demonstrated a lack of familiarity with the rules of the hyphenation algorithm, here is a short reminder of the relevant idiosyncrasies. Hyphens will not be inserted before the number of letters specified by `\lefthyphenmin`, nor after the number of letters specified by `\righthyphenmin`. For U.S. English, `\lefthyphenmin=2` and `\righthyphenmin=3`; thus no word shorter than five letters will be hyphenated. (For the details, see *The TeXbook*, page 454. For a digression on other views of hyphenation rules, see below under "English hyphenation".) This particular rule is violated in some of the words listed; however, if a word is hyphenated correctly by TeX except for "missing" hyphens at the beginning or end, it has not been included here.

Some other permissible hyphens have been omitted for reasons of style or clarity. While this is at least partly a matter of personal taste, an author should think of the reader when deciding whether or not to permit just one more break-point in some obscure or confusing word. There really are times when a bit of rewriting is preferable.

One other warning: Some words can be more than one part of speech, depending on context, and have different hyphenations; for example, 'analyses' can be either a verb or a plural noun. No such ambiguous words appear in the present list.

The reference used to check these hyphenations is *Webster's Third New International Dictionary*, Unabridged.

English hyphenation

It has been pointed out to me that the hyphenation rules of British English are based on the etymology of the words being hyphenated as opposed to the "syllabic" principles used in the U.S. Furthermore, in the U.K., it is considered bad style to hyphenate a word after only two letters. In order to make TeX defer hyphenation until after three initial letters, set `\lefthyphenmin=3`.

Of course, British hyphenation patterns should be used as well. A set of patterns for UK English has been created by Dominik Wujastyk and Graham Toal, using Frank Liang's `PATGEN` and based on a file of 114925 British-hyphenated words generously made available to Dominik by Oxford University Press. (This list of words and the hyphenation break points in the words are copyright to the OUP and may not be redistributed.) The file of hyphenation patterns may be freely distributed; it is posted on CTAN in the file `tex-archive/language/hyphenation/ukhyph.tex` and can be retrieved by anonymous FTP or using a Web browser.

Hyphenation for languages other than English

Patterns now exist for many languages other than English, including languages using accented alphabets. CTAN holds an extensive collection of patterns in `tex-archive/language/hyphenation` and its subdirectories.

The List — English words

All entries in the list below have been reported since a supplement to the list was last published in 2000.

`acry-lamide`	acryl-amide
`acry-lalde-hyde`	acryl-alde-hyde
`al-ge-braically`	al-ge-bra-i-cal-ly
`anisotropy`	an-isot-ropy
`anisotropism`	an-isot-ro-pism
`anisotropic`	an-iso-trop-ic
`anisotrop-i-cally`	an-iso-trop-i-cal-ly
`archimedean`	ar-chi-me-dean
`bolome-ter`	bo-lom-e-ter
`catenoid`	cat-e-noid
`chemother-apy`	chemo-ther-apy

`cok-er-nel`	co-ker-nel
`collineation`	col-lin-ea-tion
`com-po-nen-t-wise`	com-po-nent-wise
`dacty-lo-gram(aph)`	dactyl-o-gram(aph)
`di-alec-tic`	dia-lec-tic
`di-alec-ti-cian`	dia-lec-ti-cian
`dyslec-tic`	dys-lec-tic
`eigen-class(es)`	ei-gen-class(es)
`eigen-value(s)`	ei-gen-val-ue(s)
`finitely`	fi-nite-ly
`home-o-mor-phic(ism)`	ho-meo-mor-phic(ism)
`in-du-bitable`	in-du-bi-ta-ble
`isotropy`	isot-ropy
`isotropic`	iso-trop-ic
`leafhop-per`	leaf-hop-per
`mi-croor-gan-ism`	micro-organ-ism
`newest`	new-est
`nonar-ith-metic`	non-ar-ith-met-ic
`nu-cleotide`	nu-cleo-tide
`phenyala-nine`	phenyl-ala-nine
`pipeline(s)`	pipe-line(s)
`plan-thop-per`	plant-hop-per
`plu-gin(s)`	plug-in(s)
`portable`	por-ta-ble
`posthu-mous`	post-hu-mous
`pseu-do-g-ra-pher`	pseu-dog-ra-pher
`pseu-dogroup`	pseu-do-group
`redi-rect(ion)`	re-di-rect(-ion)
`repo-si-tion`	re-po-si-tion
`rewire`	re-wire
`rewrite`	re-write
`rewrit-ten`	re-written
`scrutiny`	scru-ti-ny
`stere-o-graphic`	stereo-graph-ic
`warmest`	warm-est
`workaround`	work-around

Names and non-English words used in English text

`Kr-ishna`	Krishna
`Kr-ish-naism`	Krish-na-ism
`Kr-ish-nan`	Krish-nan
`Mark-to-ber-dorf`	Markt-ober-dorf
`Petro-vskiĭ`	Pe-trov-skiĭ
`vi-i-ith`	viiith
`vi-ith`	viith
`wis-senschaftlich`	wis-sen-schaft-licht
`xvi-i-ith`	xviiith
`xvi-ith`	xviith
`xxi-i-ird`	xxiiird
`xxi-ind`	xxiind

◇ Barbara Beeton
American Mathematical Society
`bnb@ams.org`

LyX — An Open Source Document Processor

Laura Elizabeth Jackson and Herbert Voß

Abstract

LyX — the so-called front-end to LaTeX, which is itself a front-end to TeX — tries to optimize something that LaTeX doesn't attempt: to make a high quality layout system like (La)TeX accessible to users who are relatively uninterested in serious typographical questions. LyX is not a word processor, but a document processor, because it takes care of many of the formatting details itself, details that the user need not be bothered with. All LyX features described here apply to the current official version LyX 1.1.6fix4. The significantly expanded version 1.2 is expected this summer.

1 History

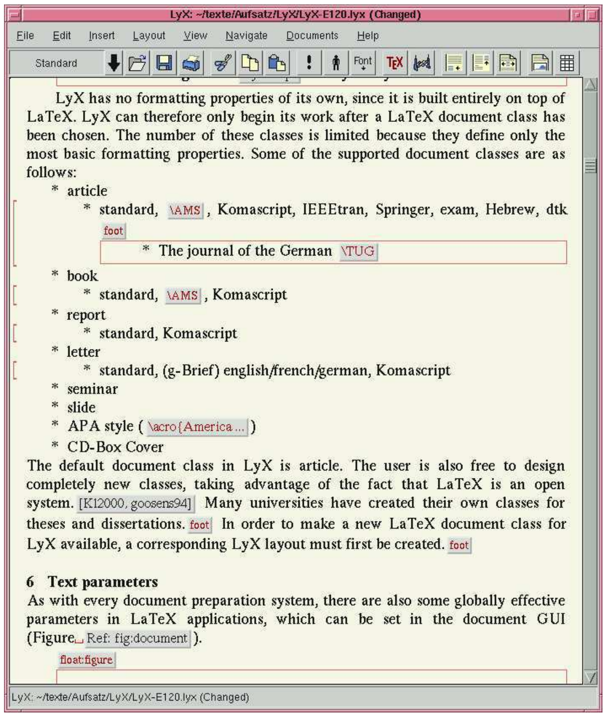
Drawing on the principles of the traditional printed word, Donald Knuth[1] developed a system that enabled users to prepare professional technical publications. Knuth called the first version of this system $\tau\epsilon\chi$, which has since come to be written TeX and pronounced "tech".

A graphical user interface is in principle not essential for LaTeX; however, in today's "all-things-bright-and-beautiful" world, the absence of a GUI makes LaTeX less accessible to the masses. This lack of a GUI was recognized in 1994/95 as the opportunity to create one; it began as a master's thesis and was first called Lyrik, then LyriX, and finally LyX. From the beginning, the main goal so that the user would be freed from having to delve into the details of (La)TeX usage. It remains to be seen whether (La)TeX can be left completely out of the discussion, because all other program environments sit on top of TeX. To understand the LyX philosophy, however, understanding (La)TeX is unimportant, especially for the user who specifically chose LyX for the distance it places between itself and (La)TeX. In addition to LyX there are other GUIs, such as *ε:doc*[2] *or TeXMacs.*[3]

2 WYSIWYG versus WYSIWYM

A buzzword of the 1990s in software development was WYSIWYG (What You See is What You Get), which enabled the user to exert quick control over a document. On the one hand this requirement presented considerable problems for the software

[1] `http://www-cs-faculty.stanford.edu/~knuth/`
[2] `http://members.magnet.at/hfbuch/edoc/`
[3] `http://www.texmacs.org/`



LyX: ~/texte/Aufsatz/LyX/LyX-E120.lyx (Changed)

LyX has no formatting properties of its own, since it is built entirely on top of LaTeX. LyX can therefore only begin its work after a LaTeX document class has been chosen. The number of these classes is limited because they define only the most basic formatting properties. Some of the supported document classes are as follows:

* article
  * standard, \AMS , Komascript, IEEEtran, Springer, exam, Hebrew, dtk foot
    * The journal of the German \TUG
* book
  * standard, \AMS , Komascript
* report
  * standard, Komascript
* letter
  * standard, (g-Brief) english/french/german, Komascript
* seminar
* slide
* APA style (\acro{America ...)
* CD-Box Cover

The default document class in LyX is article. The user is also free to design completely new classes, taking advantage of the fact that LaTeX is an open system. [K12000, goosens94] Many universities have created their own classes for theses and dissertations. foot In order to make a new LaTeX document class for LyX available, a corresponding LyX layout must first be created. foot

6 Text parameters

As with every document preparation system, there are also some globally effective parameters in LaTeX applications, which can be set in the document GUI (Figure␣ Ref: fig:document).

float:figure

LyX: ~/texte/Aufsatz/LyX/LyX-E120.lyx (Changed)

Figure 1: The LYX Main-GUI

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x \Big(a^2 + \Big(a^2 + 2a\Delta x + (\Delta x)^2\Big) \\
+ \Big(a^2 + 2\cdot 2a\Delta x + 2^2\,(\Delta x)^2\Big) \\
+ \Big(a^2 + 2\cdot 3a\Delta x + 3^2\,(\Delta x)^2\Big) \\
+ \ldots \\
+ \Big(a^2 + 2\cdot(n-1)a\Delta x + (n-1)^2\,(\Delta x)^2\Big)\Big) \\
= \frac{1}{3}\,(b^3 - a^3)
\end{aligned}
$$

Figure 2: LYX formula mode

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x\Big(a^2 + \Big(a^2 + 2a\Delta x + (\Delta x)^2\Big) \\
+\Big(a^2 + 2\cdot 2a\Delta x + 2^2(\Delta x)^2\Big) \\
+\Big(a^2 + 2\cdot 3a\Delta x + 3^2(\Delta x)^2\Big) \\
+ \quad \ldots \\
+\Big(a^2 + 2\cdot(n-1)a\Delta x + (n-1)^2(\Delta x)^2\Big)\Big) \\
= \frac{1}{3}\Big(b^3 - a^3\Big)
\end{aligned}
$$

Figure 3: WinWord formula mode

developers. On the other hand, it was becoming clear that a substantial part of the layout of scientific and technical publications is concerned with for example german DIN-norms or publisher's defaults, and the author must be able to conform to these requirements. Therefore, in the development of LYX it was decided to use the WYSIWYM principle (What You See is What You Mean); that is, what the user sees in the graphical interface bears only a global resemblance to the final output. The degree of resemblance depends on the complexity of the text, lying anywhere between 30% . . . 95%, as demonstrated in figure 1. In order to avoid misunderstanding, we reiterate that WYSIWYM is a compromise; it does not promise to achieve the perfect combination of TEX and WYSIWYG.

3 Why LYX and not ???

There can be no convincing answer to this question without paying special attention to the typographical layout. For example, the formula mode in WinWord simply cannot be compared to the one in LATEX, as figures 2 and 3 demonstrate.

It is inserted as the PostScript-output of WinWord's math editor with the use of the default font. The essential difference lies in these fonts that are used; one must dedicate a much greater amount of attention to font selection under LATEX than under Windows, with its well-known TrueType-Fonts. SUN's software package StarOffice[4] is frequently recommended for use with Linux, since it is both free of charge and professionally built. In contrast to LATEX, StarOffice can serve as a true office package with word processing, spreadsheet, and database capabilities. LATEX, on the other hand, can do only one thing: document preparation. But it accomplishes this task better than any other software package.

Every word processor has its more or less known strengths and weaknesses, which we refrain from discussing here. The strengths of LATEX lie without a doubt in the realm of technical and scientific literature. Nowadays there are practically no barriers to the use of LATEX, thanks to the multitude of freely available software packages. The capability of a completely installed (LA)TEX-LYX package is in no way inferior to an installation of an office package, and furthermore, using LATEX-LYX results in great gains in speed when formatting large documents.

[4] `http://www.sun.com/products/staroffice/`

4 The LyX-Layout

Frequently, the largest problem for LyX beginners is the realization that the layout shown in the LyX window will not be identical to the final layout produced by LaTeX. In following the philosophy, LyX endeavors to achieve as many similarities to the final layout as possible. Each of the LyX layout commands translates into one or more LaTeX layout commands, but what appears in the LyX window is merely a rough approximation of the final product. For example, for the paragraph layout option called `Title`, the associated LyX commands are as follows:

```
# LyX Title-Layout
Style Title
  Margin  Static
  LatexType  Command
  InTitle  1
  LatexName  title
  ParSkip  0.4
  ItemSep  0
  TopSep  0
  BottomSep  1
  ParSep  1
  Align  Center
  AlignPossible Center
  LabelType  No_Label
# standard font definition
  Font
    Family  Sans
    Series  Bold
    Size  Largest
  EndFont
End  #title
```

The LaTeX Name `title` produces the direct connection with LaTeX, in that it determines what should happen with the text that is shown in the LyX window. It is also possible for the user to design a completely new layout; for example, the following LaTeX code is generated from a user-defined paragraph layout with a gray background. Internally, the LyX layout command will be named `myStandardColor`, and it will correspond to the user-defined LaTeX layout command `cminipage`.

```
# Standard color-style definition
Style myStandardColor
  Margin   Static
  LatexType  Environment
  LatexName  cminipage
  ParIndent MM
  ParSkip  0.4
  Align  Block
  AlignPossible  Block, Left, Right, Center
  LabelType  No_Label
# standard font definition
  Font
    color blue
    family typewriter
  EndFont
  Preamble
    \usepackage{color,calc}
    \definecolor{myColor}{rgb}{0.9,0.9,0.9}% rgb
    \newenvironment{cminipage}{%

     [ .... ]

     \end{lrbox}%
     \fcolorbox{myColor}{myColor}%
         {\usebox{\@tempboxa}}%
    }%
  EndPreamble
End # myStandardColor
```

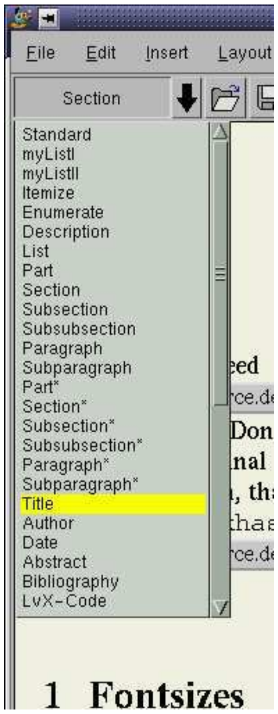


Figure 4: The LyX-Layout Menu

5 The Start

By today's standards, the LyX graphical interface appears somewhat spartan, but it contains all the essential elements that are for the most part present in all text processors. The buttons of the main LyX window can be modified to reflect the individual preferences of the user. Some additional icons are already provided in the standard system, and further ones may be created by the user and then made available to the LyX community.

LyX has no formatting properties of its own, since it is built entirely on top of LaTeX. LyX can therefore only begin its work after a LaTeX document class has been chosen. The number of these classes is limited because they define only the most basic formatting properties. Some of the document classes supported from LyX are as follows:

- article
  - standard, amsart,[5] cv, docbook (SGML), dtk,[6] egs,[7] ejour2,[8] elsart,[9] exam, Hebrew, IEEEtran, llncs,[10] Komascript, revtex, siamltex, ltugboat
- book
  - standard, amsart, docbook (SGML), Komascript
- report
  - standard, Broadway,[11] Komascript
- letter
  - standard, (g-Brief) english/french/german, Komascript
- seminar
- slide
- APA style (American Psychological Association)
- CD-Box Cover

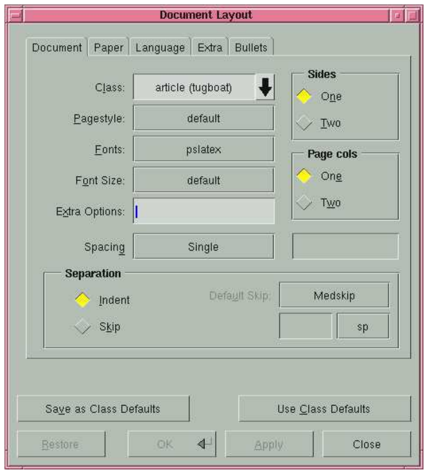
The default document class in LyX is article. The user is also free to design completely new LyX-Layouts, taking advantage of the fact that LyX is an open system. Many universities have created their own classes for theses and dissertations.[12] In order to make a new LaTeX document class available for LyX, a corresponding LyX layout must first be created.

6 Text parameters

As with every document preparation system, there are also some globally effective parameters in LaTeX applications. These can be set in the document GUI (figure 5), which holds a lot of different parameters.

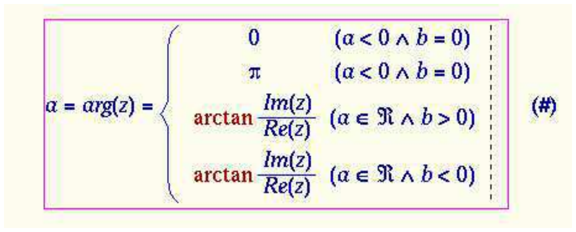
7 The Formula Generator

We have already mentioned the outstanding capability of TeX to produce mathematical formulas. LyX supports this capability in a graphical mode which almost achieves WYSIWYG quality, as seen in figure 6, which is the screenshot of the following equa-



Class: article (tugboat)
Pagestyle: default
Fonts: pslatex
Font Size: default
Extra Options:
Spacing: Single
Sides: [✓ One] Two
Page cols: [✓ One] Two
Separation: [✓ Indent] Skip
Default Skip: Medskip
sp

Figure 5: The Document GUI



$$a = arg(z) = \begin{cases} 0 & (a < 0 \wedge b = 0) \\ \pi & (a < 0 \wedge b = 0) \\ \arctan \frac{Im(z)}{Re(z)} & (a \in \Re \wedge b > 0) \\ \arctan \frac{Im(z)}{Re(z)} & (a \in \Re \wedge b < 0) \end{cases} \quad (\#)$$

Figure 6: LyX in Mathmode

tion:

$$a = arg(z) = \begin{cases} 0 & (a < 0 \wedge b = 0) \\ \pi & (a < 0 \wedge b = 0) \\ \arctan \frac{Im(z)}{Re(z)} & (a \in \Re \wedge b > 0) \\ \arctan \frac{Im(z)}{Re(z)} & (a \in \Re \wedge b < 0) \end{cases}$$

In contrast to the relatively easy-to-use math editor in WinWord, there exists in LaTeX considerably more possibilities for vertical and horizontal positioning. The WinWord math editor, though, has more menu choices than the LyX math panel, shown in figure 7.

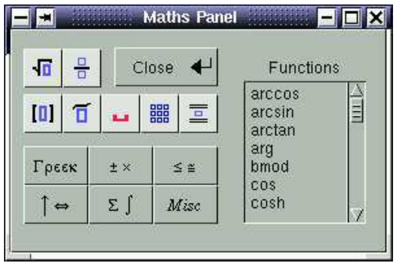


Figure 7: LyX Math-Panel

[5] American Mathematical Society
[6] The journal of the German TeX Users Group
[7] European Geophysical Society
[8] Journal of Geodesy
[9] Elsevier Journal
[10] Lecture Notes in Computer Science (Springer)
[11] Special class for writers
[12] Especially in the United States

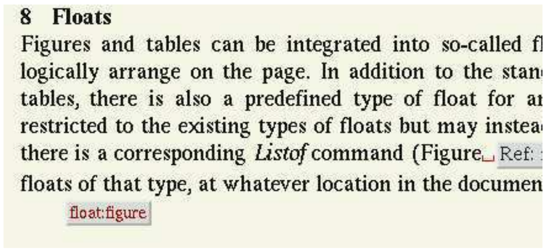

8 Floats
Figures and tables can be integrated into so-called fl
logically arrange on the page. In addition to the stan
tables, there is also a predefined type of float for an
restricted to the existing types of floats but may instea
there is a corresponding Listof command (Figure Ref:
floats of that type, at whatever location in the documen
float:figure

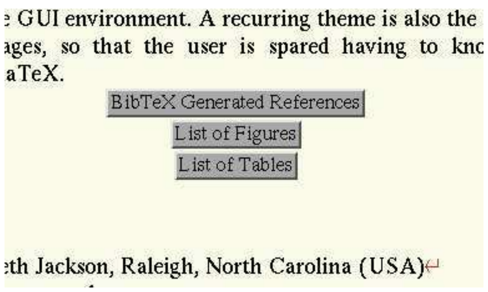
Figure 8: Figure-Float in a collapsed state

The American Mathematical Society's *amsmath*-package offers extensive functions within the context of mathematical formulas, which the upcoming version of LyX will support.[13]

8 Floats

Figures and tables can be integrated into so-called floating objects, or floats, that TeX can logically arrange on the page. In addition to the standard types of floats such as figures and tables, there is also a predefined type of float for an algorithm. In the LyX-Layout they appear in an open or collapsed layout (figure 8).

Of course, the user is not restricted to the existing types of floats but may instead create new ones. For all types of floats there are corresponding *Listof* commands (figure 9), each of which will produce a list of all floats of that type, at whatever location in the document the command is issued.


e GUI environment. A recurring theme is also the
ages, so that the user is spared having to knc
aTeX.

eth Jackson, Raleigh, North Carolina (USA)

Figure 9: Inserting List of Figures and List of Tables

9 Figures and Tables

Normally figures and tables are contained within floating objects, since their positioning is essential to the layout of the document. However, it is also possible to insert them directly into the text. All graphical objects must be in either PostScript (.ps) or encapsulated PostScript (.eps) format. LyX supports some graphic conversion via an external program like *convert* from the *imagemagic* package. Version 1.2 will have much better support for the graphics import.

[13] See also `http://www.perce.de/lyx/Equations.pdf`

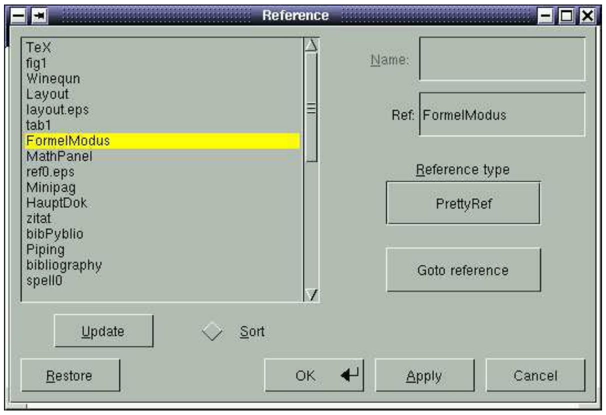


Figure 10: Menu to insert References

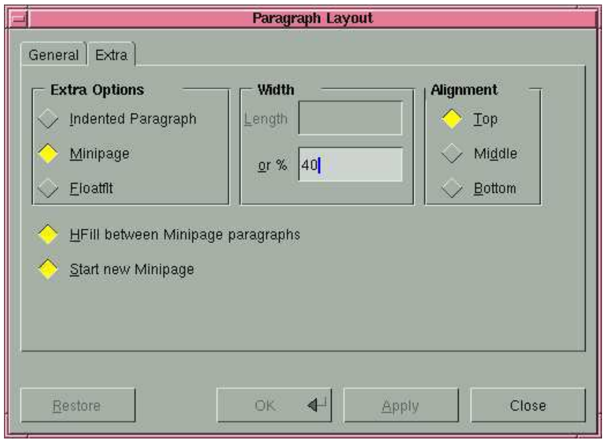
10 Cross-References

LaTeX provides several ways to include cross-references to a different location in a document, but since these methods are similar to those found in other document preparation packages, we mention them only briefly. The LaTeX package *hyperref* supports creating cross-references as a link into a pdf-file. This type of cross-reference must be placed in the LaTeX preamble (see section 17), since the current version of LyX only supports the *prettyref* package. A cross-reference using the *prettyref* package is created by selecting the type of reference from a menu of available references, as shown in figure 10. The upcoming LyX version 1.2 should contain a complete integration of the *hyperref* package.

A cross-reference must refer to a label that has been entered at some point in the document. The user can insert a label at any point in the document and choose any text for the name of the label. In a long document, the number of labels could easily exceed 100 or more; in this case, LyX provides a nice feature: it automatically adds the type of the cross-reference (for example, "sec" for section like `sec:minipages`) before the cross-reference in the text. LyX can display the labels in order of their appearance in the text or in alphabetical order.

11 Minipages

Within LaTeX, minipages are essential for handling difficult layout issues and are therefore also supported directly within LyX.



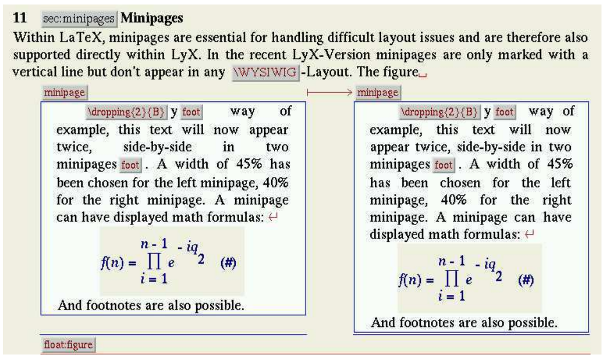
Extra Options
Indented Paragraph
[✓ Minipage]
Floatflt

Width
Length
or % 40

Alignment
[✓ Top]
Middle
Bottom

[✓ HFill between Minipage paragraphs]
[✓ Start new Minipage]

Figure 11: Paragraph layout for Minipages



11 sec:minipages **Minipages**
Within LaTeX, minipages are essential for handling difficult layout issues and are therefore also supported directly within LyX. In the recent LyX-Version minipages are only marked with a vertical line but don't appear in any \WYSIWIG -Layout. The figure

minipage

\dropping{2}{B} y foot way of example, this text will now appear twice, side-by-side in two minipages foot. A width of 45% has been chosen for the left minipage, 40% for the right minipage. A minipage can have displayed math formulas:

$$f(n) = \prod_{i=1}^{n-1} e^{-iq_2} \quad (\#)$$

And footnotes are also possible.

minipage

\dropping{2}{B} y foot way of example, this text will now appear twice, side-by-side in two minipages foot. A width of 45% has been chosen for the left minipage, 40% for the right minipage. A minipage can have displayed math formulas:

$$f(n) = \prod_{i=1}^{n-1} e^{-iq_2} \quad (\#)$$

And footnotes are also possible.

float:figure

Figure 12: Minipages in the upcoming version 1.2

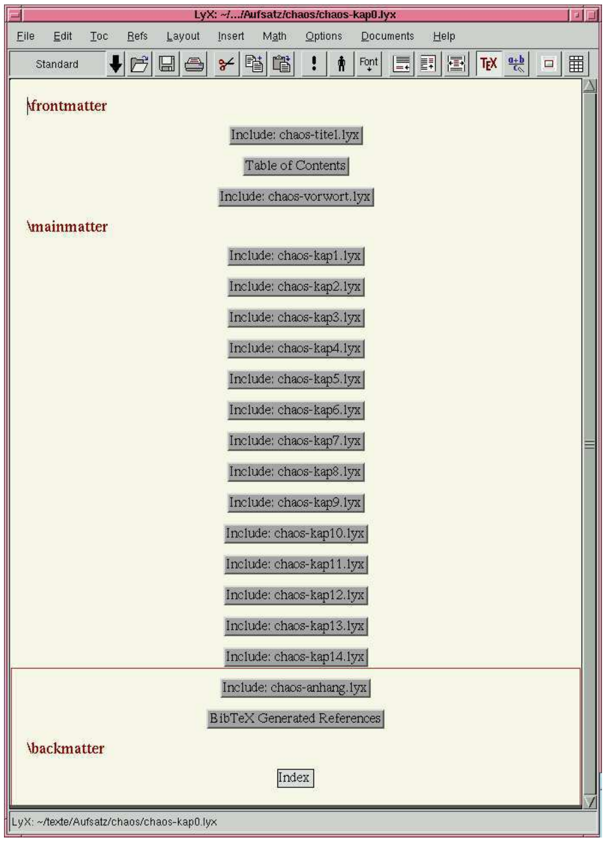
A minipage is set up using the menu found in Layout ▷ Paragraph format ▷ Extras, which is being changed for LYX version 1.2 to make certain options for minipages more straightforward and give LYX more support (figure 12). Figure 11 shows this menu as it appears in the current version of LYX.

12 Multi-Part Documents

When writing a book, there is a danger of losing the overview of the work, especially when more than a hundred pages must be organized. For this situation LYX offers the ability to break a long unwieldy work into separate, more manageable documents.

The structure of the main document might look like that shown in figure 13. It contains only the details of the chapters and sections (e.g., table of contents or index) that are to be inserted, as well as information about the manner in which the pages are to be numbered. It is not necessary for the main document to contain any text whatsoever.

LYX automatically opens all the required sub-documents, if they are not yet open, whenever the



\frontmatter
Include: chaos-titel.lyx
Table of Contents
Include: chaos-vorwort.lyx
\mainmatter
Include: chaos-kap1.lyx
Include: chaos-kap2.lyx
Include: chaos-kap3.lyx
Include: chaos-kap4.lyx
Include: chaos-kap5.lyx
Include: chaos-kap6.lyx
Include: chaos-kap7.lyx
Include: chaos-kap8.lyx
Include: chaos-kap9.lyx
Include: chaos-kap10.lyx
Include: chaos-kap11.lyx
Include: chaos-kap12.lyx
Include: chaos-kap13.lyx
Include: chaos-kap14.lyx
Include: chaos-anhang.lyx
BibTeX Generated References
\backmatter
Index

LyX: ~/texte/Aufsatz/chaos/chaos-kap0.lyx

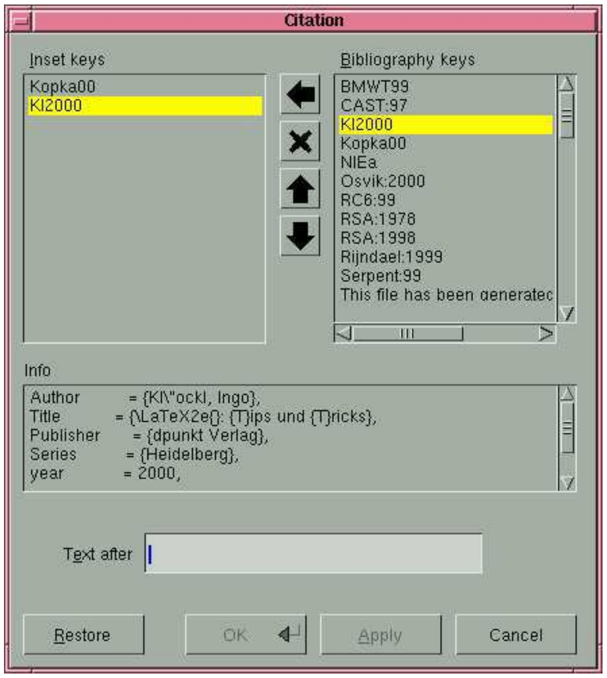
Figure 13: The main structure of a multipart document

main document is first translated (e.g., to create a pdf, PostScript, or dvi file) or when one sub-document attempts to cross-reference a location in another sub-document. LYX creates a common list of all labels from the various sub-documents to facilitate cross-references between the sub-documents.

13 Bibliography

LYX supervises the entire process of creating a bibliography: it runs the BIBTEX program when it is needed, without bothering the user with the details. The insertion of a literature citation is done by making a selection from a menu containing all the entries in the bibliographical database (figure 14), or by typing the keyword for this entry. The upcoming 1.2 version of LYX will give full *natbib*-support.

There are also several programs which make the maintaining of the database easier. In LYX, then, one only needs to choose from the database the correct entry from which LYX should make the citation. It is also possible to work with a separate citation-database program that supports input/output redirection ("piping"); Pybliographer,



Kopka00
KI2000

BMWT99
CAST:97
KI2000
Kopka00
NIEa
Osvik:2000
RC6:99
RSA:1978
RSA:1998
Rijndael:1999
Serpent:99
This file has been generatec

```
Author      = {Kl\"ockl, Ingo},
Title     = {\LaTeX2e{}: {T}ips und {T}ricks},
Publisher    = {dpunkt Verlag},
Series      = {Heidelberg},
year        = 2000,
```

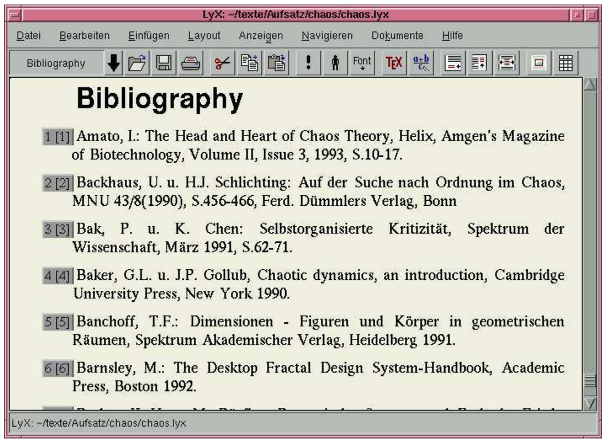
Figure 14: The citation-menu

for example, is such a program. “Piping” enables the program to directly insert a citation into the text through the so-called LyX server. This capability is found in several current BibTeX programs, such as TkBibTeX[14] and Sixpack.[15] Instead of using an external literature database, it is also possible to work with the bibliography facility that is directly supported within LyX. The citation keys must be entered within the document that is currently being edited, as shown in figure 15.

However, this procedure is not recommended, because these citations cannot be reused in the straightforward way that a database can be reused. Such citations can only be inserted into other documents using the “cut-and-paste” method, which requires that the user always have the name of the old document handy.

14 Spellcheck

Since LyX doesn’t include its own spellchecker, the user is free to employ any spellchecker that is started from the command line, such as the ispell program. The operation of the spellchecker is illustrated in figure 16; since there is nothing novel about LyX’s use of the spellchecker, we will not go into further detail.



Bibliography

1 [1] Amato, I.: The Head and Heart of Chaos Theory, Helix, Amgen's Magazine of Biotechnology, Volume II, Issue 3, 1993, S.10-17.

2 [2] Backhaus, U. u. H.J. Schlichting: Auf der Suche nach Ordnung im Chaos, MNU 43/8(1990), S.456-466, Ferd. Dümmlers Verlag, Bonn

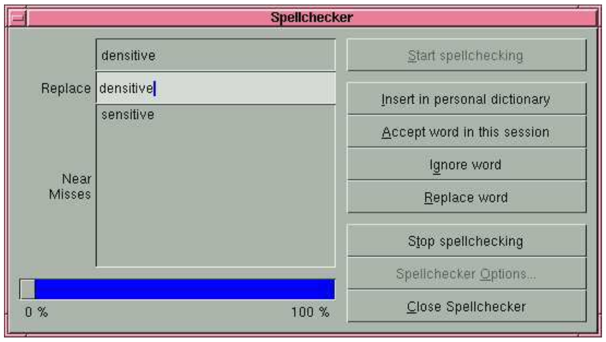
3 [3] Bak, P. u. K. Chen: Selbstorganisierte Kritizität, Spektrum der Wissenschaft, März 1991, S.62-71.

4 [4] Baker, G.L. u. J.P. Gollub, Chaotic dynamics, an introduction, Cambridge University Press, New York 1990.

5 [5] Banchoff, T.F.: Dimensionen - Figuren und Körper in geometrischen Räumen, Spektrum Akademischer Verlag, Heidelberg 1991.

6 [6] Barnsley, M.: The Desktop Fractal Design System-Handbook, Academic Press, Boston 1992.

Figure 15: The use of the bibliography layout



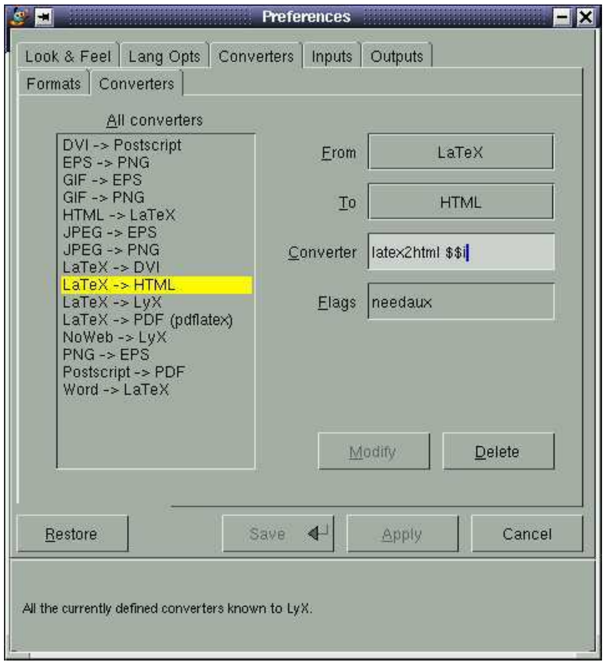
densitive
densitive
sensitive

Figure 16: Running the Spellchecker

15 Preview and Printing

The print preview process takes place completely outside of the LyX system, as well as outside the realm of LaTeX and TeX. LaTeX produces a data file in the dvi format, and the xdvi viewer can then be used to preview the document. One drawback of this process, however, is that if a document contains many errors, LaTeX will be unable to produce a dvi file at all, which in turn makes it more difficult for the user to locate the errors. As an alternative to the dvi output, LyX can also create a PostScript file which can be viewed with the *ghostview* program. This avenue is especially recommended for documents containing special effects such as colors or rotated figures and tables, since the dvi-viewer lacks the capability to display such things. If the option of printing to a file is chosen, then an external shell script can subsequently be used on the file to create a two-sided DIN-A5 output for a book.

[14] `http://www.cat.csiro.au/cmst/staff/pic/tkbibtex.html`
[15] `http://www.santafe.edu/~dirk/sixpack/`



All converters

DVI -> Postscript
EPS -> PNG
GIF -> EPS
GIF -> PNG
HTML -> LaTeX
JPEG -> EPS
JPEG -> PNG
LaTeX -> DVI
LaTeX -> HTML
LaTeX -> LyX
LaTeX -> PDF (pdflatex)
NoWeb -> LyX
PNG -> EPS
Postscript -> PDF
Word -> LaTeX

From	LaTeX
To	HTML
Converter	latex2html $$i
Flags	needaux

All the currently defined converters known to LyX.

Figure 17: Supported Import/Export formats

16 Importing and Exporting

LyX supports many import and export methods directly from the Files ▷ Import/Export menu, allowing the corresponding external program to be selected by the user. This is especially important for converting a document to HTML, since the programs for doing so differ greatly. Even though LyX has its own file format, it is currently not recognized by any known applications; therefore conversion from LyX to another format is possible only by first converting a LyX file to a TeX file. For this reason, in the list of predefined conversions shown in figure 17, which can be found under Edit ▷ Preferences ▷ Converter, LyX doesn't appear on the left-hand side of any of the conversion options.

Frequently, the conversion of a LyX document to PDF format is required. This conversion can be achieved using either the *ps2pdf* program or with the *pdftex* program; both possibilities exist within LyX. A frequently asked question is whether the capability exists to convert to and from Microsoft Word. There is no simple solution to this problem; in extreme cases, such as when complex formulas and extensive tables are present, the RTF format must be used. However, simple text can be successfully converted, both from RTF to TeX, and vice versa. LyX supports the *wvware* program, if it is installed.[16] The subsequent transfer from TeX to LyX can be accomplished through the *reLyX* import option.

LyX offers direct support of chess diagrams, bitmap graphics such as GIF, JPEG, and xfig files, which are converted with *convert* in the eps-format. In particular, the documentation of chess matches is greatly simplified.

17 The LaTeX Preamble

The LaTeX preamble constitutes the actual interface between LyX and LaTeX, because it contains all the (La)TeX-specific definitions. A large part of the LaTeX preamble could be included instead in the LyX text within TeX mode, but this action is not recommended because no insight is gained by doing so.

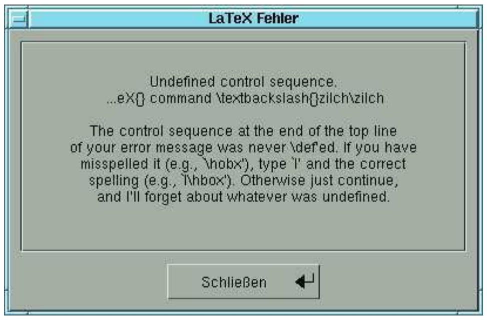
The LaTeX preamble is an integral part of every TeX file, although in theory it may be left empty if no modifications to the standard document class are desired. For LyX documents, though, at the very least the preamble contains the command describing how LyX should be printed, so that it doesn't appear as LYX. LyX itself tries to manage the preamble itself, so that the user puts only very special commands in the visible part of the preamble.

In principle the preamble is nothing more than a simple editor window in which the necessary additional packages are specified, those which aren't automatically loaded by LyX itself. The use of these simple editing functions is not one of LyX's strong points, but instead are vestiges of the past that must be lived with. One could even use the `\input` command to read in the entire preamble as an external file, and employ an external editor to modify it. A text with a complicated layout routinely will have a complicated preamble as a result.

18 Error Analysis

LyX runs LaTeX always in batchmode and tries its best to locate any errors that are generated during this process, marking them with an error flag in the spot in the document where they occur. Clicking on the error flag will generate a window (shown in figure 18) detailing the condition that caused the error, enabling the user to analyze and then eliminate the problem. These LaTeX error messages are sometimes very general, so that LyX has problems locating the error and it can only do its best.

[16] `http://www.wvware.com`



Undefined control sequence.
...eX{} command \textbackslash{}zilch\zilch

The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

Figure 18: Error message

A LaTeX run is similar to the compilation of source code in a programming language; a single error in the code may result in a very large number of error messages. In longer documents, the number of errors can easily reach one hundred or more.

19 Navigation

Navigating within an extensive document can be a challenge, so LyX offers help in the form of a document-specific menu. This menu contains the names of all section and subsections, as well as tables and figures, enabling the user to jump to any point in the text with relative ease. Figure 19 shows all sections and all figures of this text, which are all like internal links, so that jumping to any place in the text is easy.

Another navigational tool exists under Edit ▷ Find and Replace, which operates in the same manner as in other text editors. In the current version of LyX, the “Find and Replace” utility works only on text that appears on the screen; unfortunately it does not “find and replace” command sequences.

20 Documentation

A large problem with open source projects is always the documentation and help, which operate within mailing lists. The volunteers who maintain the documentation are frequently unable to keep up with the frantic pace of the developers, so that the documentation lags behind the currently available version. Furthermore, users will often skip the documentation altogether, preferring instead to send a question directly to the mailing list, a question that most likely has already been asked and answered many times over. Under the LyX menu option Help there is a selection of help documents that are themselves LyX files, which may easily be printed. Although the help files cover a considerable amount of material, they suffer from a somewhat uninspired

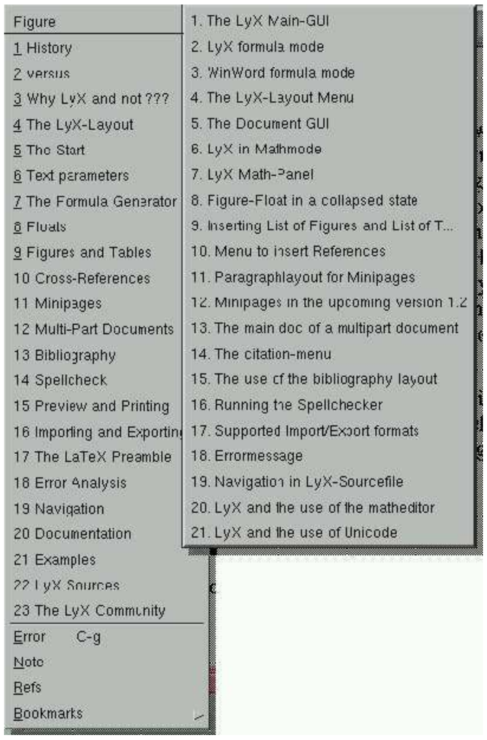


Figure 19: Navigation in a LyX-Sourcefile

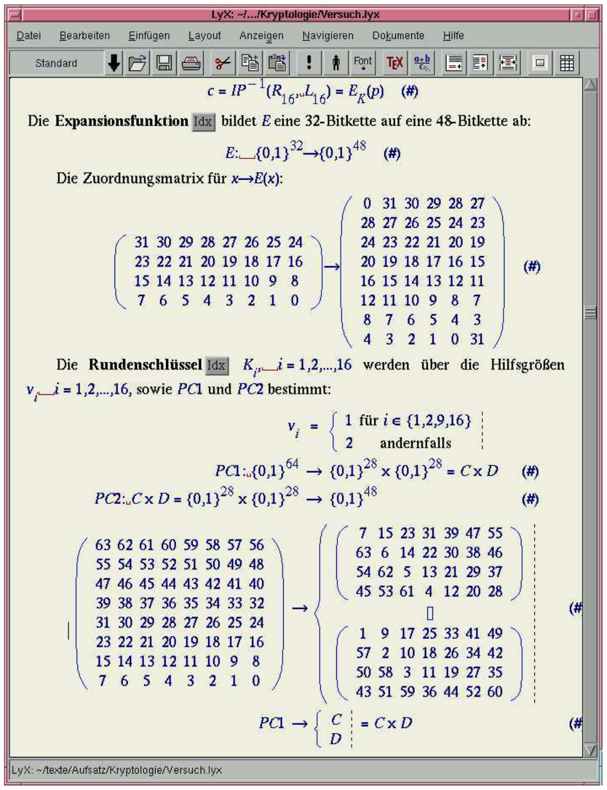
organization; for this reason users frequently have problems finding the answer to a question.

The current version of LyX (1.1.6) contains substantial differences from previous versions, yet in a few places the documentation still needs to be brought up to date. This task is being put on hold for the time being, though, while most of the documentation effort is going towards the preparation of upcoming version 1.2.

21 Examples

Now that we have completed our whirlwind tour through the basic features of LyX, we present a few examples that will demonstrate the merits of using a typesetting system like TeX in conjunction with a document processor like LyX. The first example (figure 20) shows the extensive use of LyX’s nearly WYSIWYG-Matheditor.

The next example (figure 21) shows the use of Unicode for Hebrew, Arabic and Russian. LyX also supports Right-To-Left-written languages; however, for some special issues, like search and replace, there are restrictions. More Information about this special topic of LyX is available at `http://www.math.tau.ac.il/~dekelts/lyx/`.


LyX: ~/.../Kryptologie/Versuch.lyx
Datei Bearbeiten Einfügen Layout Anzeigen Navigieren Dokumente Hilfe
Standard
Font
TEX
LyX: ~/texte/Aufsatz/Kryptologie/Versuch.lyx

$$c = IP^{-1}(R_{16}, L_{16}) = E_K(p) \qquad (\#)$$

Die **Expansionsfunktion** Idx bildet E eine 32-Bitkette auf eine 48-Bitkette ab:

$$E:\ \{0,1\}^{32} \to \{0,1\}^{48} \qquad (\#)$$

Die Zuordnungsmatrix für $x \to E(x)$:

$$\begin{pmatrix} 31 & 30 & 29 & 28 & 27 & 26 & 25 & 24 \\ 23 & 22 & 21 & 20 & 19 & 18 & 17 & 16 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 31 & 30 & 29 & 28 & 27 \\ 28 & 27 & 26 & 25 & 24 & 23 \\ 24 & 23 & 22 & 21 & 20 & 19 \\ 20 & 19 & 18 & 17 & 16 & 15 \\ 16 & 15 & 14 & 13 & 12 & 11 \\ 12 & 11 & 10 & 9 & 8 & 7 \\ 8 & 7 & 6 & 5 & 4 & 3 \\ 4 & 3 & 2 & 1 & 0 & 31 \end{pmatrix} \qquad (\#)$$

Die **Rundenschlüssel** Idx $K_i,\ i = 1,2,\ldots,16$ werden über die Hilfsgrößen $v_i,\ i = 1,2,\ldots,16$, sowie $PC1$ und $PC2$ bestimmt:

$$v_i = \begin{cases} 1 & \text{für } i \in \{1,2,9,16\} \\ 2 & \text{andernfalls} \end{cases}$$

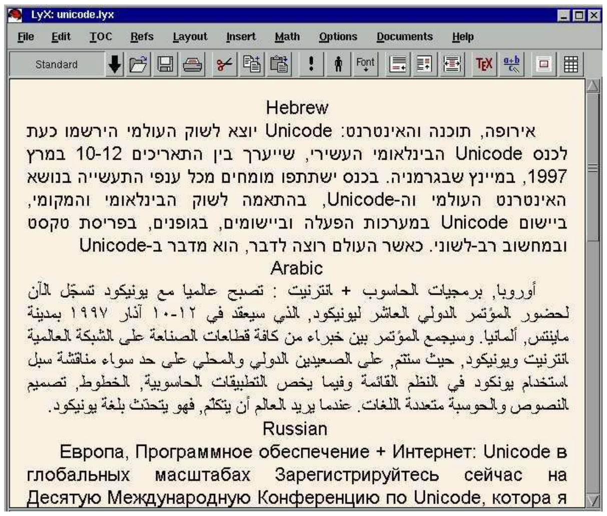
$$PC1:\ \{0,1\}^{64} \to \{0,1\}^{28} \times \{0,1\}^{28} = C \times D \qquad (\#)$$

$$PC2:\ C \times D = \{0,1\}^{28} \times \{0,1\}^{28} \to \{0,1\}^{48} \qquad (\#)$$

$$\begin{pmatrix} 63 & 62 & 61 & 60 & 59 & 58 & 57 & 56 \\ 55 & 54 & 53 & 52 & 51 & 50 & 49 & 48 \\ 47 & 46 & 45 & 44 & 43 & 42 & 41 & 40 \\ 39 & 38 & 37 & 36 & 35 & 34 & 33 & 32 \\ 31 & 30 & 29 & 28 & 27 & 26 & 25 & 24 \\ 23 & 22 & 21 & 20 & 19 & 18 & 17 & 16 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix} \to \begin{cases} \begin{pmatrix} 7 & 15 & 23 & 31 & 39 & 47 & 55 \\ 63 & 6 & 14 & 22 & 30 & 38 & 46 \\ 54 & 62 & 5 & 13 & 21 & 29 & 37 \\ 45 & 53 & 61 & 4 & 12 & 20 & 28 \end{pmatrix} \\ \square \\ \begin{pmatrix} 1 & 9 & 17 & 25 & 33 & 41 & 49 \\ 57 & 2 & 10 & 18 & 26 & 34 & 42 \\ 50 & 58 & 3 & 11 & 19 & 27 & 35 \\ 43 & 51 & 59 & 36 & 44 & 52 & 60 \end{pmatrix} \end{cases} \qquad (\#$$

$$PC1 \to \begin{cases} C \\ D \end{cases} = C \times D \qquad (\#$$

Figure 20: LYX and the use of the matheditor


LyX: unicode.lyx
File Edit TOC Refs Layout Insert Math Options Documents Help
Standard
Font
TEX

Hebrew

אירופה, תוכנה והאינטרנט: Unicode יוצא לשוק העולמי הירשמו כעת לכנס Unicode הבינלאומי העשירי, שייערך בין התאריכים 10-12 במרץ 1997, במיינץ שבגרמניה. בכנס ישתתפו מומחים מכל ענפי התעשייה בנושא האינטרנט העולמי וה-Unicode, בהתאמה לשוק הבינלאומי והמקומי, ביישום Unicode במערכות הפעלה וביישומים, בגופנים, בפריסת טקסט ובמחשוב רב-לשוני. כאשר העולם רוצה לדבר, הוא מדבר ב-Unicode

Arabic

أوروبا, برمجيات الحاسوب + انترنيت : تصبح عالميا مع يونيكود تسجّل الآن لحضور المؤتمر الدولي العاشر ليونيكود, الذي سيعقد في ١٠-١٢ آذار ١٩٩٧ بمدينة ماينتس, ألمانيا. وسيجمع المؤتمر بين خبراء من كافة قطاعات الصناعة على الشبكة العالمية انترنيت ويونيكود, حيث ستتم, على الصعيدين الدولي والمحلي على حد سواء مناقشة سبل استخدام يونكود في النظم القائمة وفيما يخص التطبيقات الحاسوبية, الخطوط, تصميم النصوص والحوسبة متعددة اللغات. عندما يريد العالم أن يتكلم, فهو يتحدث بلغة يونيكود.

Russian

Европа, Программное обеспечение + Интернет: Unicode в глобальных масштабах Зарегистрируйтесь сейчас на Десятую Международную Конференцию по Unicode, котора я

Figure 21: LYX and the use of Unicode

22 LYX Sources

The web site `http://www.lyx.org` contains all the essential information for installing LYX on all current operating systems. For OS/2 and Windows machines, an X-server is also needed. All systems must have a fully installed TEX system, which can be freely downloaded from `http://www.ctan.org` or any mirror. Further download sources can be found at `http://www.lyx.org/help/`, in particular for special versions of LYX, like the one for Solaris or Debian.

23 The LYX Community

As an open-source project, LYX must rely on the intensive collaborative efforts of users who are in constant communication about the implementation and development of LYX. The LYX homepage, as mentioned before, is at `http://www.lyx.org`. In addition, there are mailing lists for:

Users: `lyx-users@lists.lyx.org`
(medium volume)

Developers: `lyx-devel@lists.lyx.org`
(medium to high volume)

Documentation team: `lyx-doc@lists.lyx.org`
(low volume)

Directions on how to register for one of these mailing lists can be found on the LYX homepage. An extensive website with tips and tricks for LYX is located at `http://www.lyx.org/help/`. For all mailing lists there are online archives located at `http://www.mail-archive.com/lyx-???@lists.lyx.org/`, where "???" should be substituted with "devel", "user", or "doc".

A mid-term goal of LYX development is for LYX to have the capability to run in every possible GUI environment. A recurring theme is also the integration of more LATEX packages, so that the user is spared having to know about the inner workings of LATEX.

◇ Laura Elizabeth Jackson
Raleigh, North Carolina (USA)
`lejacks2@unity.ncsu.edu`

◇ Herbert Voß
Berlin (Germany)
`voss@lyx.org`

DVII: A TeX dvi file information utility

Adam H. Lewenberg

Abstract

dvii is a free utility written in portable C that extracts information from a TeX dvi file and displays it on the command line. The information can include paging, fonts, specials, and per-page message digests. The output is designed to be easily parsed by a text-processing language (such as Perl) to allow other kinds of summaries to be generated (such as a font difference utility, or to help dvips print only those pages containing certain kinds of \special's).

– – ∗ – –

1 Introduction

Consider a dvi file. Perhaps you created it by compiling a TeX, or maybe you received it in your e-mail. Here are some questions about the dvi file:

1. How many pages does it contain?
2. How do the "physical" pages correspond to the "TeX" pages? (For example, if the file is printed, what page number will be printed at the bottom of the seventh page out of the printer; it will not necessarily be 7.)
3. What fonts are called for? Which fonts are called for *on a specific page*?
4. Which pages contain (externally linked) figures?
5. Have the page breaks changed since the last time I compiled?
6. Has this file been corrupted?
7. When was this file compiled?[1]
8. Does this file use the same fonts as some other file?

The dvii program is a free utility that extracts and displays information from a TeX dvi file that allows us to answer all of these questions quickly and easily. For information on where to download the dvii utility, see section 10.

2 An Example

If we run the dvii utility on the file test.dvi we get the following output:

```
File size: 1188 bytes (1 K)
Comment string:  TeX output 2001.12.29:2041
Page count: 7
Number of fonts: 3
f:[50/cmr10/1200]::4bf16079
f:[23/cmbx10/1000]::1af22256
f:[0/cmr10/1000]::4bf16079
p:[1/1]
p:[2/2]
p:[3/3]
p:[4/4]
p:[5/5]
p:[6/-1]
p:[7/-3]
s:[3/3]:: A short special
s:[5/5]:: PSfile 1.eps
s:[5/5]:: PSfile 2.eps
s:[5/5]:: PSfile 3.EPS
s:[5/5]:: PSfile dog1.gif
s:[5/5]:: PSfile cat.eps
```

(To the see the source file test.tex see Appendix A.)

This output tells us several things. First of all, there is a summary of the dvi file including the file size, the comment string, and the number of fonts and pages in the dvi file. Next, there is more detailed information listing the fonts used, the pages (both the "physical" page and the TeX page), and the TeX specials.

3 Purpose and Motivation

Much of what dvii does could be done with Donald Knuth's dvitype and a text processing language like Perl. My motivation was not to make another dvi parser, but to create a utility that would extract specific sorts of information quickly from a dvi file.

I wrote dvii with the following goals in mind:

1. It should be *fast*, faster than dvitype.
2. It should be easy to use the output as a back end to Perl (or any other text processing language), enabling the easy manipulation of the data for more specific purposes.
3. It should be useful to the TeX community.
4. It should be portable.

I believe I have met most[2] of these goals, but if anyone has suggestions for improving this utility, I will be quite happy to hear them.

4 Consistency

You can check that a dvi file has not been corrupted by using the -C option.

```
dvii -C test.dvi

dvi file 'test.dvi' passed validation
check (level 1).
```

Here is an example where dvii detects a problem:

```
dvii -C bad.dvi

[dvi validation error] missing postamble id
(should be 2)
```

[1] This is not necessarily the same as the file date.

[2] However, it is hard to beat Knuth in program efficiency and speed!

5 Pages

You want to know the number of pages in a `dvi` file. How do you determine this? You could view it with a `dvi` viewer, or perhaps look at the `.log` file that was generated when the `dvi` file was originally created. But a `dvi` viewer is rather heavy machinery to simply find the number of pages in a `dvi` file, and, besides, you may be working from the command line where there is no `dvi` viewer handy. As for the `.log` file, it may no longer be around.

The dvii utility provides a page count virtually instantaneously. Furthermore, if you want to know the page numbering layout of physical vs. TEX pages, dvii will also tell you that. (By a "physical" page I mean the order of the page as printed, and by TEX page I mean the page as it appears printed in the footer or running head. For example, the third page out of the printer might have TEX page number −3, that is, iii.)[3]

From the example There are seven physical pages where the first five have TEX page number matching the physical page number. Physical pages 6 and 7 have TEX pages −1 and −3.

6 Fonts

One of the reasons that `dvi` files are not very portable is that the fonts that a `dvi` file uses are not embedded in the `dvi` file itself, but rather are identified by number and name. TEX leaves it to the `dvi` interpreter to find the proper external font. Thus, if you receive a `dvi` file via e-mail or download one from the web, there is a good chance it will not look the way the author intended or perhaps it will not display at all, unless you have the same fonts with the same names as the `dvi` file's author. So, a list of fonts that a `dvi` file calls for is, at times, quite useful.

The dvii utility will list each font used in a `dvi` file listing its name, font number, scale factor, and checksum. The checksum is especially useful in detecting when two fonts that seem to be the same (same name and scale) are in fact different.

From the example There are three fonts, two copies of `cmr10` (Computer Modern Roman 10pt) scaled at 100% (font number 0) and 120% (font number 50), and `cmbx10` (Computer Modern BoldFace 10pt) scaled at 100% (font number 23).

[3] More accurately, by TEX page I mean the contents of the `\count0` register which (normally) stores the page number that is printed on the page itself.

7 Specials

The TEX special is a "hook" that D. Knuth put into TEX to allow later functionality without having to rewrite TEX. To insert a special you use the `\special{}` command. For all practical purposes, TEX ignores specials,[4] merely copying the special text to the `dvi` output. It is up to the `dvi` driver to decide what meaning (if any) to give to a special.

Some uses that people have made of specials are to incorporate color, to help with the edit-compose-view cycle, and, most commonly, to allow external figure file inclusion, in particular the inclusion of EPS files. The graphics and graphicx packages along with most dvi-to-PostScript drivers indicate the inclusion of an EPS file by inserting a special that starts with the string `PSfile` and then is followed by a number of arguments including the file name and the dimensions of the figure.

The dvii utility will list all specials in a `dvi` file. If you want to list all specials that match some particular string, you can pipe the output of dvii through the standard `grep` utility.

From the example There are six specials, one on page 3, and five on page 5. The specials on page 5 appear to be included EPS files with the file name specified after the string `PSfile`.

8 Message Hashes

At this point, I have shown how to answer all the questions asked in section 1 except questions 5 and 8. For an answer to question 8 see section 9.1. In this section I will take up question 5.

The trickiest part of setting type is almost always page optimization, that is, fixing bad page breaks and getting floats (figures, tables, etc.) put in the right places. This part of composition is more an art than a science although TEX and LATEX do provide tools to help. This is why you always, **always**[5] wait until the last possible second before doing final page layout.

But, inevitably, after you have spent a week getting all your page breaks and floats in your 800 page book set just where you want them, someone comes along and insists on making some small change that has the potential of messing everything up. Wouldn't it be nice to have a quick and easy method to see how a change to the text affects the page layout for the entire document?

My solution to this is to take a numerical "snapshot" of each page before and after the change and

[4] Well, almost. It is possible for a special to affect page breaks.

[5] Always!

then see which pages' snapshots have changed. This snapshot is called a *message digest* or *checksum*. dvii can, if asked, calculate a 16-byte checksum for each page. This checksum will change if the contents of the page change. Thus, if you calculate the message digest before and after a change to the TeX source, you can see which pages have changed and thus where to look for possible new bad page breaks.

From the example Here is the message digest for `test.dvi`:

```
dvii -m test.dvi
[message digest: simple sum]
p:[1/1]::D8C977816A091771A3A631E7582DAD6D
p:[2/2]::284E8575505581BAA95B7CB132F1F435
p:[3/3]::16E5A31FF87F926DB1F86CAD165C5453
p:[4/4]::C72EE84EFA36764C537229D4968F8DF9
p:[5/5]::68E5A2E9D7320743CB85769CAAEAB023
p:[6/-1]::DE72BC3345FFDF7E779C8C667DCE3F97
p:[7/-3]::13B730AA855EB18E30CED11AB1D26FAF
```

Let `test2.tex` be an exact copy of `test.tex` except that we have changed the first word “This” to “Thus” (see source listing in Appendix A). Here is the resulting message digest.

```
dvii -m test2
[message digest: simple sum]
p:[1/1]::556A2538928963D8B5776E1245364DE6
p:[2/2]::284E8575505581BAA95B7CB132F1F435
p:[3/3]::16E5A31FF87F926DB1F86CAD165C5453
p:[4/4]::C72EE84EFA36764C537229D4968F8DF9
p:[5/5]::68E5A2E9D7320743CB85769CAAEAB023
p:[6/-1]::DE72BC3345FFDF7E779C8C667DCE3F97
p:[7/-3]::13B730AA855EB18E30CED11AB1D26FAF
```

If you look carefully, you will notice that the checksum for the first page has changed while the others have not.

For a large project you would not want to try to detect such changes by eye, so you would instead use a utility such as `diff` to detect the differences.

```
dvii -m test > test.md
dvii -m test2 > test2.md
diff test.md test2.md
2c2
< p:[1/1]::D8C977816A091771A3A631E7582DAD6D
---
> p:[1/1]::556A2538928963D8B5776E1245364DE6
```

The first two commands store the message digests in the files `test.md` and `test2.md`. The third command uses the `diff` command to find how the two files `test.md` and `test2.md` differ; in this case, the `diff` command shows us that they differ only in the first line.

9 Some applications based on dvii

The output of dvii has been designed to make it easy for text processing programs to manipulate and provide further useful information. Because of its near ubiquitousness, ease of use, and low cost (free), I use Perl. If Unix is your computing environment you probably already have Perl installed. If you use Windows, then you can download a free version. See section 10 for more information on where to get Perl.

In what follows I describe two such scripts based on Perl.[6]

9.1 `fontdiff.pl`

The Perl script `fontdiff.pl` finds the font differences between two `dvi` files, that is, it lists which fonts are present in one and not the other. Here is an example using the two `dvi` files `a.dvi` and `b.dvi`:

```
perl fontdiff.pl -l a b

Fonts in a.dvi NOT in b.dvi:
NOTE: fonts marked with * are in BOTH files
------------------------------
  f:[NN/cmbx10/1000]::1af22256
* f:[NN/cmr10/1000]::4bf16079
------------------------------


Fonts in b.dvi NOT in a.dvi:
NOTE: fonts marked with * are in BOTH files
------------------------------
  f:[NN/cmti10/1000]::fd00273a
  f:[NN/cmsl10/1000]::70ae304a
* f:[NN/cmr10/1000]::4bf16079
------------------------------
```

9.2 `specials.pl`

This Perl script creates the command-line option that works with Tom Rockicki's `dvips` so that just those pages of a `dvi` file containing specials get printed. Recall from Section 2, the `dvi` file `test.dvi` has `\special`'s on pages 3 and 5. To print just those pages using `dvips` you would type `dvips -pp3,5 test`. If you run `specials.pl` on the file `test.dvi` we get

```
perl specials.pl test
-pp3,5
```

Although this example is short, you can see how useful it would be to generate this page list automatically if your `dvi` file had hundreds of pages and dozens of figures.

Observe that in the above example page 3 was listed even though page 3 does not have a *figure* special. To list only those pages that have a `\special` matching some string, you can use the `--grep` option. For example, most TeX and LaTeX graphics inclusion packages indicate an included figure by starting the `\special` with the string `PSfile`, so to list

[6] Both Perl scripts, as well as more information on their use, are available at the dvii home page; see Section 10.

only those pages which have a special that contain the string `PSfile` you would type

```
perl specials.pl --grep PSfile test
-pp5
```

10 Where and how to get it

The dvii utility has its home page at `http://www.macrotex.net/dvii/dvii.html`. At this site you can download the C source code which consists of a single file and should compile with any standard C compiler. No special libraries are required. There are no licensing restrictions on the use of this utility.

If you do not wish to, or are unable to compile the source code, there are binaries for Windows, Linux, Solaris, and DOS.[7] To install, download the appropriate executable file and put it somewhere in your path.

You can also download a manual for dvii which explains in detail all the options. This manual is available in HTML, `dvi`, and PDF formats.

If you prefer, you can download from CTAN the source code, manual (in PDF), and a DOS/Windows executable which you will find in the `dviware/dvii` directory.

Perl is available on nearly every computing platform for no cost. If you work on a Unix or Unix-like platform Perl is probably already installed. For more information on obtaining Perl, go to the Comprehensive Perl Archive Network (CPAN) at `www.cpan.org`.

The `diff` and `grep` utilities are even more likely to be already available on Unix and Unix-like systems. If they are not, you can get GNU versions at `www.gnu.org`. If you run Windows, you can get them at the Cygwin home at `http://sourceware.cygnus.com/cygwin/`.

11 Acknowledgements

Heiko Oberdiek made significant suggestions on improving the performance of the code, and pointed out several errors. Tom Kacvinsky helped make the code work on 64-bit machines.

[7] If you can compile the code for other platforms, I would be happy to post them there. I am especially looking for a Macintosh version (does Macintosh *have* a command line?).

A The source for `test.tex`

```
% This is test.tex
This is page 1/1.
\eject
This is page 2/2.
\eject
This is page 3/3 with a short special.
Also, a rule.
\vrule width1cm depth1cm height 1cm\relax
\special{A short special}
\eject
This is page 4/4 {\bf without} any specials.
\eject
This is page 5/5 with 5 specials.
\special{PSfile 1.eps}
\special{PSfile 2.eps}
\special{PSfile 3.EPS}
\special{PSfile dog1.gif}
\special{PSfile cat.eps}
\eject
\pageno = -1
\font\a=cmr10 scaled 1200
{\a This is page 6/-1 with font cmr10
scaled 1200.}
\eject
\pageno = -3
This page is nothing special.
\eject\eject\eject\eject
\bye
```

References

[1] Donald E. Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, October 1990.

◇ Adam H. Lewenberg
211 Paddock Drive East
Savoy, Illinois 61874
adam@macrotex.net

Graphics

Drawing Graphs with MetaPost

John D. Hobby

Abstract

This paper describes a graph-drawing package that has been implemented as an extension to the MetaPost graphics language. MetaPost has a powerful macro facility for implementing such extensions. There are also some new language features that support the graph macros. Existing features for generating and manipulating pictures allow the user to do things that would be difficult to achieve in a stand-alone graph package.

1 Introduction

MetaPost is a batch-oriented graphics language based on Knuth's METAFONT, but with PostScript output and numerous features for integrating text and graphics. The author has tried to make this paper as independent as possible of the user's manual [6], but fully appreciating all the material requires some knowledge of the MetaPost language.

We concentrate on the mechanics of producing particular kinds of graphs because the question of what type of graph is best in a given situation is covered elsewhere; e.g., Cleveland [2, 4, 3] and Tufte [11]. The goal is to provide at least the power of UNIX *grap* [1], but within the MetaPost language. Hence the package is implemented using MetaPost's powerful macro facility.

The graph macros provide the following functionality:

1. Automatic scaling
2. Automatic generation and labeling of tick marks or grid lines
3. Multiple coordinate systems
4. Linear and logarithmic scales
5. Separate data files
6. Ability to handle numbers outside the usual range
7. Arbitrary plotting symbols
8. Drawing, filling, and labeling commands for graphs

In addition to these items, the user also has access to all the features described in the MetaPost user's manual [6]. These include access to almost all the features of PostScript, ability to use and manipulate typeset text, ability to solve linear equations, and data types for points, curves, pictures, and coordinate transformations.

Section 2 describes the graph macros from a user's perspective and presents several examples. Sections 3 and 4 discuss auxiliary packages for manipulating and typesetting numbers and Section 5 gives some concluding remarks. Appendix A summarizes the graph-drawing macros, and Appendix B describes some recent additions to the MetaPost language that have not been presented elsewhere.

2 Using the Graph Macros

A MetaPost input file that uses the graph macros should begin with

```
input graph
```

This reads a macro file `graph.mp` and defines the graph-drawing commands explained below. The rest of the file should be one or more instances of

```
beginfig(⟨figure number⟩);
⟨graphics commands⟩ endfig;
```

followed by `end`.

The following ⟨graphics commands⟩ suffice to generate the graph in Figure 1 from the data file `agepop91.d`:

```
draw begingraph(3in,2in);
  gdraw "agepop91.d";
  endgraph;
```

(Each line of `agepop91.d` gives an age followed the estimated number of Americans of that age in 1991 [10].)

2.1 Basic Graph-Drawing Commands

All graphs should begin with

```
begingraph(⟨width⟩,⟨height⟩);
```

and end with `endgraph`. This is syntactically a ⟨picture expression⟩, so it should be preceded by `draw` and followed by a semicolon as in the example.[1] The ⟨width⟩ and ⟨height⟩ give the dimensions of the graph itself without the axis labels.

The command

```
gdraw ⟨expression⟩ ⟨option list⟩
```

draws a graph line. If the ⟨expression⟩ is of type string, it names a data file; otherwise it is a path that gives the function to draw. The ⟨option list⟩ is zero or more drawing options

```
withpen ⟨pen expression⟩
   | withcolor ⟨color expression⟩
   | dashed ⟨picture expression⟩
```

[1] See the user's manual [6] for explanations of `draw` commands and syntactic elements like ⟨picture expression⟩.

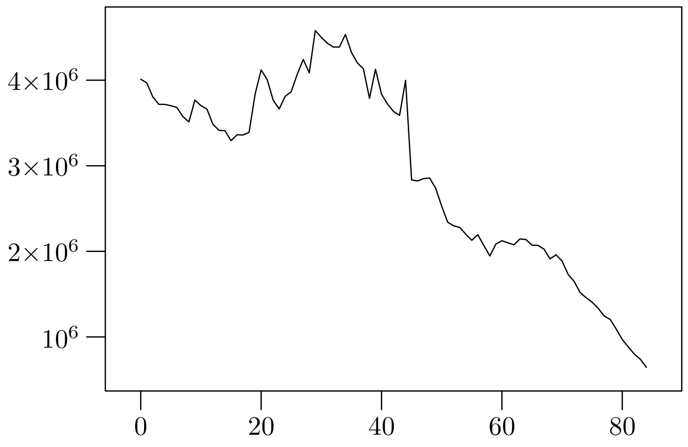


Figure 1: A graph of the 1991 age distribution in the United States

that give the line width, color, or dash pattern as explained in the User's Manual [6].

In addition to the standard drawing options, the ⟨option list⟩ in a `gdraw` statement can contain

`plot` ⟨picture expression⟩

The ⟨picture expression⟩ gives a plotting symbol to be drawn at each path knot. The `plot` option suppresses line drawing so that[2]

`gdraw "agepop91.d" plot btex` • `etex`

generates only bullets as shown in Figure 2. (Following the `plot` option with a `withpen` option would cause the line to reappear superimposed on the plotting symbols.)

The `glabel` and `gdotlabel` commands add labels to a graph. The syntax for `glabel` is

`glabel.`⟨label suffix⟩
  `(`⟨string or picture expression⟩`,` ⟨location⟩`)`
  ⟨option list⟩

where ⟨location⟩ identifies the location being labeled and ⟨label suffix⟩ tells how the label is offset relative to that location. The `gdotlabel` command is identical, except it marks the location with a dot. A ⟨label suffix⟩ is as in plain MetaPost: ⟨empty⟩ centers the label on the location; `lft`, `rt`, `top`, `bot` offset the label horizontally or vertically; and `ulft`, `urt`, `llft`, `lrt` give diagonal offsets. The ⟨location⟩ can be a pair of graph coordinates, a knot number on the last `gdraw` path, or the special location `OUT`. Thus

`gdotlabel.top(btex` $(50,0)$ `etex, 50,0)`

would put a dot at graph coordinates `(50,0)` and place the typeset text "$(50,0)$" above it. Alternatively,

```
glabel.ulft("Knot3", 3)
```

typesets the string `"Knot3"` and places it above and to the left of Knot 3 of the last `gdraw` path. (The knot number 3 is the path's "time" parameter [6, Section 8.2].)

The ⟨location⟩ `OUT` places a label relative to the whole graph. For example, replacing "`gdraw "agepop91.d"`" with

```
glabel.lft(btex \vbox{\hbox{Population}
  \hbox{in millions}} etex, OUT);
glabel.bot(btex Age in years etex, OUT)
gdraw "agepopm.d";
```

in the input for Figure 1 generates Figure 3. This improves the graph by adding axis labels and using a new data file `agepopm.d` where the populations have been divided by one million to avoid large numbers. We shall see later that simple transformations such as this can be achieved without generating new data files.

All flavors of TeX can handle multi-line labels via the `\hbox` within `\vbox` arrangement used above, but LaTeX users will find it more natural to use the `tabular` environment [9]. Troff users can use nofill mode:

```
btex .nf Population in millions etex
```

2.2 Coordinate Systems

The graph macros automatically shift and rescale coordinates from data files, `gdraw` paths, and `glabel` locations to fit the graph. Whether the range of y coordinates is 0.64 to 4.6 or 640,000 to 4,600,000, they get scaled to fill about 88% of the

[2] Troff users should replace `btex $\bullet$ etex` with `btex \(bu etex`.

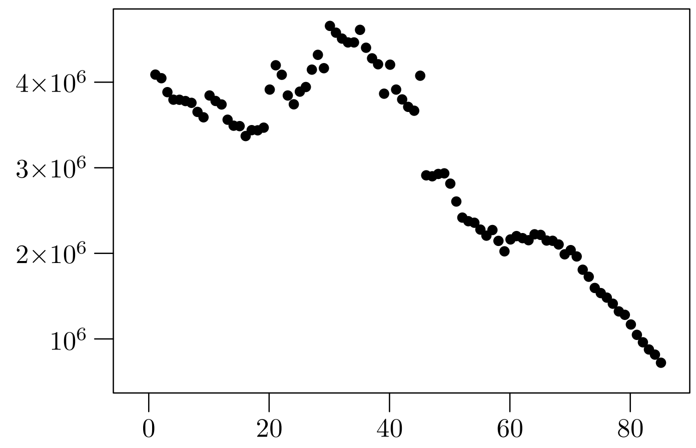


Figure 2: The 1991 age distribution plotted with bullets

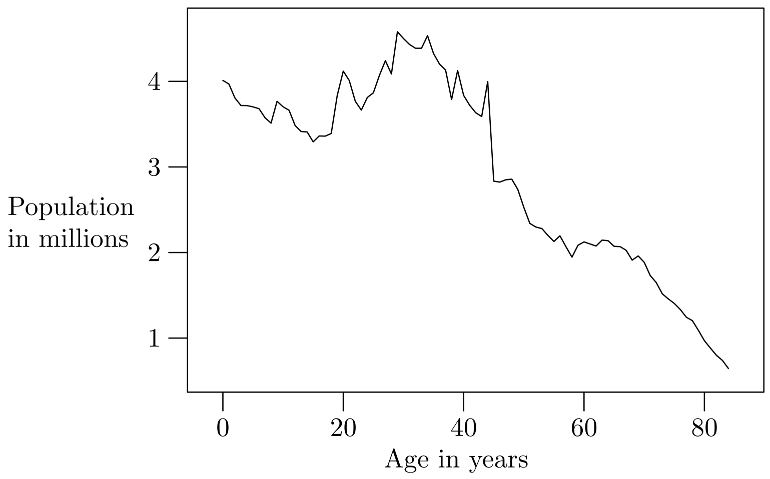


Figure 3: An improved version of the 1991 age distribution graph

height specified in the `begingraph` statement. Of course line widths, labels, and plotting symbols are not rescaled.

The `setrange` command controls the shifting and rescaling process by specifying the minimum and maximum graph coordinates:

`setrange(`⟨coordinates⟩`,` ⟨coordinates⟩`)`

where

⟨coordinates⟩ → ⟨pair expression⟩
| ⟨numeric or string expression⟩`,`
⟨numeric or string expression⟩

The first ⟨coordinates⟩ give $(x_{\min}, y_{\min})$ and the second give $(x_{\max}, y_{\max})$. The lines $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, and $y = y_{\max}$ define the rectangular frame around the graph in Figures 1–3. For example, an adding a statement

```
setrange(origin, whatever,whatever)
```

to the input for Figure 3 yields Figure 4. The first ⟨coordinates⟩ are given by the predefined pair constant `origin`, and the other coordinates are left unspecified. Any unknown value would work as well, but `whatever` is the standard MetaPost representation for an anonymous unknown value.

Notice that the syntax for `setrange` allows coordinate values to be given as strings. Many commands in the graph package allow this option. It is provided because the MetaPost language uses fixed point numbers that must be less than 32768. This limitation is not as serious as it sounds because good graph design dictates that coordinate values should be "of reasonable magnitude" [2, 11]. If you really want x and y to range from 0 to 1,000,000,

```
setrange(origin, "1e6","1e6")
```

```
draw begingraph(3in,2in);
 glabel.lft(btex \vbox{\hbox{Population} \hbox{in millions}} etex, OUT);
 glabel.bot(btex Age in years etex, OUT);
 setrange(origin, whatever,whatever);
 gdraw "agepopm.d";
 endgraph;
```

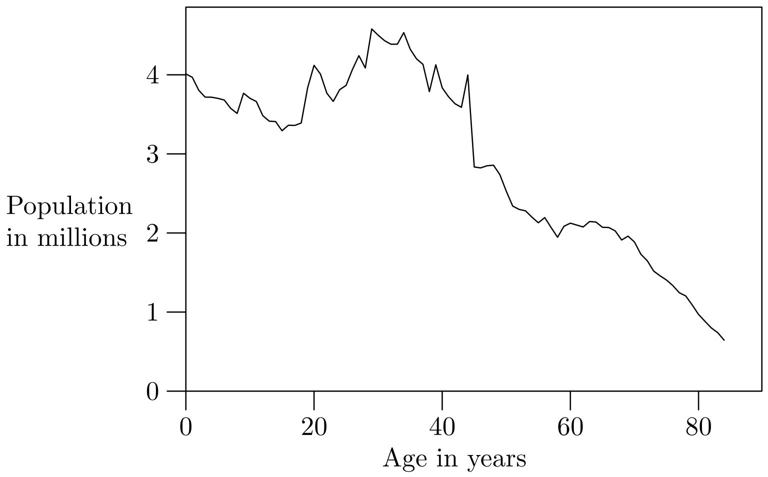


Figure 4: The 1991 age distribution graph and the input that creates it.

does the job. Any fixed or floating point representation is acceptable as long as the exponent is introduced by the letter "e".

Coordinate systems need not be linear. The `setcoords` command allows either or both axes to have logarithmic spacing:

⟨coordinate setting⟩ →
  `setcoords(`⟨coordinate type⟩`,`⟨coordinate type⟩`)`
⟨coordinate type⟩ →
  `log` | `linear` | `-log` | `-linear`

A negative ⟨coordinate type⟩ makes x (or y) run backwards so it is largest on the left side (or bottom) of the graph.

Figure 5 graphs execution times for two matrix multiplication algorithms using

```
setcoords(log,log)
```

to specify logarithmic spacing on both axes. The data file `matmul.d` gives timings for both algorithms:

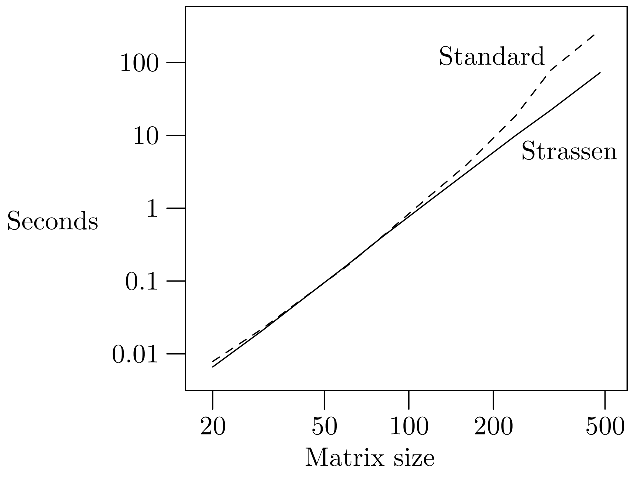
```
20  .007861  standard MM: size, seconds
30  .022051
40  .050391
60  .15922
80  .4031
120 1.53
160 3.915
240 18.55
320 78.28
480 279.24


20  .006611  Strassen: size, seconds
30  .020820
40  .049219
60  .163281
80  .3975
120 1.3125
160 3.04
240 9.95
320 22.17
480 72.60
```

A blank line in a data file ends a data set. Subsequent `gdraw` commands access additional data sets by just naming the same data file again. Since each line gives one x coordinate and one y coordinate, commentary material after the second data field on a line is ignored.

Placing a `setcoords` command between two `gdraw` commands graphs two functions in different coordinate systems as shown in Figure 6. Whenever you give a `setcoords` command, the interpreter examines what has been drawn, selects appropriate x and y ranges, and scales everything to fit. Everything drawn afterward is in a new coordinate system that need not have anything in common with the old coordinates unless `setrange` commands enforce similar coordinate ranges. For instance, the two `setrange` commands force both coordinate systems to have x ranging from 80 to 90 and y starting at 0.

When you use multiple coordinate systems, you have to specify where the axis labels go. The default is to put tick marks on the bottom and the left side of the frame using the coordinate system in



```
draw begingraph(2.3in,2in);
 setcoords(log,log);
 glabel.lft(btex Seconds etex,OUT);
 glabel.bot(btex Matrix size etex,
   OUT);
 gdraw "matmul.d" dashed evenly;
 glabel.ulft(btex Standard etex,8);
 gdraw "matmul.d";
 glabel.lrt(btex Strassen etex,7);
 endgraph;
```

Figure 5: Timings for two matrix multiplication algorithms with the corresponding MetaPost input.

```
draw begingraph(6.5cm,4.5cm);
 setrange(80,0, 90,whatever);
 glabel.bot(btex Year etex, OUT);
 glabel.lft(btex \vbox{\hbox{Emissions in} \hbox{thousands of}
   \hbox{metric tons} \hbox{(heavy line)}}etex, OUT);
 gdraw "lead.d" withpen pencircle scaled 1.5pt;
 autogrid(,otick.lft);
 setcoords(linear,linear);
 setrange(80,0, 90,whatever);
 glabel.rt(btex \vbox{\hbox{Micrograms} \hbox{per cubic}
   \hbox{meter of air} \hbox{(thin line)}}etex, OUT);
 gdraw "lead.d";
 autogrid(otick.bot,otick.rt);
 endgraph;
```

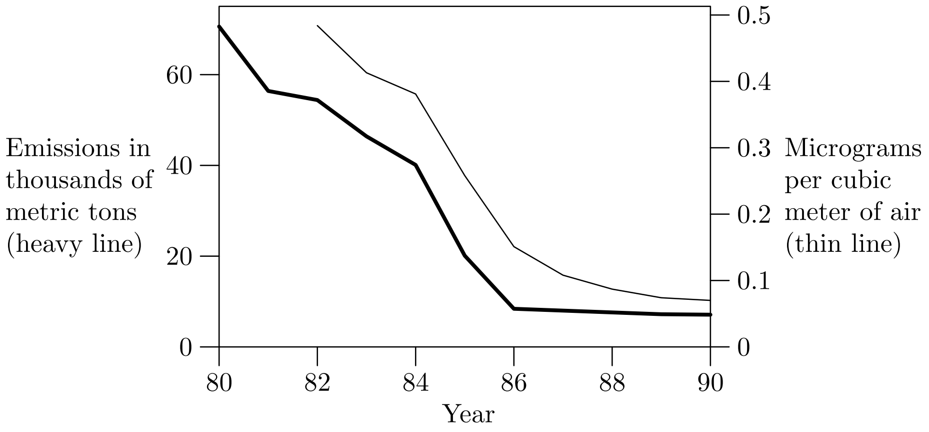


Figure 6: Annual lead emissions and average level at atmospheric monitoring stations in the United States. The MetaPost input is shown above the graph.

effect when the `endgraph` command is interpreted. Figure 6 uses the

```
autogrid(,otick.lft)
```

to label the left side of the graph with the y coordinates in effect before the `setcoords` command. This suppresses the default axis labels, so another `autogrid` command is needed to label the bottom and right sides of the graph using the new coordinate system. The general syntax is

```
autogrid(⟨axis label command⟩,
  ⟨axis label command⟩) ⟨option list⟩
```

where

```
⟨axis label command⟩ →
  ⟨empty⟩ | ⟨grid or tick⟩ ⟨label suffix⟩
⟨grid or tick⟩ → grid | itick | otick
```

The ⟨label suffix⟩ should be `lft`, `rt`, `top`, or `bot`.

The first argument to `autogrid` tells how to label the x axis and the second argument does the same for y. An ⟨empty⟩ argument suppresses labeling for that axis. Otherwise, the ⟨label suffix⟩ tells which side of the graph gets the numeric label. Be careful to use `bot` or `top` for the x axis and `lft` or `rt` for the y axis. Use `otick` for outward tick marks, `itick` for inward tick marks, and `grid` for grid lines. The ⟨option list⟩ tells how to draw the tick marks or grid lines. Grid lines tend to be a little overpowering, so it is a good idea to give a `withcolor` option to make them light gray so they do not make the graph too busy.

2.3 Explicit Grids and Framing

In case `autogrid` is not flexible enough, axis label commands generate grid lines or tick marks one at a time. The syntax is

```
⟨grid or tick⟩.⟨label suffix⟩
  (⟨label format⟩, ⟨numeric or string expression⟩)
  ⟨option list⟩
```

where ⟨grid or tick⟩ and ⟨label suffix⟩ are as in `autogrid`, and ⟨label format⟩ is either a format string like `"%g"` or a picture containing the typeset numeric label.

The axis label commands use a macro

```
format(⟨format string⟩,
  ⟨numeric or string expression⟩)
```

to typeset numeric labels. Full details appear in Section 4, but when the ⟨format string⟩ is `"%g"`, it uses decimal notation unless the number is large enough or small enough to require scientific notation.

The example in Figure 7 invokes

```
format("
```

explicitly so that grid lines can be placed at transformed coordinates. It defines the transformation `newy`$(y) = y/75 + \ln y$ and shows that this function increases almost linearly.[3] This is a little like using logarithmic y-coordinates, except that y is mapped to $y/75 + \ln y$ instead of just $\ln y$.

Figure 7 uses the command

```
frame.⟨label suffix⟩ ⟨option list⟩
```

to draw a special frame around the graph. In this case the ⟨label suffix⟩ is `llft` to draw just the bottom and left sides of the frame. Suffixes `lrt`, `ulft`, and `urt` draw other combinations of two sides; suffixes `lft`, `rt`, `top`, `bot` draw one side, and ⟨empty⟩ draws the whole frame. For example

```
frame dashed evenly
```

draws all four sides with dashed lines. The default four-sided frame is drawn only when there is no explicit `frame` command.

To label an axis as `autogrid` does but with the labels transformed somehow, use

`auto.x` or `auto.y`

for positioning tick marks or grid lines. These macros produce comma-separated lists for use in `for` loops. Any x or y values in these lists that cannot be represented accurately within MetaPost's fixed-point number system are given as strings. A standard macro package that is loaded via

```
input sarith
```

defines arithmetic operators that work on numbers or strings. Binary operators `Sadd`, `Ssub`, `Smul`, and `Sdiv` do addition, subtraction, multiplication, and division.

One possible application is rescaling data. Figure 4 used a special data file `agepopm.d` that had y values divided by one million. This could be avoided by replacing "`gdraw "agepopm.d"`" by

```
gdraw "agepop91.d";
for u=auto.y:
  otick.lft(format("  endfor
autogrid(otick.bot,)
```

2.4 Processing Data Files

The most general tool for processing data files is the `gdata` command:

```
gdata(⟨string expression⟩, ⟨variable⟩,
  ⟨commands⟩)
```

It takes a file name, a variable v, and a list of commands to be executed for each line of the data file. The commands are executed with `i` set to the input line number and strings v`1`, v`2`, v`3`, … set to the input fields on the current line. A null string marks the end of the v array.

[3] The manual [6] explains how `vardef` defines functions and `mlog` computes logarithms.

```
vardef newy(expr y) = (256/75)*y + mlog y enddef;
draw begingraph(3in,2in);
 glabel.lft(btex \vbox{\hbox{Population} \hbox{in millions}} etex, OUT);
 gdraw "ttimepop.d";
 for y=5,10,20,50,100,150,200,250:
   grid.lft(format("%g",y), newy(y)) withcolor .85white;
 endfor
 autogrid(grid.bot,) withcolor .85white;
 frame.llft;
 endgraph;
```

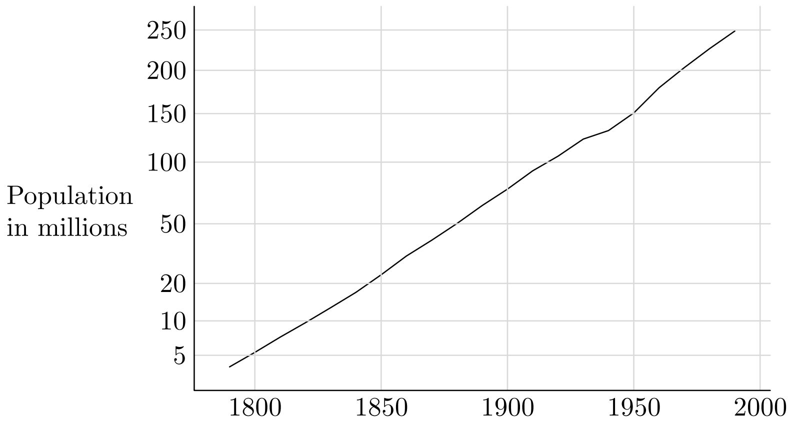


Figure 7: Population of the United States in millions versus time with the population re-expressed as $p/75 + \ln p$. The MetaPost input shown above the graph assumes a data file `ttimepop.d` that gives (year, $p/75 + \ln p$) pairs.

Using a `glabel` command inside of `gdata` generates a scatter plot as shown in Figure 8. The data file `countries.d` begins

```
20.910 75.7 US
 1.831 66.7 Alg
```

where the last field in each line gives the label to be plotted. Setting `defaultfont` in the first line of input selects a small font for these labels. Without these labels, no `gdata` command would be needed. Replacing the `gdata` command with

```
gdraw "countries.d" plot btex $\circ$etex
```

would change the abbreviated country names to open circles.

Both `gdraw` and `gdata` ignore an optional initial '`%`' on each input line, parse data fields separated by white space, and stop if they encounter an input line with no data fields. Leading percent signs make graph data look like MetaPost comments so that numeric data can be placed at the beginning of a MetaPost input file.

It is often useful to construct one or more paths when reading a data file with `gdata`. The `augment` command is designed for this:

`augment.`⟨path variable⟩ `(`⟨coordinates⟩`)`

If the path variable does not have a known value, it becomes a path of length zero at the given coordinates; otherwise a line segment to the given coordinates is appended to the path. The ⟨coordinates⟩ may be a pair expression or any combination of strings and numerics as explained at the beginning of Section 2.2.

If a file `timepop.d` gives t, p pairs, `augment` can be used like this to graph `newy(`p`)` versus t:

```
path p;
gdata("timepop.d", s,
  augment.p(s1, newy(scantokens s2)); );
gdraw p;
```

(MetaPost's `scantokens` primitive interprets a string as if it were the contents of an input file. This finds the numeric value of data field `s2`.)

Figure 9 shows how to use `augment` to read multiple column data and make multiple paths. Paths `p2`, `p3`, `p4`, `p5` give cumulative totals for columns 2 through 5 and pictures `lab2` through `lab5` give corresponding labels. The expression

```
image(unfill bbox lab[j]; draw lab[j])
```

executes the given drawing commands and returns the resulting picture: "`unfill bbox lab[j]`" puts

```
defaultfont:="cmr7";
draw begingraph(3in,2in);
  glabel.lft(btex \vbox{\hbox{Life}\hbox{expectancy}} etex, OUT);
  glabel.bot(btex Per capita G.N.P. (thousands of dollars) etex, OUT);
  setcoords(log,linear);
  gdata("countries.d", s,
    glabel(s3, s1, s2);
  )
  endgraph;
```

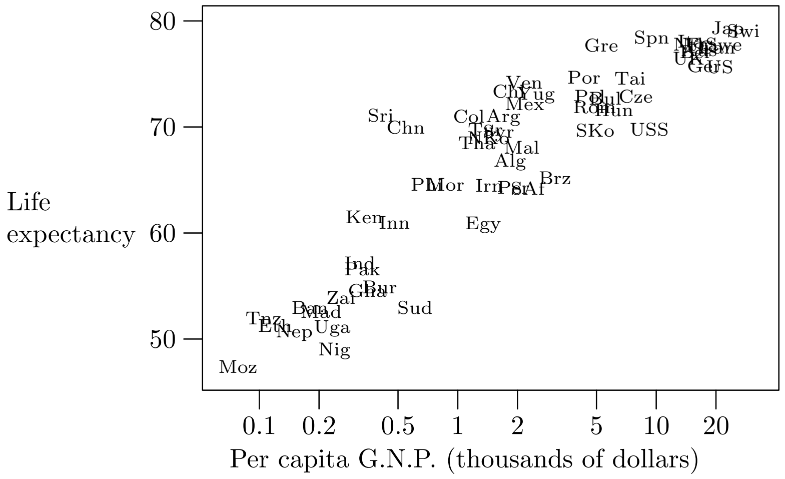


Figure 8: A scatter plot and the commands that generated it

down a white background and "`draw lab[j]`" puts the label on the background. The `gfill` command is just like `gdraw`, except it takes a cyclic path and fills the interior with a solid color. The color is black unless a `withcolor` clause specifies another color. See the manual [6] for explanations of `for` loops, arrays, colors, and path construction operators like `--`, `cycle`, and `reverse`.

3 Manipulating Big Numbers

MetaPost inherits a fixed-point number system from Knuth's METAFONT [8]. Numbers are expressed in multiples of 2^{-16} and they must have absolute value less than 32768. Knuth chose this system because it is perfectly adequate for font design, and it is guaranteed to give identical results on all types of computers. Fixed-point numbers are seldom a problem in MetaPost because all computations are based on coordinates that are limited by the size of the paper on which the output is to be printed. This does not hold for the input data in a graph-drawing application. Although graphs look best when coordinate axes are labeled with numbers of reasonable magnitude, the strict limits of fixed-point arithmetic would be inconvenient.

A simple way to handle large numbers is to include the line

```
input sarith
```

and then use binary operators `Sadd`, `Ssub`, `Smul`, and `Sdiv` in place of `+`, `-`, `*`, and `/`. These operators are inefficient but very flexible. They accept numbers or strings and return strings in exponential notation with the exponent marked by "`e`"; e.g., `"6.7e-11"` means 6.7×10^{-11}.

The unary operator[4]

`Sabs` ⟨string⟩

finds a string that represents the absolute value. Binary operators `Sleq` and `Sneq` perform numeric comparisons on strings and return boolean results.

The operation

`Scvnum` ⟨string⟩

finds the numeric value for a string if this can be done without overflowing MetaPost's fixed-point number system. If the string does not contain "`e`", it is much more efficient to use the primitive operation

`scantokens` ⟨string⟩

The above operators are based on a low-level package that manipulates numbers in "`Mlog` form."

[4] The argument to a unary operator need not be parenthesized unless it is an expression involving binary operators.

```
draw begingraph(3in,2in);
  glabel.lft(btex \vbox{\hbox{Quadrillions}\hbox{of BTU}} etex, OUT);
  path p[];
  numeric t;
  gdata("energy.d", $,
    t:=0; augment.p1($1,0);
    for j=2 upto 5:
       t:=t+scantokens $[j]; augment.p[j]($1,t);
    endfor)
  picture lab[];
  lab2=btex coal etex; lab3=btex crude oil etex;
  lab4=btex natural gas etex; lab5=btex hydroelectric etex;
  for j=5 downto 2:
    gfill p[j]--reverse p[j-1]--cycle withcolor .16j*white;
    glabel.lft(image(unfill bbox lab[j]; draw lab[j]), .7+length p[j]);
  endfor
  endgraph;
```

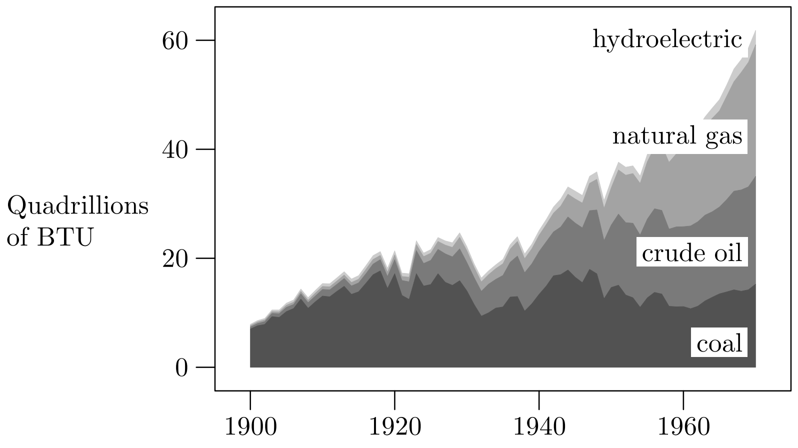


Figure 9: A graph of U.S. annual energy production and the commands that generated it

A number x in `Mlog` form represents

$$\mu^{2^{16}x}, \quad \text{where } \mu = -e^{2^{-24}}.$$

Any value between 1.61×10^{-28} and 3.88×10^{55} can be represented this way. (There is a constant `Mten` such that $k *$ `Mten` represents 10^k for any integer k in the interval $[-29, 55]$.)

The main reason for mentioning `Mlog` form is that it allows graph data to be manipulated as a MetaPost path. The function

`Mreadpath(`⟨file name⟩`)`

reads a data file and returns a path where all the coordinates are in `Mlog` form. An internal variable `Gpaths` determines whether `gdraw` and `gfill` expect paths to be given in `Mlog` form. For example, this graphs the data in `agepop91.d` with y coordinates divided by one million:

```
interim Gpaths:=log;
gdraw Mreadpath("agepop91.d")
  shifted (0,-6*Mten);
```

4 Typesetting Numbers

The graph package needs to compute axis labels and then typeset them. The macro

`format(`⟨string expression⟩`,`
  ⟨numeric or string expression⟩`)`

does this. You must first `input graph` or `input format` to load the macro file. The macro takes a format string and a number to typeset and returns a picture containing the typeset result. Thus

`format("%g",2+2)` yields 4

and

`format("%3g","6.022e23")` yields $6.02{\times}10^{23}$

A format string consists of

- an optional initial string not containing a percent sign,
- a percent sign,
- an optional numeric precision p,
- one of the conversion letters `e`, `f`, `g`, `G`,
- an optional final string β.

The initial and final strings are typeset in the default font (usually `cmr10`), and the typeset number is placed between them. For the `e` and `g` formats, the precision p is the number of significant digits allowed after rounding; for `f` and `G`, the number is rounded to the nearest multiple of 10^{-p}. If the precision is not specified, the default is $p = 3$. The `e` format always uses scientific notation and the `f` format uses ordinary decimal notation but reverts to scientific notation if the number is at least 10000. The `g` and `G` formats also revert to scientific notation for nonzero numbers of magnitude less than 0.001.

The `format` macro needs a set of templates to determine what font to use, how to position the exponent, etc. The templates are normally initialized automatically, but it is possible to set them explicitly by passing five picture expressions to `init_numbers`. For instance, the default definition for TeX users is

```
init_numbers(btex$-$etex, btex$1$etex,
  btex${\times}10$etex,
  btex${}^-$etex, btex${}^2$etex)
```

The first argument tells how to typeset a leading minus sign; the second argument is an example of a 1-digit mantissa; third comes whatever to put after the mantissa in scientific notation; next come a leading minus sign for the exponent and a sample 1-digit exponent.

Picture variable `Fe_plus` gives a leading plus sign for positive numbers, and `Fe_base` gives whatever should precede the exponent when typesetting a power of ten. Calling `init_numbers` initializes `Fe_plus` to an empty picture and constructs `Fe_base` from its second and third arguments.

5 Conclusion

The graph package makes it convenient to generate graphs from within the MetaPost language. The primary benefits are the power of the MetaPost language and its ability to interact with TeX or troff for typesetting labels. Typeset labels can be stored in picture variables and manipulated in various ways such as measuring the bounding box and providing a white background.

We have seen how to generate shaded regions and control line width, color, and styles of dashed lines. Numerous other variations are possible. The full MetaPost language [6] provides many other potentially useful features. It also has enough computing power to be useful for generating and processing data.

A Summary of the Graph Package

In the following descriptions, italic letters such as w and h denote expression parameters and words in angle brackets denote other syntactic elements. Unless specified otherwise, expression parameters can be either numerics or strings. An ⟨option list⟩ is a list of drawing options such as `withcolor .5white` or `dashed evenly`; a ⟨label suffix⟩ is one of `lft`, `rt`, `top`, `bot`, `ulft`, `urt`, `llft`, `lrt`.

A.1 Graph Administration

`begingraph(`w`,`h`)` Begin a new graph with the frame width and height given by numeric parameters w and h.

`endgraph` End a graph and return the resulting picture.

`setcoords(`t_x`,` t_y`)` Set up a new coordinate system as specified by numeric flags t_x, t_y. Flag values are ±`linear` and ±`log`.

`setrange(`⟨coordinates⟩`,` ⟨coordinates⟩`)` Set the lower and upper limits for the current coordinate system. Each ⟨coordinates⟩ can be a single pair expression or two numeric or string expressions.

A.2 Drawing and Labeling

All of the drawing and labeling commands can be followed by an ⟨option list⟩. In addition to the usual MetaPost drawing options, the list can contain a `plot` ⟨picture⟩ clause to plot a specified picture at each data point.

The drawing and labeling commands are closely related to a set of similarly named commands in plain MetaPost. The `gdrawarrow` and `gdrawdblarrow` commands are included to maintain this relationship.

`gdotlabel.`⟨label suffix⟩`(`p`,` ⟨location⟩`)` This is like `glabel` except it also puts a dot at the location being labeled.

`gdraw` p Draw path p, or if p is a string, read coordinate pairs from file p and draw a polygonal line through them.

`gdrawarrow` p This is like `dgraw` p except it adds an arrowhead at the end of the path.

`gdrawdblarrow` p This is like `dgraw` p except it adds an arrowhead at each end of the path.

`gfill` p Fill cyclic path p or read coordinates from the file named by string p and fill the resulting polygonal outline.

`glabel.`⟨label suffix⟩`(`p`,` ⟨location⟩`)` If p is not a picture, it should be a string. Typeset it using `defaultfont`, then place it near the given location and offset as specified by the ⟨label suffix⟩. The ⟨location⟩ can be x and y coordinates, a pair giving x and y, a numeric value giving a time on the last path drawn, or `OUT` to label the outside of the graph.

A.3 Grids, Tick Marks, and Framing

`auto.`⟨`x` or `y`⟩ Generate default x or y coordinates for tick marks.

`autogrid(`⟨axis label command⟩`,` ⟨axis label command⟩`)` Draw default axis labels using the specified commands for the x and y axes. An ⟨axis label command⟩ may be ⟨empty⟩ or it may be `itick`, `otick`, or `grid` followed by a ⟨label suffix⟩.

`frame.`⟨label suffix⟩ ⟨option list⟩ Draw a frame around the graph, or draw the part of the frame specified by the ⟨label suffix⟩.

`grid.`⟨label suffix⟩`(`f`,`z`)` Draw a grid line across the graph from the side specified by the ⟨label suffix⟩, and label it there using format string f and coordinate value z. If f is a picture, it gives the label.

`itick.`⟨label suffix⟩`(`f`,`z`)` This is like `grid` except it draws an inward tick mark.

`otick.`⟨label suffix⟩`(`f`,`z`)` This is like `grid` except it draws an outward tick mark.

A.4 Miscellaneous Commands

`augment.`⟨variable⟩`(`⟨coordinates⟩`)` Append ⟨coordinates⟩ to the path stored in ⟨variable⟩.

`format(`f`,` x`)` Typeset x according to format string f and return the resulting picture.

`gdata(`f`,` ⟨variable⟩`,` ⟨commands⟩`)` Read the file named by string f and execute ⟨commands⟩ for each input line using the ⟨variable⟩ as an array to store data fields.

`init_numbers(`s`,` m`,` x`,` t`,` e`)` Provide five pictures as templates for future `format` operations: s is a leading minus; m is a sample mantissa; x follows the mantissa; t is a leading minus for the exponent e.

`Mreadpath(`f`)` Read a path for the data file named by string f and return it in "`Mlog` form".

A.5 Arithmetic on Numeric Strings

It is necessary to `input sarith` before using the following macros:

`Sabs` x Compute $|x|$ and return a numeric string.

x `Sadd` y Compute $x + y$ and return a numeric string.

`Scvnum` x Return the numeric value for string x.

x `Sdiv` y Compute x/y and return a numeric string.

x `Sleq` y Return the boolean result of the comparison $x \leq y$.

x `Smul` y Compute $x * y$ and return a numeric string.

x `Sneq` y Return the boolean result of the comparison $x \neq y$.

x `Ssub` y Compute $x - y$ and return a numeric string.

A.6 Internal Variables and Constants

`Autoform` Format string used by `autogrid`. Default: `"%g"`.

`Fe_base` What precedes the exponent when typesetting a power of ten.

`Fe_plus` Picture of the leading plus sign for positive exponents.

`Gmarks` Minimum number of tick marks per axis for `auto` and `autogrid`. Default: 4.

`Gminlog` Minimum largest/smallest ratio for logarithmic spacing with `auto` and `autogrid`. Default: 3.0.

`Gpaths` Code for coordinates used in `gdraw` and `gfill` paths: `linear` for standard form, log for "`Mlog` form".

`Mten` The "`Mlog` form" for 10.0

B New Language Features

The `graph.mp` macros and the arithmetic routines in `marith.mp` and `sarith.mp` use various language features that were introduced in Version 0.60 of the MetaPost language. We summarize these features here because they are not covered in existing documentation [6, 5]. Also new is the built-in macro

`image(`⟨drawing commands⟩`)`

that was used in Section 2.4 to find the picture produced by a sequence of drawing commands.

B.1 Reading and Writing Files

A new operator

`readfrom` ⟨file name⟩

returns a string giving the next line of input from the named file. The ⟨file name⟩ can be any primary expression of type string. If the file has ended or cannot be read, the result is a string consisting of a single null character. The preloaded `plain` macro package introduces the name `EOF` for this string.

After `readfrom` has returned `EOF`, additional reads from the same file cause the file to be reread from the start.

The opposite of `readfrom` is the command

`write` ⟨string expression⟩ `to`⟨file name⟩

This writes a line of text to the specified output file, opening the file first if necessary. All such files are closed automatically when the program terminates. They can also be closed explicitly by using `EOF` as the ⟨string expression⟩. The only way to tell if a `write` command has succeeded is to close the file and use `readfrom` to look at it.

B.2 Extracting Information from Pictures

MetaPost pictures are composed of stroked lines, filled outlines, pieces of typeset text, clipping paths, and `setbounds` paths. (A `setbounds` path gives an artificial bounding box as is needed for TeX output.) A picture can have many components of each type. They can be accessed via an iteration of the form

`for` ⟨symbolic token⟩ `within` ⟨picture expression⟩ `:`
⟨loop text⟩ `endfor`

The ⟨loop text⟩ can be anything that is balanced with respect to `for` and `endfor`. The ⟨symbolic token⟩ is a loop variable that scans the components of the picture in the order in which they were drawn. The component for a clipping or `setbounds` path includes everything the path applies to. Thus if a single clipping or `setbounds` path applies to everything in the ⟨picture expression⟩, the whole picture could be thought of as one big component. In order to make the contents of such a picture accessible, the `for`...`within` iteration ignores the enclosing clipping or `setbounds` path in this case.

Once the `for`...`within` iteration has found a picture component, there are numerous operators for identifying it and extracting relevant information. The operator

`stroked` ⟨primary expression⟩

tests whether the expression is a known picture whose first component is a stroked line. Similarly, the `filled` and `textual` operators return `true` if the first component is a filled outline or a piece of typeset text. The `clipped` and `bounded` operators test whether the argument is a known picture that starts with a clipping path or a `setbounds` path. This is true if the first component is clipped or bounded or if the entire picture is enclosed in a clipping or `setbounds` path.

There are also numerous part extraction operators that test the first component of a picture. If `p` is a picture and `stroked p` is true, `pathpart p` is the path describing the line that got stroked, `penpart p` is the pen that was used, `dashpart p` is the dash pattern, and the color is

`(redpart p, greenpart p, bluepart p)`

If the line is not dashed, `dashpart p` returns an empty picture.

The same part extraction operators work when `filled p` is true, except that `dashpart p` is not meaningful in that case. For text components, `textual p` is true, `textpart p` gives the text that got typeset, `fontpart p` gives the font that was used, and `xpart p`, `ypart p`, `xxpart p`, `xypart p`, `yxpart p`, `yypart p` tell how the text has been shifted, rotated, and scaled. The `redpart`, `greenpart`, and `bluepart` operators also work for text components.

When `clipped p` or `bounded p` is true, `pathpart p` gives the clipping or `setbounds` path and the other part extraction operators are not meaningful. Such non-meaningful part extractions do not generate errors — they return null values instead: the trivial path `(0,0)` for `pathpart`, `nullpen` for `penpart`, an empty picture for `dashpart`, zero for `redpart`, `greenpart`, `bluepart`, and the null string for `textpart` or `fontpart`.

One final operator for extracting information from a picture is

`length` ⟨picture primary⟩

This returns the number of components that a `for`...`within` iteration would find.

B.3 Other New Features

The `marith.mp` and `sarith.mp` packages use numbers of magnitude 4096 or more. Since such numbers can cause overflow problems in MetaPost's linear equation solving and path fitting algorithms, they are normally allowed only as intermediate results. This limitation is removed when the internal variable `warningcheck` is zero. In earlier versions of MetaPost, the limitation could be removed for variables but explicit constants were always restricted to be less than 4096.

For completeness, we also mention one other new feature of MetaPost Version 0.60. When TeX material is included in a picture via the `btex`...`etex` feature, the thickness of horizontal and vertical rules gets rounded to exactly the right number of pixels; i.e., interpreting MetaPost output according to the PostScript scan conversion rules [7] makes the pixel width equal to the ceiling of the unrounded width. In fact, a similar relationship holds for all line widths. The generated PostScript sets line widths by first transforming to device coordinates and rounding appropriately.

References

[1] Jon L. Bentley and Brian W. Kernighan. Grap—a language for typesetting graphs. In *Unix Research System Papers*, volume II, pages 109–146. AT&T Bell Laboratories, Murray Hill, New Jersey, tenth edition, 1990.

[2] William S. Cleveland. *The Elements of Graphing Data*. Hobart Press, Summit, New Jersey, 1985.

[3] William S. Cleveland. A model for studying display methods of statistical graphics (with discussion). *Journal of Computational and Statistical Graphics*, 3, to appear.

[4] William S. Cleveland. *Visualizing Data*. Hobart Press, Summit, New Jersey, to appear.

[5] John D. Hobby. Introduction to MetaPost. In *EuroTEX '92 Proceedings*, pages 21–36, September 1992.

[6] John D. Hobby. A user's manual for MetaPost. Computing Science Technical Report no. 162, AT&T Bell Laboratories, Murray Hill, New Jersey, April 1992. Available as http://cm.bell-labs.com/cs/cstr/162.ps.gz.

[7] Adobe Systems Inc. *PostScript Language Reference Manual*. Addison Wesley, Reading, Massachusetts, second edition, 1990.

[8] Donald E. Knuth. *METAFONT the Program*. Addison Wesley, Reading, Massachusetts, 1986. Volume D of *Computers and Typesetting*.

[9] Leslie Lamport. *LATEX: A Document Preparation System*. Addison Wesley, Reading, Massachusetts, 1986.

[10] U.S. Bureau of the Census. *Statistical Abstracts of the United States: 1992*. Washington, D.C., 112th edition, 1992.

[11] Edward R. Tufte. *Visual Display of Quantitative Information*. Graphics Press, Box 430, Cheshire, Connecticut 06410, 1983.

◇ John D. Hobby
Bell Laboratories
Room 2C-458
700 Mountain Ave.
Murray Hill, NJ 07974-0636
`hobby@research.bell-labs.com`

Reports

The status quo of the $\mathcal{NTS}$ project

Hans Hagen[1]

The reason

In the last decade, several initiatives were started in extending "TEX The Program". Most closely related to the original is ε-TEX. This program adds some primitives to TEX that provide more control over expansion, extends the range of registers beyond 255, and provides bidirectional typesetting at the paragraph level. The fact that ε-TEX is programmed within the original `WEB` concept makes it a close relative.

Donald Knuth's main motivation for writing TEX was the need to typeset his own books in the best of typographic traditions. Therefore, it will be no surprise that its typographic engine favours the English script over other, more complicated, scripts. Composed characters and glyphs, advanced ligatures, complicated input encodings, and tightly integrated multi-directional typesetting, are not handled well by TEX, but they are covered by Omega, yet another relative of good old TEX. Omega not only provides an advanced input translation processor, it also extends the range of registers. In contrast to ε-TEX, Omega can handle a large number of math font families. However, it is especially the multi-lingual capabilities that have given Omega a well-deserved position in the family of TEX descendants.

The third major descendant of TEX is pdfTEX. Where ε-TEX demonstrates quite well that TEX can be extended, and Omega gives TEX its place in typesetting non-western languages, pdfTEX lets TEX survive in the turbulent Internet environment. It does so by providing an alternative back-end, which enables TEX users to prepare documents that can be distributed, viewed and printed without additional resources; in color, with graphics included, and enhanced with hyperlinks and widgets.

Because pdfTEX can be combined with ε-TEX it can also provide the ε-TEX goodies, but it offers more. pdfTEX extends TEX's paragraph building

This article first appeared in *Die TEXnische Komödie*, 1/2001, pages 36–53. Reprinted with permission.

[1] I want to express my thanks to Jerzy Ludwichowski, Karel Skoupý, and Volker RW Schaa for proofreading this article, improving the English and providing suggestions. Don't confuse my opinions with theirs.

routines with character protruding (marginal kerning) as well as horizontal font expansion (*hz* optimization). In doing so, pdfTEX ensures that TEX is still quite up to date and ready for the near future.

There are a few more extensions, like those provided by MLTEX, which focuses on 8-bit encodings and mapping, but these extensions are small compared to the ones already mentioned. Being useful for European languages, they are often part of the mainstream TEX distributions, probably without users being aware of it.

So, to summarize the current state of TEX, we can classify the programs developed so far as follows:

- TEX: the stable and bug-free ancestor
- ε-TEX: the useful successor
- Omega: the much-needed extension
- pdfTEX: the successful descendant

pdfTEX differs from the other two TEX descendants in that it goes a step further in combining more tools into one. This is a logical consequence of the fact that it is a typesetting engine as well as a back-end. It has to handle all aspects of fonts, images and resources. It does so by using new code, written within the `WEB` paradigm, but it also uses existing code, available as precompiled C libraries, while some of its subsystems are written from scratch in C instead of Pascal.

When TEX was written, Pascal was one of the favourite structured languages. In order to make TEX portable, Knuth sacrificed some of Pascal's features and implemented his own memory management. Also, instead of relying on Pascal data structures, he used his literate programming environment `WEB` as a wrapper. As a result, extending TEX is possible, but only to a certain extent. The main reason for this is that many data structures are reused and/or overloaded. Another handicap is that many variables have a global nature, so that one should be very careful in manipulating them. TEX is one of the few programs that really benefit from faster machines since the code is highly optimized, but sometimes these optimizations have the nasty side effect that they obscure what the code does. It is no secret that pdfTEX demonstrates quite well that the limits of extending TEX within its current concept have been reached.

At the time when ε-TEX took shape, Omega prototypes started to show up, and pdfTEX was not yet invented, there was already a more structured discussion taking place on re-implementing "TEX The Program". This re-implementation should be done in such a way that extending TEX would be more easy. This envisaged successor has been designated as The New Typesetting System, or $\mathcal{NTS}$ for short.

For quite some time, the ε-TEX and the $\mathcal{NTS}$ projects were combined and hosted by the German user group DANTE. Since the start of the project, DANTE has been funding it substantially. This makes the project unique in the TEX world, since the projects ε-TEX, Omega and pdfTEX were not funded at all, or at least not to that extent. Before discussing the $\mathcal{NTS}$ project, we will spend some words on the environment where these developments take place.

The environment

Visiting a TEX user group meeting is a special experience. Such meetings often look more like a gathering of family and friends than a conference of experts. This is not to say that the people present are not experts. Actually, they are an interesting mix of highly qualified professionals with many areas of interest. They share the feeling that TEX is special, and by using TEX they can express their knowledge on paper in the way that they want. Although a minority of them has in-depth typographic knowledge by education, they embody quite some expertise in the, sometimes even dark, areas of high quality automatic typesetting.

Given that everything related to computers evolves fast, the TEX community is rather stable. Many users will stick to using TEX when they are permitted, and even when they are forced to use commercial software in their offices, they keep an eye on TEX. However, open source software is gaining attention and we may consider TEX and friends to be one of the oldest examples of open source. (It is in this respect interesting to observe that TEX distributions are always struggling with the public licenses, that somehow do not fit them well. Many TEX distributions depend on stability and consistency and thereby sometimes pose some restrictions, mainly to guarantee their users a working system.)

One of the main drives for using TEX is that it makes one independent, especially if one also uses related or similar free tools. Although there are commercial versions of TEX available, some with quite interesting extensions, the wish to be independent implies that the successor of TEX has to originate from the user community and so far, the extensions mentioned before did so.

As a demonstration that TEX could be extended, Donald Knuth added the `\special` and `\write` primitives. I think that there is much truth in saying that although they can be qualified as "just" extensions, both mechanisms have given TEX an edge over competitors. Two decades after TEX was born, we make documents with lots of graphics, color, and extensive referencing, all of which would not be possible without those primitives.

This demonstrates that what can be regarded as an interesting example of an extension today, tomorrow can prove to be a necessity. Currently, pdfTEX has some extensions that are waiting to be used to the full extent some day in the future.

The number of people that understand enough of programming, typography and user interfacing to extend "TEX The Program", is not that large. Therefore, the statement that TEX is extensible is rather an optimistic one. Even if a successor would be implemented using today's technologies, this would not change much. And if some limitations of the good old TEX can be qualified as fundamental shortcomings, this does not automatically mean that replacing them by better alternatives can be achieved in a couple of days programming. For some problems there are no simple solutions, and some of the current limitations are quite natural, given the solution space.

The development of pdfTEX is a good demonstration that, although many people are involved in testing the core program, only a few people are involved in the actual development of the program. Actually, the making of pdfTEX is mainly a one-person job, namely Hàn Thế Thành's. But, this one person can fall back on the experience embodied in the TEX community. Experts in the areas of fonts, images, PDF and macro writing can be consulted and when they see the potential of the extensions, they are willing to participate. The number of experts is small, but their expertise is available whenever needed. Those operating at the cutting edge of what TEX can do want to be involved, and often are involved. Fortunately only one person pulls the car, which means that right from the start working prototypes were available, bugs were being fixed quite fast, and what is even more important, design decisions were made.

Because TEX has its own DVI output format, the whole suite of related programs (think of DVI viewers and converters and font generators) is rather independent from commercial developments. Because pdfTEX is used also to produce PDF output, it is more dependent on the outside world. It is no secret that Hàn Thế Thành has spent quite a lot of time in keeping (buggy) viewers happy and figuring out the real PDF specs. One (maybe only philosophical) question we should ask ourselves is if we want to be that dependent. Both alternatives ask their price.

In the last few years we have seen that (finally) the TEX community managed to get a hold on their multitude of files and resources. There is a well-defined TEX directory structure and there are some de facto standard distributions with binaries, fonts, macros and more. As a direct result, extensions like ε-TEX, Omega, and pdfTEX are available for everyone who uses TEX and on many platforms.

This also means that the maintainers of those resources (distributions) can ensure that such extensions are being integrated into the current framework of TEX in a natural way. For instance, when Omega is part of a distribution, its unique (re-)encoding and font resources are available too. Or, when pdfTEX is on someone's system, one can also be sure that the right configuration files are around somewhere. Development of new technologies is integrated into the constant process of updating and distributing TEX.

I already mentioned TEX user groups. They are organized by country or language and many of them have regular meetings and journals. Although the number of members differs from hundreds to thousands, the number of users that attend meetings is often not more than 75–100. A survey by the NTG showed that many members, when asked for the reason to be a member, responded that they are a member out of sympathy. Although many of them do not understand everything that is published, they are happy to be kept informed that there are developments. It shows them that TEX is alive. Of course members also like the regular distribution of CD-ROMs and the support that mailing lists provide.

So, whereas the large audience wants to be kept informed and is willing to support the TEX community, a small group actively attends meetings where issues like the future of TEX, extending TEX, and writing of macros are discussed. It will be no surprise that this group harbours many of the people that also take part in the developments.

We can summarize the main characteristics of the TEX community as follows:

- the developers want to be involved,
- the maintainers want to be in control,
- the users want to be kept informed, and
- they all want to be independent.

It is in this framework of TEX developments and the TEX community that I will discuss the current state of the $\mathcal{NTS}$ project. So far I have been rather general in my remarks, but I will be more explicit from now on. The following observations can therefore be seen as personal ones, and I express my sincere hope that future developments may benefit from them.

The project

I started this article by mentioning a few extensions to TEX of which the $\mathcal{NTS}$ project was planned to become one. It was started in the early nineties, and after some years of discussion the decision was made to re-implement "TEX The Program" using a modern programming language and applying today's software technology.

In spite of the fact that the project has run for nearly ten years, it is quite unknown. One reason for this is that for a long time it was only a mental exercise. Where each of ε-TEX, Omega and pdfTEX at a certain point led to a real usable product $\mathcal{NTS}$ only existed in the minds of a few people. I don't know much about what took place in those early days, but I am told that $\mathcal{NTS}$ was discussed by a broad audience, but at the moment when I joined the team, the group of people that were taking part in it had become rather small.

At a certain point in time the $\mathcal{NTS}$ dream became an official project and there are not that many of them in the TEX community. Most efforts are concentrated around a rather active group of developers, and driven by users who see the benefits from those efforts. The community is rather open, and the lines of communication are short. This means that when someone becomes aware of an effort that is of common interest, this knowledge spreads rather fast.

Knowing that TEX has some limitations and that ε-TEX could not solve them all, it should not be a surprise that $\mathcal{NTS}$ became the magic successor that was supposed to solve those problems. Its official project statement gave it a reputation beforehand. The magic resulted from the fact that for a long time there had been talks of a successor, but no real progress was seen. It is interesting to observe that meanwhile some extensions have been implemented in ε-TEX, Omega and pdfTEX in a quite acceptable way, which proves that demand can lead to solutions quite effectively.

In many user groups, or sometimes between user groups, projects are being launched with ambitious goals. Some of these projects keep rolling while others get stuck in the conceptual phase or merge with other efforts. Most projects in one way or another contribute to the constant developments, if only because their ideas merge with others. None of these projects is really official, and as far as I know, none of them is like the $\mathcal{NTS}$ project.

The $\mathcal{NTS}$ project, for instance, has an interesting structure. There is a managing director, a project manager, a technical director, about three members and (since recently again) a treasurer. The real work is done by *one* paid programmer. Although undoubtedly the original ideas behind this structure were sound, in practice it does not work out that well. One reason is that this is not a real project in the sense of projects that are being run within institutions or companies. There are no clear roles, and there are no clear functions amid the structure. The project is not embedded in research, but there have even been suggestions to organize the project as a legal body. Apart from occasional email exchanges, there is no day-to-day communication, no formal responsibilities and there is no planning.

However, there is progress, which is mainly due to the fact that there is a professional programmer involved. Thanks to DANTE, the project was able to hire such a programmer. One of the quoted reasons behind making the conversion work into a paid job was that it would speed up the process. Another reason was that it would lead to a consistent redesign. We can safely agree with the second reason, but right from the start it has proven to be impossible to estimate how much time was needed.

The latter is in itself interesting. Given that TEX is considered to be a well-documented program, and given that it is almost bug free, the first very optimistic estimate was that a conversion would take a few months using a rapid prototyping language. This later became more than two years because the prototyping stage was omitted. So far, each intermediate estimate for the moment when the first stage could be finished has been wrong.

This has its (in itself valid) reasons. As I mentioned before, users want to be in control, and part of this control is in using stable tools. And, "TEX The Program" is as stable as a program can be, both in terms of functionality and in terms of reliability. This is clearly proved by the fact that during the process of re-implementation, no bug has shown up in the original TEX, although there are certainly questionable areas. However, in the process of cleaning up and reaching full compatibility, a real bug in TEX surfaced when processing the TEXbook.[2]

For many users stability means that any future extensions, like $\mathcal{NTS}$, should be able to compile existing documents and macros. For some users, this

[2] The bug is related to `\xleaders` and makes the last leading box disappear in an inconsistent way. Karel Skoupý and Bernd Raichle did an in-depth analysis of this bug and will report on this.

also means that the result should be 100% compatible, both in terms of DVI output as well as the log file content.

A considerable amount of time has been spent on making the re-implementation 100% TEX compatible. As a side effect, the new code is not as beautiful as it could be, due to some strange dependencies, resulting from the requirement that also the log files should be identical. However, this also resulted in the new implementation being quite bug free, because the programmer had to test every tiny aspect in order to get exactly the same DVI and log files as TEX does. Full compatibility is only the starting point, and future (extended) versions would be upward compatible in functionality, but will not necessarily produce the same output.

I will not elaborate on the pros and cons of the conversion, the problems encountered, the joy and frustrations of the programmer, the quality of the code, portability and the performance of the re-implementation. In due time Karel Skoupý, the $\mathcal{NTS}$ programmer, will share his insights with us in a more systematic way, as he already did at several user group meetings. However, I think that the project missed a chance to research in a systematic way why it took so long to go from one implementation to another, especially since the language of choice, Java, qualified as a highly portable and easy to use language.

In an earlier stage of the project a rapid prototyping language was considered but this option has been rejected in favour of Java. Given some negative experience with this language, in terms of sub-optimal performance, lack of portability and an insufficient design (features), Karel has been discussing alternatives with some experts in object-oriented programming. It is his strong belief that, given the object-oriented design of the current re-implementation, switching to another language is not a real problem.

Because we were dealing with a program that is very well documented, "which does not automatically mean that the subject at hand is easy and trivial", it is an interesting question why the re-implementation took so much effort. Since no systematic data has been gathered during the project, we will never know the complete answer to this question.

Another fact that became clear, especially in the final stage of the re-implementation, was that good old TEX runs much faster. The Java re-implementation is far more memory hungry and about 30 times slower in processing the TEXbook, and thereby much slower when used in large applications. When Knuth wrote TEX, department computers were much slower than today's desktops. So what exactly is slow? Anyhow, when one watches the page numbers appearing so slowly on the screen, one gets a good impression on how precise Knuth must have been in writing code in order not to waste much time waiting. One may argue that speed is not an issue, but evolving macro packages are getting more and more demanding and new features in the typesetting engine will ask for much more processing power.

I already mentioned that ε-TEX, Omega, and pdfTEX have been created by individuals but were developed with the help of users and experts. As a result, these programs are really used. The $\mathcal{NTS}$ project on the other hand has had a rather low profile. When the first alpha versions were made available, only a few people did a few tests. One reason for this is that the implementation is uncomfortably slow, is not as portable as the development environment promises, is not yet embedded in the existing file structures, and, most of all, does not offer anything new. I believe that there are also a few more reasons for this isolation on which I will elaborate later.

Some of the ideas behind the original project were to boost TEX into the future by providing a successor with more advanced features, as well as providing means to add a user interface. A third objective was that anyone could take the code and extend the program.

Even if we can envision those more advanced features, these are not goals that are reached fast. There are a few good ideas about areas of extensions. But to say for instance that, given a nice re-implementation, we can build a stable and full functional multi-column mechanism is a gross over-simplification of the problems at hand. Giving TEX a nice user interface is not by definition something that goes hand in hand with its batch processing character. And, how many people really understand the issues that TEX has to deal with to the extent that he or she can extend the program?

People use word processors for everyday tasks, and these programs have become better over time. In typesetting, WYSIWYG page layout programs have become more sophisticated, and some of the features that made TEX famous, like its paragraph builder, have made it into some of those. On the other hand, TEX is one of the few programs which can deal with today's document encoding formats, like for instance XML, in advanced ways. It is also one of a few programs that can handle database output with ease and speed. And, in the math arena it is still the best.

Times are changing, both in terms of demands and usage patterns. The main objective for a TEX successor is to provide better and more flexible general purpose routines to handle any input, typeset any document, in any language. In this respect the NTS project is far more ambitious than its predecessors ε-TEX, Omega and pdfTEX. But while all of these are already available, used and appreciated, the full NTS implementation is still a dream.

The status

One could expect that an effort like NTS would make other developments obsolete. But the opposite can be observed. Even after 20 years of TEX, user group meetings show that TEX is far from being dead. At such meetings, users often demonstrate new applications. They demonstrate specific ε-TEX, Omega or pdfTEX features and demonstrate new and advanced macros. When discussing those features, and possible future extensions, NTS never is part of the discussion.

In spite of being overloaded with official functions, the project team has not managed to get a good and promising reputation. In general, publicity has been handled at a bare minimum. And, even where the project is known, it is not so per definition in the positive sense.

One reason for this is that at a certain moment in time, politics entered the project. I must admit that I am only partially aware of the fine details of the political issues, since much of what I know comes from secondary sources. Surely some of the DANTE internal affairs influenced the project. On the other hand, the generous contributions and positive attitude of past and present DANTE boards towards the NTS project have ensured that at least the first main objective, the TEX re-implementation, has been achieved. Unfortunately the project lost some valuable German participants already in its early stage, which in my opinion has damaged the project.

I already pointed out that this project has quite a number of official tasks in its organization. Since I am participating in more "projects" than NTS alone, I can safely conclude that this has been counterproductive rather than productive. No other project in the TEX world has such a formal structure, no other project has spent so much user group money, and no other project has such a vague reputation as the NTS project. Instead of having a stronghold in the TEX community, this project has isolated itself beyond an acceptable limit.

I want to summarize the previous observations as follows:

- the NTS effort is largely unknown,
- the project is not really managed,
- the re-implementation is not embedded in research,
- the project objectives seem to be out of sync with reality,
- publicity has been handled badly or not at all, and
- the project is too isolated from other developments.

It may be clear that most of the conclusions result from the fact that the project was organized in such a way that the key players in the TEX community were only minimally involved. In this respect, I think that one way or another, the project became a hostage of its own structure. In spite of this, one of the objectives, namely the re-implementation of "TEX The Program" has been achieved. In the next section I will therefore elaborate on the future of the project as I see it.

The short term objective of the NTS project was to re-implement TEX. At the time of this writing, NTS can process the TEXbook. As Karel and I demonstrated at the DANTE October 2000 meeting, there is still a small problem in processing the METAFONTbook, and the trip test is passed largely, but not completely. Personally I presented the program with some more complicated situations and apart from a few not so dramatic bugs I am impressed by what Karel has achieved so far.

In the week before DANTE 2001 Karel announced that NTS has reached the beta stage. An important milestone was reached, namely that NTS can operate in the de facto standard TDS (the so-called `texmf` tree). From that moment on NTS could be really used as a replacement for traditional TEX.

In the continuous process of debugging, the programmer will also clean up some messy code, improve the performance where possible and document the source to the extent needed for further development. Because the team is very aware of the fact that users expect any TEX to be stable, and will expect the same from a re-implementation, the official release date is left to the programmer.

We can safely assume that in the summer of 2001 the code will be present in the TEX archives and part of distributions. At that moment we can start evaluating if the money spent so far has been worth it. This may be a good place to mention that the main official contributions to the project were from DANTE (85,000 DM), GUTenberg (3,000 EUR), TUG ($ 5,000), CSTUG (20,000 CZK for Karel's expenses), an unknown donor (5,000 DM), and the NTG (3,000 HFL) which means that until now the whole project has consumed over 100,000 DM. The finances were managed by DANTE, and the regular payments to the programmer went through Masaryk University in Brno (Czech Republic). This university also provided Karel with an email account and internet facilities, for which it deserves the team's gratitude.

By the way I want to note that at DANTE 2001 the membership decided to provide a regular budget for projects related in any kind to TEX, METAFONT, METAPOST and friends. For a couple of years, the NTG has had a similar budget for projects. The $\mathcal{NTS}$ project has demonstrated the need for such financing requirements. One obstacle has been the proper way to handle transactions in such a way that it fits into the tax regimes of the countries that are involved. This topic is a good candidate for the agenda of future cross-user group board meetings.

So, we can now safely conclude that:

- $\mathcal{NTS}$ version zero is there as a beta release, but still being debugged and cleaned up,
- some basic documentation will be provided,
- soon everyone can take the source and go ahead,
- so far the project has cost about 100,000 DM, and that
- thanks to Masaryk University we were able to transfer the money to the programmer.

Especially the fact that there is not much money left, causes the need to look into the future.

In recent publications in the GUTenberg magazine (spring 2000) and the TUG proceedings (fall 2000), some team members have drawn conclusions with regard to the project, its history, status and future. These conclusions were not discussed within the team, so a less informed reader could understand them as the voice of the whole team. Unfortunately, I don't share the views aired in those articles and, if I am right, also some other team members disagree. To state it clearly, the following section reflects my own thoughts and therefore should not be taken as the views of the whole $\mathcal{NTS}$ team.

The future

At a certain moment in time I got involved in discussions with regard to ε-TEX, which at that time were also related to $\mathcal{NTS}$. I must say that those discussions were quite interesting, and each proposal was considered in detail. Some made it into ε-TEX already, others could make it into future versions of ε-TEX, but those that were too complicated were put on the agenda for $\mathcal{NTS}$.

After a while, I got involved in the more ambitious $\mathcal{NTS}$ project, first as a reviewer for DANTE, later as a project member with the obligation to report to the DANTE membership about the progress of the project, since reporting had proven to be a weak spot of the project.

I have only been involved in the last stage of the project, a period when not many fundamental discussions were taking place within the team. Nonetheless, I carry pleasant memories of the discussions concerning the design that I had with Karel whenever I was visiting him in Brno. I saw it as my main contribution to make sure that this stage was finished and tried as hard as possible to be of help to him.

So, in the light of my experience, how do I see the future of $\mathcal{NTS}$, or to be more specific, how do I think a TEX successor should be developed? What lessons can be learned from the past, and how should we proceed?

I already remarked that the project is rather isolated from the rest of the TEX community and I see no indication that this will change soon. Given this, and given that I don't regard myself as being a real member of the $\mathcal{NTS}$ team any longer, if only because I am not one of the founding members, I feel that my role will be finished as soon as the first official release is there.

The language. I think that at this stage, the positive conclusion can be drawn that at least there is a working re-implementation, possibly with all the flaws that the language of choice imposed, but a major goal is reached. This means that we have a pretty good starting point for further development.

At a certain stage in the project, the decision was made to use the Java programming language. Such a decision is not easy, especially since everyone has his or her favourite language. At that time, Java was brand new and promising, and the public relations were good.

In every discussion I had so far, this choice is being highly criticized and not without reason. An interesting aspect is that when discussing alternatives, the availability comes up as a criterion. When $\mathcal{NTS}$ started the re-implementation, Java's future was yet unsure and portability was (and to some extent is) still an issue. Since we cannot foresee the future yet, any choice can be the wrong one.

In the current version of $\mathcal{NTS}$ some lines are commented out in order to let the program run on all platforms. In due time Karel will reflect on the re-implementation with respect to the language used and I'm sure that he will discuss how Java compares to other languages and how well it suits proper object-oriented programming.

I think that, in order to succeed, a group of very dedicated people is far more important than the programming language, especially if languages are chosen that compile to the heavily portable C language. It may even be of a certain charm if the language of choice is special, and very well-suited for the task. A strong belief in the virtues of a language is equally important to the success as dedication to high quality typesetting. It is my strong belief that the project should be directed by those who do the work. This is not to say that there is no need for advisers in any of the areas involved.

The fact that TEX was programmed in WEB and Pascal did not stop it from becoming available on nearly all platforms. An important aspect of Knuth's efforts was the documentation. Flagged as literate programming, the WEB system stimulates a particular way of programming. Programmers may like it or not, this has its charm, and it has certainly given TEX its place in the history of software development.

One thing that strikes me when people discuss a re-implementation of TEX, the language of choice is a major item. Of course we can wonder why we should keep on re-implementing TEX, and if re-implementing $\mathcal{NTS}$ is an issue, but at least I want to remark that the people involved in extending TEX should feel comfortable with the language that is used. There have been attempts to rewrite TEX, and I know of at least one other re-implementation project going on, but going from idea to full conception is not trivial, if not to speak of coming up with the right structuring for extensions. Current TEX has some flaws, but is nevertheless rather powerful (and often underestimated), so a successor had better be really good in order to succeed.

At TUG 2000 in Oxford, a number of the people involved in maintaining and extending TEX were present (among them some well known TEX experts like Hàn Thế Thành, Karel Skoupý, Fabrice Popineau, John Plaice). Since the descendants of TEX have all reached a more or less mature state, their creators shared their views on the future of TEX with the others experts present. Apart from the shared vision that those developments should converge in the near future, they all have strong opinions about the languages that are most suitable for a re-implementation. Most people involved in less trivial TEX programming agree on the fact that in order to extend, we need to re-implement. But in what language and in which architecture is a non-trivial decision.

Functional languages are the first choice, but this choice is more a (challenging) academic one, and it is understood that they are not the most stimulating candidates for users who want to extend TEX themselves. After some discussion, the language of choice was the object-oriented language Eiffel, which especially John Plaice considered to be a good candidate for a re-implementation of Omega.

Although I am completely new to this language, I cannot deny that reading the specs alone already gives me the good feeling that it suits such a project well. It compares to what I felt when for the first time I read the TEXbook, the METAFONTbook, the (real) books about Modula, SmallTalk, Lisp and the like.

But is a functional language, or a language with a vision like Eiffel the best choice on the long run? In this respect I owe much to Fabrice Popineau for sharing with me his balanced visions on the ideal languages versus practical languages (like C^{++}). Whatever the outcome of merging these efforts into the worthy and stable successor will be, I am sure that those talented people will make the right decisions with regards to the tools to use.

The design. Some time ago Karel and I discussed the viability to implement a successor in layers, like an efficient core in a pure imperative object-oriented language, a programming layer in a functional language, and on top of that the macro language. Whatever choices are made, the languages that are used should be able to interface to other languages. Especially pdfTEX demonstrates how useful it is to fall back on existing libraries, like those that deal with font embedding, bitmap and PDF inclusion and compression.

So, given that we can organize an enthusiastic group of people who want to spend time and effort on a successor, and given that we have a reasonable starting point in the well-organized TEX re-implementation called $\mathcal{NTS}$, there is a good chance that in the near future a real successor will show up.

At the TUG 2000 conference as well as preceding conferences the basis for cooperation has already been laid. But, we are talking of another project, with another name, this time properly embedded in the TEX community, and (again) carried by the user groups. Given the complexity of the typographic problems at hand, this should not be a naïve effort to come up with a collection of a thousand classes for everyone to extend, but a stable, flexible, and still extendable program, that can carry on the tradition started by TeX for another 20 years. As said, the existing extensions combined with the $\mathcal{NTS}$ redesign of TeX, provide a pretty good starting point.

Whatever course developments take, the results should be highly usable, (intermediate) distributions have to be stable, and the system should be open for future extensions. Of course it should also solve our most persistent typographic problems.

The environment. Another interesting development is that at TUG 2000 in Oxford, Karel was offered the opportunity to join the ETH in Zurich. There can be no doubt that a project like $\mathcal{NTS}$ or its successor will benefit from the possibility to embed it in proper research. We will learn more about those options when Karel has moved to Zurich (around the summer of 2001).

A result of a more close cooperation with the developers of TeX's multi-lingual follower Omega will also mean that developments can be related to the fundamental research that will follow the next release of Omega (this was presented at TUG 2000).

Apart from the fact that the (new) project could benefit from more fundamental research, an academic environment also gives access to all kind of resources. Given that for developers such environments can be inspiring in themselves, this will enlarge the chance of success.

The organization. One thing that can be learned from the current $\mathcal{NTS}$ project is that this is not the way to organize a project in the TeX community. The ε-TeX, Omega and pdfTeX projects demonstrate clearly how a successor can be developed successfully, while the $\mathcal{NTS}$ project demonstrates the contrary. And, at a much higher cost.

At TUG 2000, I have participated in discussions between the developers of pdfTeX and Omega and experienced programmers and users from the TeX community. To some extent, these discussions were a continuation of discussions at previous user group meetings and from email exchange.

For me, it is always a great experience to see how people share their ideas about future TeXs, the languages of choice, and the possibilities to integrate ideas. It demonstrates the real power of the TeX community when it comes to combining efforts. It also shows the way in which the next stage in developing a successful successor should take place.

One of the leading mottos of the $\mathcal{NTS}$ project is that "anyone can take the source and go forward". Given that the current team — except for the programmer — is not functioning in optimal form and seems to be unable to keep up its promises, this seems to be the right moment to take it at its word and start a new project.

Informal discussions at user group meetings have also demonstrated that it is quite possible to organize those who play a role in developments in a new team. I would not suggest this if I were convinced that the current team could be reorganized. Unfortunately there is too much historic ballast involved to guarantee success. Therefore I think that as soon as $\mathcal{NTS}$ version zero is released, the moment has come to start a new thread in the development of the successor. We need a fresh restart, run in such a way that user groups are involved in the proper way. We cannot do without a team, but apart from a group of people who can represent their user group, we also need dedicated teams for research, development and testing.

Let's do it. The current $\mathcal{NTS}$ team has managed to re-implement TeX in an object-oriented way, so in a sense it has accomplished its main objective. It is my strong belief that in order to achieve the more ambitious goals, a new team of enthusiastic and active people is needed. During the last couple of years I have received enough signals that such people are there waiting to get going.

At Bachotek 2001 as well as EuroTeX 2001 there will be $\mathcal{NTS}$ related sessions. Especially the (expected to be memorable) Bachotek meeting will provide the right ambiance to make such a fresh start. There, in the woods along the lake, team members Jerzy Ludwichowski and Karel Skoupý will present $\mathcal{NTS}$ in its full glory and invite us to discuss the future. I hope that you will be there too.

◇ Hans Hagen
Hasselt, The Netherlands
October 2000 – March 2001
`pragma@wxs.nl`

The Treasure Chest

Packages posted to CTAN

"What's in a name?" I did not realize that Jan Tschichold's typographic standards lived on in the `koma-script` package often mentioned on usenet (in `comp.text.tex`) until I happened upon the listing for it in a previous edition of "The Treasure Chest". This column is an attempt to give TeX users an ongoing glimpse of the trove which is CTAN.

This is a chronological list of packages posted to CTAN between June and December 2000 with descriptive text pulled from the announcement and edited for brevity — however, all errors are mine. Packages are in alphabetic order and are listed only in the last month they were updated. Individual files / partial uploads are listed under their own name if so uploaded. If not otherwise noted, packages are in `macros/latex/contrib/supported/`. Subdirectories (e.g. `foo`) of `macros/latex/contrib/` are listed as `.../foo/` to save space.

Corrections and suggestions are welcome.

June 2000

`abstract` Gives you control over abstracts, and in particular provides for a one column abstract in a two column paper.

`amsclass:` in `macros/latex/required/amslatex/classes`
AMS-LaTeX document classes and theorem package. Bug-fix updates.

`arrayjob:` in `macros/generic/arrayjob`
The `arrayjob` package provides array data structures in LaTeX, like those in Fortran, Ada or C, and macros to manipulate them.

`catdvi:` in `dviware`
DVI to plain text translator aiming to replace `dvi2tty`.

`dvichk:` in `dviware`
(V.1.91) Checks `.dvi/.log` files and displays the page numbers found on standard output.

`epsfview:` in `systems/mac`
AppleScript tool (for Mac) mainly intended for viewing figures generated with METAPOST, even if they have negative coordinates.

`geometry` (V.2.3) An easy and flexible user interface to customize page layout. Update adds `columnsep` and `footnotesep` options, `vtex` option is added to support VTeX, sophisticated magnification setting.

`ifmslide` A package (v.0.3) from the Institute of Mechanics (ifm) Univ. of Technology, Darmstadt, Germany, for producing slides with LaTeX 2ε. Based on the concept of `pdfslide`, but completely rewritten for compatibility with `texpower` and `seminar`.

`ifsym:` in `fonts`
Fonts with symbols for alpinistic, electronic, meteorological, geometric, etc., usage. A LaTeX 2ε package simplifies usage.

`jas99_m.bst:` in `biblio/bibtex/contrib`
Update of `jas99.bst`, modified for better conformity to the *American Meteorological Society*.

`LaTeX WIDE:` in `nonfree/systems/win32/LaTeX_WIDE`
A demonstration version of an integrated editor and shell for TeX — free for noncommercial use, but without registration, customization is disabled.

`lhelp:` LaTeX 2ε macro package of simple, "little helpers" converted into `dtx` format. Includes common units with preceding thinspaces, framed boxes, start new odd or even pages, draft markers, notes, conditional includes (including EPS files), and versions of `enumerate` and `itemize` which allow spacing to be changed.

`makecmds` Provides commands to make commands, environments, counters and lengths. Moved from `.../misc` to `supported`.

`mathpazo:` in `fonts`
Package (fonts and LaTeX style file) for mathematical typesetting with the Palatino fonts.

`secdot:` in `.../other/misc`
Section numbers with trailing dots.

`substr` Provides commands to deal with substrings in strings: Determine if a string contains a substring, count appearances of a substring in a string.

`texdoctk:` in `systems/unix/teTeX/1.0/contrib`
(V.0.4.0) A Perl/Tk-based GUI for easy access to package documentation; the databases it uses are based on the `texmf/doc` subtrees of teTeX v.1.0.x, but database files for local configurations with modified/extended directories can be derived from them. Update adds: view multiple documents simultaneously, UI improvements, internal text viewer, uses `kpsewhich`, bug fixes, manpage.

`TeXnicCenter:` in `systems/win32`
An integrated development environment (IDE) for developing LaTeX documents in Windows (95, 98, 2000, NT 4.0). Project oriented, unlimited "output types", customizable editor, structure View, etc.

`titling` Provides control over the typesetting of the `\maketitle` command, and makes information from it permanently available.

`tugboat/t-of-c:` in `digests`
`tb1594.cnt`, `tb1695.cnt`, `tb1998.cnt`, `tb2099.cnt` and `tbcv20.tex`.

`varindex.sty:` in `.../misc`
Provides a convenient front-end for the `index` command. For example, it allows generation of multiple index entries in almost any form by a single command. Extremely customizable. Works with all versions of LaTeX and probably most other TeX formats, too.

`WinShell:` in `systems/win32/`
Bug fix to WinShell, a graphical user interface for easily working with TEX. It is *not* a TEX system so requires one to have a system such as MikTEX or TEXLive installed.

July 2000

`amsmath:` in `macros/latex/required/amslatex/math`
(V.2.13) Bug fix for an equation numbering problem with the `hyperref` package.

`arydshln` New version (1.4). Extension of the `array` package which allows dashed lines/rules in tables, and control over the spacing of vertical rules.

`bbl2html:` in `biblio/bibtex/utils`
Converts a LATEX `.bbl` file to (mostly) formatted HTML code. Probably also works if applied directly to a `.tex` file.

`bytefield` The `bytefield` package helps the user create illustrations for network protocol specifications and anything else that utilizes fields of data. These illustrations show how the bits and bytes are laid out in a packet or in memory.

`chngpage:` in `biblio/bibtex/utils`
(V.1.1) Provides commands to change the page layout in the middle of a document. Now uses empty arguments instead of zero lengths, new `adjustwidth` environment for extra wide (or narrow) paragraphs or over-width floats

`combine` (V.0.41) Bundles individual documents into a single document, such as when preparing a conference proceedings. The auxiliary `combinet` package puts the titles and authors into the main document's Table of Contents. Now cooperates with the `abstract` and `titling` packages.

`CWEBbin:` in `web/c_cpp`
(V.3.61) A set of change files (to be applied with the TIE processor) that make the original sources usable with ANSI-C/C++ compilers on UNIX/Linux, MS Windows, and Amiga. Now fully supports "HyperCWEB".

`Fontmap.cmr:` in `fonts/cm/ps-type1/contrib`
Makes Computer Modern available to ghostscript.

`french:` in `languages`
Light version has a maximum of automatic features: translation, layout, microtypography, etc., but no specific commands are available.

`Hexdump.sty:` in `macros/generic`
Reads an ASCII hexdump file and puts it formatted into the document. Additional macros included, e.g., for a Directory of Dumps.

`html2text:` in `support`
HTML-to-text converter for UNIX, which, unlike `lynx-dump`, handles tables.

`koma-script`
Reimplementation of the LATEX classes (`article`, `report`, `book`, `letter`), "implementing European rules of typography and paper formats as documented in Tschichold (*Selected Papers on Book Design and Typography*)."

`newvbtm` Allows one to define one's own `verbatim`-like environment with variants.

`notoccite.sty:` in `.../other/misc`
Prevents erroneous numbering of the cites in the `toc` or `lof` or `lot` when using `bibtex/unsrt`.

`ntabbing:` in `.../other`
An extension of the tabbing environment that supports automatic line numbering with referencing.

`patchcmd` Provides a command `\patchcommand` that adds materials at the beginning and/or end of the replacement text of an existing macro.

`Portuguese-Portugal:` in `systems/win32/winedt/dict`
WinEdt dictionary — Portuguese (Portugal version).

`poster-os2:` in `support`
A program to generate large-size and multi-sheet posters from PostScript files. Also works in DOS.

`pstoedit:` in `support`
(V.3.20) Converts PostScript and PDF files to other vector graphic formats so that they can be edited graphically.

`scientificviewer:` in `nonfree/systems/win32`
Scientific Viewer 3.5 is a free program for reading and printing (read-only) documents created with Scientific Notebook, Scientific Word or Scientific WorkPlace by MacKichan Software, Inc. Scientific Viewer can also be used to view many native LATEX documents.

`semantic` (V.2.0 alpha) Adds customizable math mode ligatures; support for typesetting reserved words in a consistent way and the drawing of inference rules has been substantially improved, allowing deeper nesting.

`stdclsdv` (V.1.1) allows package writers to learn what sectioning divisions are provided by the document's class.

`Swiss-German:` in `systems/win32/winedt/dict`
WinEdt dictionary — Swiss-German.

`ticket` Provides an easy interface to produce visiting cards, labels for your files, stickers, pins and other stuff for your office, conferences, etc.

`umoline` Allows underlines while allowing line breaking.

`wotree.sty:` in `graphics/dratex`
Supports drawing of Warnier/Orr diagrams. Documentation is at `http://www.cis.ohio-state.edu/~gurari/wo-diag/wo-diagrams.html`.

`yafoot` Contains three style files: `pfnote.sty` to enclose footnote numbers within a page;
`fnpos.sty` to control the position of footnotes;
`dblfnote` to make footnotes double-columned.

August 2000

`accents`: in `.../bezos`
(V.1.2) Quick fix in `accents` to work with `amsmath`.

`accfonts`: in `fonts/utilities`
V.0.212 of the program `vpl2vpl` (bug fix), updated `CHANGES` file.

`amslatex-primer`: in `info`
An updated version of "Getting up and running with AMS-LaTeX". An attempt to give enough information for a newcomer to LaTeX (but with some plain TeX or AMS-TeX experience) to quickly be able to make use of AMS-LaTeX. Updated to current version (LaTeX 2ε).

`BioCon` Typesets some biological entities. Initial version (0.01) which only typesets species.

`circuit_macros`: in `graphics`
(V.4.9) A set of macros for drawing high-quality electric circuit diagrams containing fundamental elements, amplifiers, transistors, and basic logic gates. Several tools and examples for other types of diagrams are also included. More robust NOT_gate, Function `pmod`(), macro `shade`(), etc.

`cuisine` Package for typesetting recipes in steps in which each ingredient is on the left of the page next to the method step in which it is used.

`curves` Draws curves in the standard LaTeX picture environment using parabolas between data points with continuous slope at joins. V.1.42 (bug fix) can still be used with LaTeX 2.09. V.1.50 can only be used with standard LaTeX. It has improved documentation, greater drawing accuracy and is more efficient. Four package options for `dvi` specials improve drawing performance and some specials work with color.

`directory`: in `biblio/bibtex/contrib`
(V.1.15) Adds cellular phone fields, flexible definition of headers to generate nicer address booklets, use of `hyperrref` to produce directories with hyperlinks.

`dvi2bitmap`: in `nonfree/dviware`
A utility to convert TeX `dvi` files directly to bitmaps.

`dvipsconfig`: in `dviware`
(V.1.5) A set of PostScript header files for `dvips` to control various printer functions such as paper size, duplex, and paper source (e.g., manual feeder, envelope feeder, and trays 1, 2, and 3). Adds paper size `usledger` and `addpsctrl`, which inserts these controls into existing files.

`expressg`: in `graphics/metapost/contrib/macros`
(V.1.4) Provides facilities to assist in drawing diagrams that consist of boxes, lines and annotations. Although particular support is provided for creating EXPRESS-G diagrams, examples are provided of IDEF0, IDEF1X, Shlaer-Mellor, E-R, OMT, NIAM, and other diagrams. Update adds outines for generating piecewise linear paths with either sharp or rounded corners, diagram showing labeling convention for a drum and general tidying up of the code.

`germbib`: in `biblio/bibtex/contrib`
Now supports the language packages `ngerman.sty` and `babel.sty`.

`isodate` Tunes the output format of the `\today` command providing `\isodate`, `\numdate`, `\shortdate`, `\TeXdate` and `\origdate` and two additional commands to print a date argument using the actual date format for output. Compatible with `bibgerm` style file.

`jkthesis` Updated class for formatting a thesis (bug fix and now has ASCII-encoding). Documentation in German only.

`LaTeX 2000/06/01` The last (nominal) 6-monthly release of LaTeX 2ε. Further updates will be on an annual basis.

`latex2man`: in `support`
(V.1.11) Bug-fix update. A tool to translate UNIX manual pages written with LaTeX into a format understood by the UNIX man(1) command.

`listbib` Lists the contents of `.bib` files in a printable format. V.2.2, enhancements to the `listbib` shell script, example `listbib.cfg`, support for entry fields URL and 'totalpages' (`custom-bib` uses them).

`MiKTeX`: in `systems/win32/miktex/2.0-beta`
V.2.0 beta 4 update. MiKTeX is a free TeX distribution for Windows.

`oands`: in `fonts/archaic`
METAFONT version of some odd characters that are used as symbols when transliterating ancient scripts (like Hieroglyphs).

`springer`: in `obsolete/macros/plain/contrib`
Springer-Verlag Heidelberg/Berlin officially pulled the macros in `macros/plain/contrib/springer`, hence the move to obsolete. Up-to-date LaTeX files are at: `macros/latex/contrib/supported/springer`.

`tcltexed`: in `support`
An editor written in the scripting language Tcl/Tk for writing LaTeX documents.

`textcomptst`: in `info/textcomp-list`
A list of all symbols available using the package `textcomp.sty`.

`textmerg`: in `.../other`
Minor problem fixed and placed in public domain. Word-processor-style merges for TeX and LaTeX.

`truncate.sty`: in `.../other/misc`
Allows truncation of horizontal (lr mode) text to a specified width. New version allows truncation at arbitrary characters, without loading any special hyphenation patterns.

`tth-2.7-3.i386.rpm`: in `support`
Version update to 2.7-3. TeX (or LaTeX) to HTML converter. Includes equations using native fonts in tables.

`TVS`: in `support/TVS`
TeX Versioning System — Perl script collects complete source codes of TeX documents in order to be

able to re-typeset them exactly the same way. New version 1.0 contains better support for packing format file sources and improved documentation (in TEXinfo format).

`undertilde` Provides an `\utilde` command that puts a tilde under math material.

`vertbars` An extension to the `lineno` package to put vertical rules at left (right) of lines instead of numbers. Only complete paragraphs can be barred.

September 2000

`adobe.zip:` in `.../psnfssx`
Replaces files (`alucida.dtx/ins` & `adobe.dtx/ins`) which were "lost" during restructuring the PSNFSS distribution.

`ams2bib:` in `biblio/bibtex/utils`
A Perl script that translates $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX references into BIBTEX.

`archaic:` in `fonts`
Samples of archaic fonts.

`blindtext:` in `.../minutes`
Creation of text to get an impression of the look of classes.

`bsamples:` in `fonts/archaic`
Sampler showing all 5 of the bookhands fonts.

`CBGreek:` in `systems/win32/bakoma/contrib`
CBGreek fonts (Small Caps) in Type 1 font format.

`dvipscol:` in `.../oberdiek`
Fix color stack overflows by `dvips` (ε-TEX recommended).

`elsart.cls:` in `.../elsevier`
Elsevier class for compuscript (as opposed to manuscript) submission of publications. License clarification (now LPPL).

`engord:` in `.../oberdiek`
Converts numbers to English ordinal numbers.

`fac` LATEX 2ε class file and guide for the Springer Verlag journal *Formal Aspects of Computing*.

`fltpoint` V.1.0c is an update needed for the new version of the `rccol` package with minor changes to the documentation. Provides basic floating point calculations inside TEX.

`fncylab.sty:` in `.../misc`
Package allows customizing the appearance of labels (without modifying internal commands), and provides a `\labelformat` command for changing the format of references to labels.

`hieroglf:` in `fonts/archaic`
METAFONT rendition of some 60 plus Egyptian Hieroglyphs (used between 3000 BC and 100 AD), and accompanying files to use them in LATEX. (5 new glyphs.)

`hypcap:` in `.../oberdiek`
Fix for using `hyperref` with figures, table environments and the star forms.

`inslrmaj:` in `fonts/archaic`
METAFONT rendition of the Insular Majuscule bookhand in use in Ireland (and England) between the sixth and ninth centuries, and the necessary files for use with LATEX.

`jtbnew.bst:` in `biblio/bibtex/contrib`
Bibliography stylefile for the *Journal of Theoretical Biology*'s new style.

`latex2rtf-1.8aa-os2.zip:` in `support`
OS/2 port of `latex2rtf` converter. Needs `emxrt.zip`, version 0.9d or higher.

`lh:` in `fonts/cyrillic`
Bug-fix update (v.3.4b). Cyrillic fonts including characters for languages often not supported such as Kazakh.

`listings` LATEX source code printer. Language update.

`listliketab` Makes list-like tabulars so that the user can add additional columns to each entry.

`lshort:` in `info/lshort/italian`
The Italian version of `lshort` (*The Not So Short Introduction to LATEX 2ε*), version 3.15.

`lshortth:` in `info/lshort`
Thai translation of `lshort` (*The Not So Short Introduction to LATEX 2ε*).

`MetaPost4OS2:` in `graphics/metapost`
Replaced with a symbolic link to the MS-DOS version.

`mfpic:` in `graphics`
(V.0.4 beta) A pdfTEX package for drawing pictures using METAFONT or METAPOST commands.

`MHequ:` in `.../other`
A package to easily create multicolumn equation environments, and to tag the equations therein. New version fixes compatibility problems with the packages `showkeys` and `hyperref`.

`minutes` A package (v.1.4b) for setting minutes and building collections of minutes.

`model-harv.pdf:` in `.../elsevier`
Instructions for use and template files for Elsevier authors.

`model-num.pdf:` in `.../elsevier`
Instructions for use and template files for Elsevier authors.

`mv4vtex.zip:` in `fonts/psfonts/marvosym/vtex`
Instructions for installing the Marvosym font for use with VTEX/Free and a font map file.

`nomencl` Help formatting a nomenclature using MakeIndex. V.3.1 adds support for Croatian, more examples and the `ins` file creates some sample configuration files.

`numprint` Prints numbers with separators and exponent if necessary. New version includes an option for units.

`oldprsn:` in `fonts/archaic`
METAFONT and LATEX files for typesetting with the Old Persian cuneiform font, in use between 500 and 350 BC.

`othello` Creates othello boards using LaTeX.

`pdfcolmk:` in `.../oberdiek`
Fixes color problems with pdfTeX at page breaks.

`pdftex.def:` in `macros/pdftex/graphics`
(V.0.3e) Adds viewport and trim with clip support.

`protosem:` in `fonts/archaic`
METAFONT rendition of a Proto-Semitic script used about 1600 BC in the Middle East, and the necessary files for use in LaTeX.

`rccol` (V.1.1) Supports different decimal signs in input and output. Provides 'right-centered', optionally rounded numbers in tabulars.

`refcheck` (V.1.8) Fix of bug when `\ref`, `\pageref` occurred in a material processed to `\write`. Also improved compability with AMS-LaTeX and `HyperRef`.

`refcount:` in `.../oberdiek`
Extracts the numbers from references.

`settobox:` in `.../oberdiek`
Defines commands similar to LaTeX's `\settowidth` commands for boxes.

`template-harv.tex:` in `.../elsevier`
Instructions for use and template files for Elsevier authors.

`template-num.tex:` in `.../elsevier`
Instructions for use and template files for Elsevier authors.

`tfmpktest.pl:` in `fonts/utilities`
Checksum of `pk` fonts. Can delete bad `pk` font with checksum mismatch and recreate it.

`thumbpdf:` in `support`
The package provides support for thumbnails with pdfTeX, and plain/LaTeX formats. Requirements: Perl5, ghostscript, pdfTeX. V.2.5 reduces filesize by only loading fonts once.

`ugarite:` in `fonts/archaic`
METAFONT and LaTeX files for typesetting with the Ugaritic Cuneiform script (dating from about 1300 BC).

`vpe:` in `.../oberdiek`
Enables source specials for `pdf` files (clicking on special annotations will launch an editor with the source file at the source line) in VTeX/Linux, pdfTeX, and with dvips.

`wordcount` Provides a relatively easy and accurate way of counting the number of characters and words in a LaTeX document.

October 2000

`Augie:` in `fonts`
A Type 1 handwriting font, based on American style casual handwriting. The font itself is freeware by emerald city fontwerks `www.speakeasy.org/~ecf`. The package adds metrics, `.vf`, `.fd` and `map` files.

`BibGene:` in `biblio/bibtex/utils/bibgene`
Macintosh bibliographic database program; freeware application for maintaining databases of references. `BibGene` 1.2.2 is PowerPC-only. For versions for M68K Macs, see `http://www.ics.uci.edu/~eppstein/bibs/bibgene/`.

`ConcProg` Sets music concert programmes, with support for part divisions, etc.

`coordsys` Provides commands for typesetting number lines (coordinate axes) and coordinate systems in the picture environment.

`CWEB3.6.2forMacOS:` in `web/systems/mac/cweb`
Improved port of CWEB (no fake `/dev/null` files any more) for MacOS v.3.6.2.

`ecta.bst:` in `biblio/bibtex/contrib/economic`
Econometrica bibliography style — now follows exactly the official recommendations. Also compatible with `natbib`.

`emTeXTDS:` in `systems/os2/emtex-contrib`
Version 0.54 of the emTeX/TDS distribution for OS/2, featuring the latest releases of LaTeX, AMS-LaTeX and other packages, as well as several bug fixes.

`fancyhdr` Update of the `fancyhdr` package: added LPPL license, some small enhancements (error messages), some bug fixes in `extramarks.sty`, obsolete file removed and documentation updated

`hpsdiss` Class file (documented in `.dtx`) developed by Hanspeter Schmid to typeset his dissertation. `.pdf` sample provided.

`hyperref:` (V.6.71) Bug fixes and enhancements of this package for creating/enabling hyperlinks in documents.

`isi2bibtex:` in `biblio/bibtex/utils`
(V.0.40) `Isi2bibtex` converts an Institute for Scientific Information (ISI, known in the UK as 'BIDS' or 'MIMAS WoS') bibliographic database file to a BibTeX file for use with TeX and LaTeX

`jurabib` Various forms of short and long citations — now more flexible and no longer just for German law students. Changes in 0.5d: English documentation, option to place the `howcited` remark for all entries `french.sty`-compatibility, `keyval`-interface more intuitive, minimal example document for the humanities, etc.

`KTeXShell:` in `systems/unix`
A graphical user interface to TeX, LaTeX, and related programs, running on Linux/UNIX with KDE. It is not another WYSIWYG approach. Instead it provides a document development interface where you can define your files, edit, compose, view, and print them with a mouseclick.

`logsys` Extends the `oordsys` package by providing logarithmic, semi- and double-logarithmic coordinate systems and grids.

`mathetx/mathmtx:` in
`fonts/utilities/fontinst-prerelease/inputs`

A partial collection of metric and encoding files for `fontinst` v.1.915.

`macos-fonts.zip`: in `systems/mac/fonts/oztex`
Collection of freely available PS fonts in MacOS format. Includes: Belleek, CharterBT, LOGO, MANUAL, MathPazo, RSFS and Utopia. Can be adapted for use with other TEX systems in the MacOS.

`NRC` Macros for typesetting papers for submission to journals published by the National Research Council of Canada.

`rfc2bib.awk`: in `biblio/bibtex/utils`
gAWK script to automatically generate BIBTEX entries from IETF RFCs.

`skak`: in `fonts`
Package for typesetting chess. Bug fix (regarding knight and queen movement).

`TeXshade` Package (1.4a update) for setting nucleotide and protein alignments in LATEX. Features new possiblities for positioning the legend and some changes in the documentation and the FAQ-list.

`titleref.sty`: in `.../other`
Version 3.0. Provides a `\titleref` command to cross-reference section (and chapter, etc.) titles and captions like `\ref` and `\pageref`. Now licensed as PD.

`verbdef.sty`: in `.../other`
Define robust commands which expand to verbatim text — can be used in arguments for other commands/macros.

November 2000

`arabtex-oztex.sit`: in `systems/mac/fonts/oztex`
A MacOS adaption of Taco Hoekwater's PostScript Type 1 version of Dr. Prf. Klaus Lagally's Nastaliq arab fonts, with additional support for OzTEX.

`bibtopic` A package for sectioned/multiple bibliographies. Update to version, 1.0j, fixes a problem with `babel.sty`'s redefinition of `\ifthenelse`. V.1.0e adds compatibility with `hyperref` and another update added compatibility for `natbib` v.7.0.

`dinat`: in `biblio/bibtex/contrib/german`
Enhancement of the old dinat-style for proceedings, inbook and misc. Improves handling of names to avoid problems with the "von" part.

`filippou`: in `language/greek/package-babel/hyphenation`
Update of the hyphenation patterns for ancient and modern Greek in polytonic (multi-accent) and monotonic (uni-accent) systems. Works with the `greek` option of `babel` or Dryllerakis' GreeKTeX.

`jcc.bst`: in `biblio/bibtex/contrib/chem-journal`
BIBTEX style file for *Journal of Computational Chemistry*.

`jpc.bst`: in `biblio/bibtex/contrib/chem-journal`
BIBTEX style file for *Journal of Physical Chemistry*.

`keystroke` Provides macros for the graphical representation of the keys on a computer keyboard.

`LatexHelpBook`: in `info`
HTML-based help for LATEX for Windows 98, 95, NT 4, and 5.

`mpic21.zip`: in `graphics/pictex/mathspic`
`mathsPIC` (DOS vers 2.1), a DOS filter program for use with PiCTEX.

`pccp.bst`: in `biblio/bibtex/contrib/chem-journal`
BIBTEX style file for the journal *Physical Chemistry Chemical Physics*.

`Prosper` A LATEX class on top of the `seminar` class. Permits easily writing slides for both printing and display with a video projector, including animation effects such as incremental display. Several slide styles are available and new ones are easily added.

`revcompchem`: in `biblio/bibtex/contrib/chem-journal`
BIBTEX style file for *Reviews in Computational Chemistry*.

`rtf2latex2e`: in `support`
Beta release. `rtf2latex2e` 1.0 beta 3 converts RTF files into LATEX 2_ε. Detects text style (bold, italic, color, big, small, etc.), reads tables (simple to semi-complex), converts embedded MathType equations, converts most Greek and math symbols, reads footnotes (not in tables), support for use of the `fontenc` package, translates hyperlinks using the `hyperref` package.

`TransFig`: in `graphics`
TransFig is a set of tools for creating TEX documents with graphics which are portable, in the sense that they can be printed in a wide variety of environments.

`ucs.sty` This package implements a facility for mapping Unicode characters to LATEX macros and to use UTF-8 as an input encoding with standard LATEX 2_ε.

`xfig`: in `graphics`
Xfig is a menu-driven tool that allows the user to draw and manipulate objects interactively in an X window. The resulting pictures can be saved, printed on PostScript printers or converted to a variety of other formats.

`xipa-oztex.sit`: in `systems/mac/fonts/oztex`
A Stuffit 5 archive containing MacOS versions of the PostScript Type 1 XIPA fonts created by Taco Hoekwater. These fonts provide IPA93 characters plus extensions to be used with Times Roman and Helvetica for use with Fukui Rei's `tipa` package.

`XyMTeX` Update. Package including LATEX documentstyle options for typesetting chemical structural formulas.

`ziffer` Formats numbers with the correct German spacing (even in math mode).

December 2000

`bakoma`: in `systems/win32`
BaKoMa TEX system upgrade to v.3.10.

`bakoma/index.html`: in `systems/win32/bakoma`
Updated version of this file with new information about the BaKoMa TeX mailing list which is intended for news, questions, answers, and user discussions about BaKoMa TeX Software.

`BibTexMng`: in `biblio/bibtex/utils`
Easy to use bibliographic software for Windows. Combines online searching, reference management, bibliography making, and information sharing into a single user-friendly environment. It was written to be used with LaTeX, using BibTeX.

`bophook` (V.0.01) Hook for adding material at the beginning of each page.

`bpchem`: in `.../other`
This package provides commands and environments, which are useful for typesetting chemical formulæ as well as breakpoints in multiline chemical names and a set of commands to enumerate/reference chemical substances.

`bundledoc`: in `support`
Post-processor for the `snapshot` package that bundles together the classes, packages, and files needed to build a given LaTeX document into a `.tar.gz` file, suitable for moving across systems, transmitting to a colleague, etc.

`docmfp` Update to v 1.1 of the `docmfp` package. This extends the `doc` package to document non-LaTeX code, such as METAFONT and METAPOST, or even C code. Adds a generalized `\Describe` macro and a `Code` environment.

`dvi2tty-german_umlauts.patch`: in `nonfree/dviware/dvi2tty`
With this patch, `dvi2tty` replaces some special characters (a-,o-,u-umlauts, sz-ligature, a-ring) by their Latin-1 representation.

`eso-ex3.tex`: in `.../ms/contrib`
Example for the LaTeX package `eso-pic` showing how to include pages from an external (PDF) document.

`fancyhdr`: in `/info/german`
A short german introduction to `fancyhdr` with examples.

`faq`: in `usergrps/uktug`
V.2.5 of the FAQ by the UK TeX Users Group, also available from `http://www.tex.ac.uk/faq`.

`FiNK` The LaTeX 2ε File Name Keeper, first public release. This package keeps track of files `\input`'ed (the LaTeX way) or `\include`'ed in your document. Also comes with support for AUC-TeX.

`FiXme` A LaTeX 2ε package for inserting "fixme" notes in draft documents. V.1.1, the first public release — provides a way to insert "fixme" notes in draft documents, either in the margin of the document, as index entries, in the log file and/or as warnings on stdout or to summarize them in a list. Also comes with support for AUC-TeX.

`itamsldoc`: in `info/italian`
Italian translation of the `amsmath` documentation.

`latex.zip`: in `systems/win32/miktex/1.20`
LaTeX 2ε update (2000/6/1) for the MiKTeX 1.20e distribution.

`LigaTeX`: in `support`
Version 0.2.0 of LigaTeX (a package which removes certain ligatures from text). Now switched on and off by the `babel`-package macros.

`LGrind`: in `nonfree/support/lgrind`
V. 3.65. Last update by Michael Piefel (due to non-free license issues) for the pretty printer `LGrind` which can produce nice LaTeX from source code. Moved from `support/lgrind` (but a symbolic link to that location has been added).

`lineno.sty` Update for compatibility with `longtable`.

`ltx3info.pdf`: in `.../ms/contrib`
Article describing the motivation, achievements and future of the LaTeX3 Project.

`MacDevnag`: in `language/devanagari/contrib`
Macintosh port of the `devnag` program — a preprocessor which is part of the Devanagari package (in `language/devanagari/distrib`).

`minitoc` Update to version 34. This package adds mini tables of contents by chapters, parts or sections (and minilofs, minilots). Updated documentation (a section for use with `tocbibend` and a French translation (`fminitoc.ps`) and additional `.mld` files for alternate names of languages.

`mtgreek` A `mathtime.sty` supplement, uppercase Greek letters displayed as either roman or italic glyphs.

`pxfonts`: in `fonts`
Final update to v.3.1 of the PX fonts (based on Palatino). Only obvious bugs, if they are found, will be fixed through patches, no modification to or adding of letters/symbols in the near future.

`references`: in `support`
Updated to v.3.6. `references` is bibliographic software for authors of scientific manuscripts and for management of bibliographic data. Supports LaTeX including BibTeX. Able to import bibliographic records in the MEDLINE format.

`rtkinenc` Similar to the standard package `inputenc`, but allows the user to specify a fallback procedure to use when the text command corresponding to some input character isn't available.

`scriptfonts`: in `info/symbols/math`
A summary of readily-available script fonts for use in mathematical typesetting.

`setspace.sty` The `setspace` package, version 6.7, has three new options, to set default spacing (`singlespacing`, `onehalfspacing`, `doublespacing`) when the package is loaded.

`tcldoc` Defines a couple of environments and commands for documenting Tcl (Tool Command Language) source code in `.dtx`-style documented source files.

`TeXmacs:` in `systems/unix`
: Update to v.0.3.0-7 of GNU TeXmacs, a text editor inspired by the popular TEX typesetting system and the emacs editor. Runs on PCs under Linux and on SUN computers: "It is reasonable to expect that it will run on most UNIX/X-Windows systems in the near future."

`trsym:` in `fonts`
: Horizontal and vertical symbols used for transformations (e.g. Laplace transformation) including inverse transformation.

`txfonts:` in `fonts`
: Final update to v.3.1 of the TX fonts (a complete set of fonts with math support based on Times and Helvetica). Only obvious bugs, if they are found, will be fixed through patches, no modification to or adding of letters/symbols in the near future.

`u8tex.el:` in `support/emacs-modes`
: Quail package to input TEX characters (and some more) using familiar notation. Changes in v.1.2: Documentation, adoption of some HTML abbreviations, more `amssymb` characters and bug fixes.

`varindex` Bug fix update.

`WinEdt:` in `nonfree/systems/win32`
: Official release of WinEdt 5.2 (shareware). Complete information at `http://www.winedt.com`.

`xdoc2:` in `macros/latex/exptl/xdoc`
: Second prototype for the hypothetical `xdoc` package. Reimplements some of the features found in the standard LATEX `doc` package. Additionally provides support for defining new commands similar to `\DescribeMacro` and new environments similar to the macro environment, for two-sided document layouts, for external cross-referencing, for making index entries for invisible characters, and for optionally ignoring certain prefixes (such as `@` and `@@`) in macro names when sorting them.

`zefonts:` in `fonts`
: New version of `zefonts` with two new `fd` files and a comparison between `aefonts` and `zefonts`.

◇ William F. Adams
75 Utley Drive, Ste. 110
Mechanicsburg, PA 17055
USA
`willadams@aol.com`

Tutorials

Publishing legacy documents on the Web

George Grätzer

Abstract

A great deal has been written recently about publishing LaTeX documents on the Web. But what happens if you are not lucky enough to have your document in LaTeX?

I am going to describe my adventures putting old documents on the Web. There are a few pitfalls on the way. If you follow this how-to guide, you can get your legacy articles on the Web in no time at all.

1 Introduction

I have made an effort in the past few years to make all my mathematical research articles available on the Web. If you check my Web site: `http://www.math.umanitoba.ca/homepages/gratzer/` you find all my articles, 165–202, in PDF format. The turning point was 1994, when David Carlisle's graphics package came into use. After 1994, I wrote all my articles in standard LaTeX and I included the diagrams—saved in EPS format—with the

```
\includegraphics
```

command of the graphics package. Now, seven or so years later, LaTeX is the same, EPS is the same, all the articles 165–202 typeset today as they did when they were written.

If you want to read about how to publish such documents on the Web, read Chapter 14 of my book [2]; if you want to read a whole book on the topic, read Michel Goossens and Sebastian Rahtz (with Eitan Gurari, Ross Moore, and Robert Sutor) [1]. The best book on the technical aspects of PDF is Thomas Merz [3].

I would like to thank R. Padmanabhan, Jacob Palme, and Thomas Merz for reading the manuscript and giving good advice.

2 Legacy documents

All my articles written BC (Before Carlisle) are legacy articles—with a few exceptions.

Here are the major categories of legacy documents:

1. Old documents written on a typewriter and then typeset in a printing shop. All my articles written before 1990 (1–132) fall into this category.

2. Documents written in an old word processor that is no longer available or in an old version of a word processor that went through too many changes. For instance, my papers that were written in Word 6.0/1995 (on the Mac) would need significant editing: many symbols appear wrong (was there a change in the encoding vector of the Symbol font?) and they are full of mysterious messages:

   Error! Bookmark not defined.

   You find lots of examples of this in the group 1990–99.
3. LaTeX papers whose source code has been misplaced; LaTeX papers that utilize packages or document classes that are no longer available or that are not compatible with today's LaTeX.
4. LaTeX papers with diagrams drawn with drawing programs that are no longer around. For instance, article 161.
5. LaTeX 2.09 and $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX papers. Although Appendix G of [2] gives detailed and pretty straightforward instructions on how to convert such articles to LaTeX, carrying out such a conversion may be too large a task.

3 The quick and dirty solution

It became clear to me that a number of people going to my Web site are looking for older articles. So I decided that I will put all my articles on the Web.

This sounded really easy to do:

1. I scan the article.
2. I use Adobe Acrobat to turn the TIFF files created by the scanner to a PDF file.

It was indeed easy to do, but the result was terrible. Firstly, I came to realize that the edges of an old reprint were not necessarily cut parallel with the printed lines. So the scanned images were crooked. I also did not know how to optimally set my scanner. And when I set it to 300 dpi, black and white bitmap, the printed version came out 50% magnified, and as a result, really ugly (similar to a 200 dpi scan). Why is that? Go to the print dialog box of a PostScript printer, and choose the Acrobat pane; by default, it has a checkmarked box, `Fit to Page`. Uncheckmark the box, and the PDF file will print properly.

So you must warn the users, to leave the `Fit to Page` box unchecked. A better solution is in the next section.

4 The proper solution

We now use the following procedure to obtain the PDF files for legacy documents.

1. Scan each page of the document at 300 dpi, black and white bitmap.

Justification. The first decision to be made is at what dpi to scan. Since most printers these days are 1200 dpi printers, there is a temptation to scan at 1200 dpi.

Table 1 shows the size of a typical (large) printed page.

A 12 page paper at 300 dpi is about 1MB; it is more than 3MB at 600 dpi. At 1200 dpi, it is almost 6MB, obviously too large for most people to download.

Another way of increasing the file size is by using grayscale, instead of black and white bitmap, for the scanning. This increases the file size dramatically, and often decreases the quality of the PDF document. Avoid using grayscale unless there are grayscale illustrations in the document.

2. Open each page in Photoshop (or another similar application) and perform the following steps (stated specific to Photoshop):

Step 1. Change the page to grayscale
`Image>Mode>Grayscale`
keeping the `Image Ratio` at the default value 1.

Step 2. Enlarge the page and set the grid visible.
`View>Show>Grid`

Step 3. Use the eraser to get rid of dirt on the page.

Step 4. If necessary, rotate the picture in increments of 0.1 degrees to make the lines straight.
`Image>Rotate Canvas>Arbitrary...`

Step 5. When the lines are straight, change the page to black and white bitmap
`Image>Mode>Bitmap...`
keeping the `Output` at 300 pixels/inch.

Step 6. Change the canvas size to 8.5 inches by 11 inches; place the original image in the center (the default).
`Image>Canvas Size...`

Step 7. Save the image with `Save As...`, keep the TIFF format, and checkmark `LZW Compression`.

Justification. Step 1 is necessary, otherwise Step 4 cannot be done. Step 6 is useful, because then it no longer matters whether the `Fit to Page` box is checked.

Comment 1. Photoshop allows the creation of "buttons" to facilitate this process. The buttons are colored and carry a descriptive name to indicate the action that is carried out by one click on the button. I have five buttons for this procedure:

1. `To grayscale` It is colored gray. It does Steps 1 and 2.

Table 1: Size of a one-page scanned file

Scanning at	150 dpi	200 dpi	300 dpi	600 dpi
Size of TIFF file	172K	300K	668K	2.5MB
Size of PDF file	36K	52K	88K	212K
Quality of printed document	poor	poor	O.K.	excellent

2. `Turn clockwise` It turns clockwise by 0.1 degrees.
3. `Turn counterclockwise` It turns counterclockwise by 0.1 degrees.
4. `To bitmap` It is colored white. It does Step 5.
5. `Large canvas` It does Step 6.

Such a set of buttons can be saved. Then the set can be loaded as necessary.

Comment 2. J. Palme [4] raises the question how can you make a PDF document "international", that is, printable in North America in standard letter format, and outside of North America in A4 format. The summary of his advice is:

> If you are using U.S. Letter paper format, ensure that both the left and right margins are at least 21 mm (0.8 in).

Note that Step 6 ensures this in most circumstances, certainly with the 150 or so of my legacy articles.

Comment 3. You may get better PDF files if you make Step 4 slightly more sophisticated. Scan the document at 600 dpi. Then straighten the pages as follows:

- Select the Measure Tool (it hides behind the Eyedropper Tool).
- Drag it to draw a straight line across the page.
- Choose the menu item:
  `Image>Rotate Canvas>Arbitrary...`
  and a dialogue box comes up, showing the rotation necessary to straighten the page. Carry out the action by clicking `OK`.

Theoretically, this should give a better quality PDF document. In actual practice, I cannot see the improvement. Step 4, as recommended, has the advantage of simplicity. My papers were converted by a service bureau (Sri RAM Technicraft, e-mail: `sukanya8@mb.sympatico.ca`). Since they use untrained persons for such work, they appreciated the simplicity of the process.

3. Now take a look at how the pages are numbered. The scanner assigned them names such as

```
File1.xyz
File2.xyz
```

and so on. This will create problems if you have more than nine pages. So rename them

```
File01.xyz
File02.xyz
```

4. In Acrobat, choose `File>Import>Image`, choose all the pages in the document, and click on `Done`. The PDF file will be ready in a few seconds.

5. Acrobat will balk if you wish to make a PDF document of more than 50 pages. In this case, make more than one PDF document, and merge them as follows: open the first document in Acrobat. Choose `Document>Insert Pages...`; in the open dialog box, select the second PDF file. In the dialog box that comes up, for Location select `After`, and for page select `Last`. Clicking on `OK` will merge the two documents. Proceed thus until all the documents are merged.

Figure 1 shows a few lines of a scanned page on the Web. It is reminiscent of math papers printed on old 300 dpi laser printers.

5 Covers

Unfortunately, old reprints had covers containing vital information, maybe even the author and the title. As a rule, the journal information was on the cover only. So it is necessary in many instances to include the front cover with the article.

The covers pose special problems: they are often colored (say, medium blue) and somewhat the worse for the wear. If you scan them as in Section 4, the scanned image may be full of black spots — the image seems to be damaged beyond repair.

Here is what you have to do.

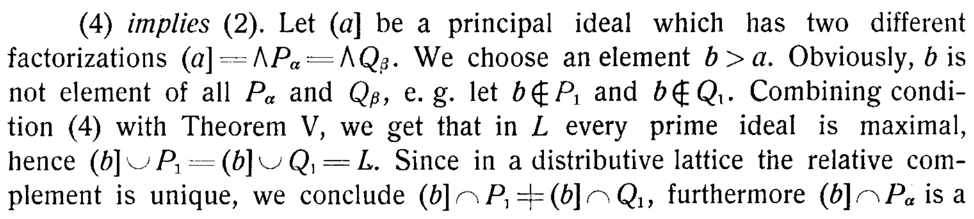
Step 1. Scan the cover at 600 dpi, grayscale.

Comment. Even the 600 dpi bitmap may be full of dirt or even completely black!

Step 2. In Photoshop, choose
`Select>Color Range...`.
In the dialogue box, pull down the `Select` menu, and choose `Shadows`. This selection consists of all the printed letters and logos (and a few dirt spots).

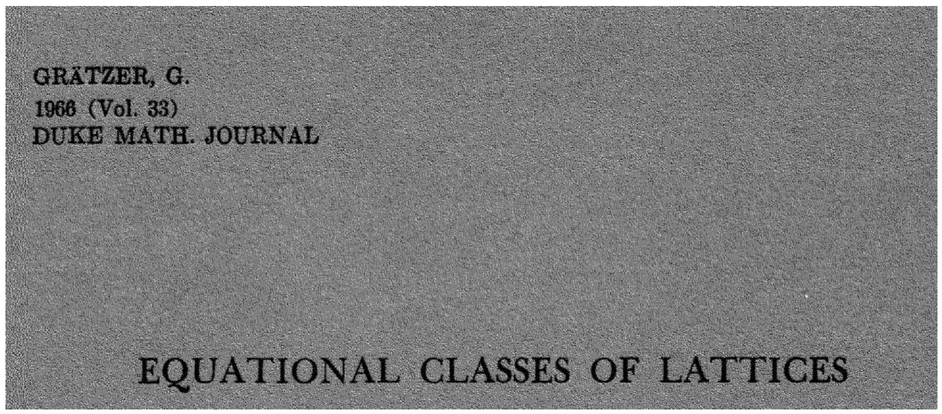
Step 3. Fill the selected area with black. This will change the grayscale print to black.

Step 4. Invert the selection with
`Select>Inverse`,
and fill the selection with white. This will create



(4) *implies* (2). Let $(a]$ be a principal ideal which has two different factorizations $(a] = \wedge P_\alpha = \wedge Q_\beta$. We choose an element $b > a$. Obviously, b is not element of all P_α and Q_β, e. g. let $b \notin P_1$ and $b \notin Q_1$. Combining condition (4) with Theorem V, we get that in L every prime ideal is maximal, hence $(b] \cup P_1 = (b] \cup Q_1 = L$. Since in a distributive lattice the relative complement is unique, we conclude $(b] \cap P_1 \neq (b] \cap Q_1$, furthermore $(b] \cap P_\alpha$ is a

Figure 1: 300 dpi scan on the Web.



GRÄTZER, G.
1966 (Vol. 33)
DUKE MATH. JOURNAL

EQUATIONAL CLASSES OF LATTICES

Figure 2: Cover scanned at 600 dpi.

a white background. Clean up the dirt, and if necessary, repeat Steps 3 and 4. Now you should have a beautiful black and white cover.

Figure 2 shows a portion of a cover scanned grayscale at 600 dpi. Figure 3 is the cleaned up version.

R. Padmanabhan suggested to cut from the cleaned up cover the information missing from the first page and paste it on the top of the first page. This saves storage and download time. However, the reader may be under the impression that the scanned pages are faithful representations of the original, which would not remain quite true with this scheme.

6 Future

Obviously, some years in the future, my Web site as set up today will appear obsolete. When most users will have very fast Web connections, when data storage will be measured in hundreds of gigabytes, all articles will be scanned at 2400 dpi.

More importantly, Adobe Acrobat introduced the capture command, which allows us to store a legacy document in two ways: as an image and as text (linked to the image). This has the advantage that we can view a legacy document as it looked originally, and the text layer allows complete searching capabilities. Unfortunately, I was completely unsuccessful in my attempts to use capture for my documents. Even PDF files converted from LaTeX showed a very high failure rate.

You can get a glimpse of the future at `http://www.jstor.org/`, the Web site of JSTOR, The Journal Storage, The Scholarly Journal Archive. The image files are created at 600 dpi and the text recognition is accomplished with proprietary software (and carefully proofread). The mathematics journals include the Proceedings of the American Mathematical Society and the Transactions of the American Mathematical Society.

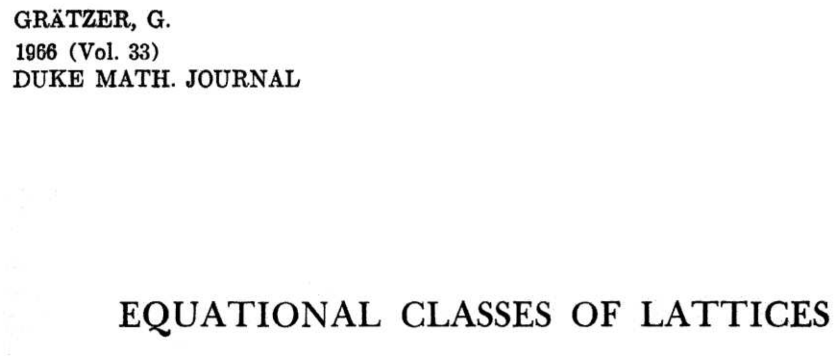
The future will have all this and links: *internal links* in an article from "by Lemma 5" to Lemma 5, from "see [12]" to the citation [12]; and *externally*, from citation [12] to the actual article.

References

[1] Michel Goossens and Sebastian Rahtz (with Eitan Gurari, Ross Moore, and Robert Sutor), *The LaTeX Web Companion: Integrating TeX, HTML and XML*. Addison-Wesley, Reading, MA, 1999.

[2] George Grätzer, *Math into LaTeX*, third edition, Birkhäuser Verlag, Boston, Springer-Verlag, New York, 2000. xl+584 pp. ISBN 0-8176-4131-9, ISBN 3-7643-4131-9.

[3] Thomas Merz, *Web Publishing with Acrobat PDF*. Springer-Verlag New York, 1998.



GRÄTZER, G.
1966 (Vol. 33)
DUKE MATH. JOURNAL

EQUATIONAL CLASSES OF LATTICES

Figure 3: Cover cleaned up.

[4] Jacob Palme, *Making Postscript and PDF International*, Network Working Group, Request for Comments: 2346, Stockholm University, May 1998.
`http://dsv.su.se/jpalme/ietf/jp-ietf-home.html\#anchor1470437`, `http://www.ietf.cnri.reston.va.us/rfc/rfc2346.txt`

◇ George Grätzer
Department of Mathematics
University of Manitoba
Winnipeg MN, R3T 2N2
Canada
`gratzer@cc.umanitoba.ca`
`http://server.math.umanitoba.ca/homepages/gratzer/`

Anatomy of a macro

Denis Roegel

Abstract

In this article, we explain in detail a TEX macro for computing prime numbers. This gives us an opportunity to illustrate technical aspects often ignored by TEX beginners.

This article is dedicated to Chrystel Barraband for whom the first version was written in 1993.

This article is a translation, with corrections, of the article "Anatomie d'une macro" published in the *Cahiers GUTenberg*, number 31, December 1998, pages 19–27. Reprinted with permission.

Introduction

A TEX macro can be seen as the definition of a command by other commands. Both the definition of a command and the way arguments are passed obey rules which are both precise and simple, but which are often overlooked, though indispensable to a good understanding of TEX.

Moreover, the call of a TEX macro is a very different process from what happens in classical languages. It is similar to a macro call in the C preprocessor and it is hard to imagine programming with such a language! A macro call merely entails a replacement or a substitution, but it can also call other macros, including itself, which allows recursion.

Computing prime numbers

We will focus on the computation of prime numbers. $n > 1$ is prime if n is divisible only by itself and 1. If

n is odd, it is sufficient to divide n by $3, 5, 7, \ldots, p \leq \lfloor\sqrt{n}\rfloor$. For, if n can be divided by $p > \lfloor\sqrt{n}\rfloor$, then n can also be divided by $q < \lfloor\sqrt{n}\rfloor$. The divisors p will be tried until $p^2 > n$.

Macros

The following example, from *The TEXbook* (Knuth, 1984), is of an advanced level but will allow us to go straight to the heart of the matter. The macro `\primes` makes it possible to determine the first n prime numbers, starting with 2. For instance, `\primes{30}` returns the first 30 prime numbers. Here are all the definitions.[1] We will then analyze them in detail:

```
\newif\ifprime \newif\ifunknown
\newcount\n \newcount\p
\newcount\d \newcount\a
\def\primes#1{2,~3% assume that #1>2
  \n=#1 \advance\n by-2 % n more to go
  \p=5 % odd primes starting with p
  \loop\ifnum\n>0 \printifprime
       \advance\p by2 \repeat}
\def\printp{, % invoked if p is prime
  \ifnum\n=1 and~\fi
  \number\p \advance\n by -1 }
\def\printifprime{\testprimality
                  \ifprime\printp\fi}
\def\testprimality{{\d=3 \global\primetrue
  \loop\trialdivision
     \ifunknown\advance\d by2 \repeat}}
\def\trialdivision{\a=\p \divide\a by\d
  \ifnum\a>\d \unknowntrue
  \else\unknownfalse\fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse
              \unknownfalse\fi}
```

Declarations

First, we declare two booleans, or more precisely two tests.

```
\newif\ifprime
```

`\ifprime` is equivalent to `\iftrue` if "`prime`" is true. This boolean will make it possible to see if a number must be printed; thus, in `\printifprime`, the expression `\ifprime\printp\fi` means that if `\ifprime` is evaluated to `\iftrue`, then `\printp` (that is, the macro that will print the number of interest to us, namely `\p`) will be executed, otherwise nothing will happen.

```
\newif\ifunknown
```

[1] The code was slightly reformatted to fit in the columns.

"`unknown`" will be true if we are not yet sure whether `\p` is composed or not. Neither is known. Initially, "`unknown`" is thus true and the `\ifunknown` test succeeds. If "`unknown`" is false, we have knowledge about `\p`'s primality, that is, we know if `\p` is prime or not.

Next the code defines a few integer variables useful in what follows:

- `\newcount\n`

  `\n` is the number of prime numbers that remain to be printed.
- `\newcount\p`

  `\p` is the current number for which primality is tested.
- `\newcount\d`

  `\d` is a variable containing the sequence of trials of divisors of `\p`.
- `\newcount\a`

  `\a` is an auxiliary variable.

Main macro

The main macro is `\primes`. It takes an argument. When the macro is defined, this argument has the name `#1`. If there were a second argument, it would be `#2`, etc. (It is not possible to have — directly — more than nine arguments; indirectly however, one can have as many arguments as one wants, including a variable number, which could for instance be a function of one of the arguments.)

```
\def\primes#1{2,~3%
  \n=#1 \advance\n by-2 %
  \p=5 %
  \loop\ifnum\n>0 \printifprime
      \advance\p by2 \repeat}
```

When the `\primes` macro is called, for instance with 30, `\primes{30}` is replaced by the body of `\primes` (that is, the group between braces which follows the list of `\primes`' formal arguments), in which `#1` is replaced by the two characters 3 and 0. `\primes{30}` hence becomes (we have removed spaces at the beginning of the lines, because they are ignored by TEX):

```
2,~3%
\n=30 \advance\n by-2 %
\p=5 %
\loop\ifnum\n>0 \printifprime
    \advance\p by2 \repeat
```

What happens now? We print "`2,~3`", that is, 2 followed by a comma, followed by an unbreakable space (i.e., the line will *in no case* be split after the comma); then 30 is assigned to `\n`. Immediately, 2 is subtracted from `\n`, and `\n` then contains the number of primes that remain to be printed. To keep it simple, we have assumed that at least the three first primes must be displayed. Therefore, we are sure that `\n` is at least equal to 1. This is also why it was possible to put a comma between 2 and 3, because we know that 3 is not the last number to be printed. We want the last number printed to be preceded by "`and`". Hence, when we ask `\primes{3}`, we want to obtain "`2, 3, and 5`". It should also be noticed that the "`%`" after "3" is essential to prevent insertion of a spurious space. "3" will be followed by a comma when `\printp` is called. The "`%`" after the second and third lines are not really needed since TeX gobbles all spaces after explicit numbers; these "`%`" signs appear only as remnants of comments.

We said that `\p` is the current number whose primality must be tested. We must therefore initialize `\p` to 5, since it is the first odd number after 3 (which we don't bother to check if it is prime or not).

The body of `\primes{30}` ends with a loop:

```
\loop\ifnum\n>0 \printifprime
    \advance\p by2 \repeat
```

It is a `\loop/\repeat` loop. In general, these loops have the form

```
\loop A text \if...   B text \repeat
```

This loop executes as follows: it starts with `\loop`, the A text is executed, then the `\if...` test. If this test succeeds, the B text is executed, then `\repeat` makes us return to `\loop`. If the test fails, the loop is over.

Hence, in the case of `\primes{30}`, it amounts to execute

```
\printifprime\advance\p by2
```

as long as `\n` is strictly positive, that is, as long as prime numbers remain to be printed. In order for this to produce the expected result, it is of course necessary to decrement the value of `\n`. This is done every time a number is printed with the call to `\printifprime`.

As a consequence, if at least one number remains to be printed, `\printifprime` will be called and will print `\p` if `\p` is prime. Whatever the result, we pass then to the next odd number with `\advance\p by2`.

Printing

The prime numbers are printed with `\printp`:

```
\def\printp{, %
    \ifnum\n=1 and~\fi
    \number\p \advance\n by -1 }
```

This macro is called only when `\p` is prime (see its call in `\printifprime`). In any case, this macro has no arguments and gets expanded into

```
, %
\ifnum\n=1 and~\fi
\number\p \advance\n by -1
```

that is a comma and a space, followed by "`and` " if `\n` equals 1 (in the case where the number to be printed is the last one), followed by `\p` (the `\number` function is analogous to `\the` and converts a variable into a sequence of printable characters); finally, `\n` is decremented by 1, as announced, and this allows a normal unfolding of the `\loop...\repeat` loop in the `\primes` macro.

The macro `\printifprime` is called by `\primes`. It calls the function computing the primality of `\p` and this determines if `\p` must be printed or not.

```
\def\printifprime{\testprimality
                 \ifprime\printp\fi}
```

As one can guess, the `\testprimality` macro sets the "`prime`" boolean to "`true`" or "`false`," or if one prefers, it makes the `\ifprime` test succeed or fail.

Primality test

The macro testing `\p`'s primality uses the classical algorithm where divisions are tried by numbers smaller than `\p`'s square root.

```
\def\testprimality{{\d=3 \global\primetrue
  \loop\trialdivision
    \ifunknown\advance\d by2 \repeat}}
```

This macro is more complex because it involves an additional "group," shown here by the braces. Therefore, when `\testprimality` is expanded, we are left with

```
{\d=3 \global\primetrue
 \loop\trialdivision
   \ifunknown\advance\d by2 \repeat}
```

meaning that what happens between the braces will be — when not otherwise specified — local to that group. This was not the case in the expansions seen previously.

Let us first ignore the group. What are we doing? 3 is first assigned to `\d` where `\d` is the divisor being tested. We will test 3, 5, 7, etc., in succession, and this will go on as long as it is not known for certain whether `\p` is prime or not. As soon as we know if `\p` is prime or composed, the "`unknown`" boolean will become false and the `\ifunknown` test will fail.

Now, let us look at this again: we start with `\d=3`; the default is to consider `\p` prime, hence the "true" value is given to the "prime" boolean. This is normally done with

```
\primetrue
```

but in our case, it would not be sufficient. Indeed, at the end of

```
{\d=3 \primetrue
 \loop\trialdivision
   \ifunknown\advance\d by2 \repeat}
```

all variables take again their former value, because the assignments are *local* to the group. But the "prime" boolean is used when the \ifprime... test is being done in \printifprime, which is called after \testprimality. The group must therefore be *transcended* and the assignment is coerced to be global. This is obtained with

```
\global\primetrue
```

The remainder is then obvious: an attempt is made to divide \p by \d, and this is the purpose of \trialdivision. If nothing more has been discovered, that is, if "unknown" is still "true", the value of the trial divisor is set to the next value with \advance\d by2. Sooner or later this process stops, as shown by the \trialdivision definition.

The additional group in \testprimality can now be explained. If the group is not introduced, the expansion of \primes{30} leads to

```
...
\loop\ifnum\n>0 \printifprime
  \advance\p by2 \repeat
```

Plain TeX defines \loop as follows:

```
\def\loop#1\repeat{\def\body{#1}\iterate}
\def\iterate{\body\let\next\iterate
  \else\let\next\relax\fi \next}
```

Therefore, the initial text is expanded into

```
\def\body{\ifnum\n>0 \printifprime
  \advance\p by2 }\iterate
```

Hence, the \loop...\repeat construct becomes

```
\ifnum\n>0 \printifprime\advance\p by2
           \let\next\iterate
\else \let\next\relax\fi \next
```

If \n > 0, this leads to

```
\printifprime ...
\let\next\iterate \next
```

and hence to

```
\testprimality ...
\let\next\iterate \next
```

and to

```
... \loop\trialdivision
    \ifunknown\advance\d by2 \repeat ...
\let\next\iterate \next
```

Now, \iterate will call \body, but the \body definition called will be the one defined by the second (inner) \loop, and chaos will follow! This explain why a group has been introduced. The group keeps the inner \body definition away from the outer \loop construct, hence each \iterate call produces the appropriate result.

Division trials

The last macro is where the actual division of \p by \d is made. An auxiliary variable \a is used.

```
\def\trialdivision{\a=\p \divide\a by\d
  \ifnum\a>\d \unknowntrue
  \else\unknownfalse\fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse
              \unknownfalse\fi}
```

\p is copied into \a, then \a is divided by \d. This puts into \a the *integer part* of $\frac{\texttt{\textbackslash p}}{\texttt{\textbackslash d}}$. Two cases must then be considered:

1. if \a $>$ \d, that is, if \d is smaller than the square root of \p, we are still in unknown territory. \d may be a divisor of \p, or there might be another divisor of \p larger than \d and smaller than the square root of \p root. The "unknown" boolean is therefore set to "true" with \unknowntrue.
2. if \a $\leq$ \d, we assume that we know, or at least, that we will know momentarily. We write therefore \unknownfalse.

In order to be sure, we must check if there is a remainder to \p's division by \d, or rather to \a's division by \d: \a is therefore multiplied by \d:

```
\multiply\a by\d
\ifnum\a=\p \global\primefalse
            \unknownfalse\fi
```

If \p is found again, it means that \d is one of \p's divisors. In that case, \p is of course not prime and the "prime" boolean is set to false with \primefalse. Since \trialdivision is actually located in the group surrounding the body of the \testprimality macro, and since the "prime" is needed outside \testprimality, the group must once again be transcended and the "prime" assignment must be forced to be global. Hence:

```
\global\primefalse
```

Finally, in the case where \d divides \p, we set \unknownfalse, which has as sole effect of causing the loop to end:

```
\loop\trialdivision
  \ifunknown\advance\d by2 \repeat
```

that is, no other divisor is tested. One can observe that there is no `\global` in front of `\unknownfalse`, because `\ifunknown` is used within and not outside the group.

If `\p` is not found again after the multiplication, it means that `\d` is not a divisor of `\p`. At that time, we had

- either `\a` $\leq$ `\d`, and therefore `\a` $<$ `\d` (otherwise `\p` would have been found after the multiplication), and hence `\unknownfalse`, therefore the loop

  ```
  \loop\trialdivision
    \ifunknown\advance\d by2 \repeat
  ```

  stops and since this happens in the context

  ```
  \d=3 \global\primetrue
  \loop\trialdivision
    \ifunknown\advance\d by2 \repeat
  ```

  where "`prime`" had been set to true, we conclude naturally that, no divisor having been found up to `\p`'s square root, `\p` is prime.

  Therefore, at the end of `\testprimality`'s call, `\ifprime` succeeds and `\p` is printed.

- or `\a` $>$ `\d`: in that case, we know nothing more, `\unknowntrue`, and the next divisor must be tried.

Conclusion

This ends the explanation of these macros, apart from a few subtleties which were not mentioned.

It takes TEX a lot of time to do complex operations such as the ones described. In order to execute`\primes{30}`, TEX spends more time than it needs on average to typeset a whole page with plain TEX. `\trialdivision` is expanded 132 times. With `\primes{1000}` there are 41331 expansions and with `\primes{10000}` there are 1441624 expansions.

It should be stressed that the previous macros are given in *The TEXbook* (Knuth, 1984, pp. 218–219), with the following lines as the only explanation:

> The computation is fairly straightforward, except that it involves a loop inside a loop; therefore `\testprimality` introduces an extra set of braces, to keep the inner loop control from interfering with the outer loop. The braces make it necessary to say '`\global`' when `\ifprime` is being set true or false. TEX spent more time constructing that sentence than it usually spends on an entire page; the `\trialdivision` macro was expanded 132 times.

TEX's programming language is quite peculiar and we gave only a glimpse of it. The interested reader should dive into TEX's "bible", namely Donald Knuth's *TEXbook* (Knuth, 1984).

Acknowledgments

I would like to thank an anonymous referee for noticing an important error in the French version of the article.

References

Knuth, Donald E. *The TEXbook*. Addison-Wesley, Reading, MA, USA, 1984.

◇ Denis Roegel
LORIA
Campus scientifique
BP 239
54506 Vandœuvre-lès-Nancy cedex
FRANCE
`roegel@loria.fr`
`http://www.loria.fr/~roegel/`

Macros

Macros with optional arguments

Victor Eijkhout

Users of LaTeX are familiar with macros that have optional arguments, such as `\newcommand`.

```
\newcommand\testa{ ... }
\newcommand\testb[2]{ ... }
```

Here, the second argument is optional; its inclusion alters the workings of `\newcommand`. Wouldn't it be nice to be able to write such macros yourself?

Let's set ourselves the project, for now, of writing a macro with one optional and one required argument. If the optional, first, argument is omitted, the value of the second, required one should be used. That is, our macro, which we shall call `\aa`, should have the following behaviour: the input

```
\aa [1]2
\aa 2
```

should give the output

```
Opt: [1] Req: [2]
Opt: [2] Req: [2]
```

The crux to these optional arguments is a test for the occurrence of the opening square bracket; that is, somehow we need to peek at what follows the macro. For this, TeX has the `\futurelet` command. The example

```
\futurelet\x\y z
```

has the effect of

```
\let\x z\y
```

That is, the first argument `\x` is '`\let`' to whatever follows the second argument `\y`, and then the second argument is executed. Why does this help us? Well, we can now `\futurelet` the token after `\aa` to, say, `\next`, and then call a macro that investigates whether `\next` is a square bracket, and acts accordingly. The 'acting accordingly' part means calling one or the other of two macros, the one handling the case where there was a square bracket, the other the case where there wasn't.

```
\def\aa{\futurelet\next\aaX}
\def\aaX{%
  \ifx[\next \expandafter\aaXX
  \else \expandafter\aaXXX \fi}
```

We now define two separate macros, one with and one without optional argument.

```
\def\aaXX[#1]#2{
  Opt: [#1] Req: [#2]\par}
\def\aaXXX#1{
  Opt: [#1] Req: [#1]\par}
```

Since we decided that calling the macro without the optional argument would have the effect of doubling the required argument — something that happens frequently in the internals of LaTeX — we can write

```
\def\aaXXX#1{\aaXX[#1]{#1}}
```

and save ourselves some code duplication.

If, instead of duplicating the first required argument, we wanted to use some default value for the omitted optional argument, we would write

```
\def\aaXXX#1{\aaXX[<default>]{#1}}
```

Just one remark. Note that until the end, where we call `\aaXX`, we never look at the other arguments, and we do not care how many of them there are. In calls such as `\aa [1]234...` we repeatedly replace the `\aa` control sequences by other control sequences. Only the final macros `\aaXXX` — in the case of an optional argument — and `\aaXX` — in the case of none — touch the actual arguments.

Now let's consider the case where the optional argument is not the first but the second. Say, we want a macro `\bb` that, called as

```
\bb 1[2]3
\bb 13
```

gives

```
One: [1] Opt: [2] Req: [3]
One: [1] Opt: [3] Req: [3]
```

The trick here is to scoop up the first argument and store it away:

```
\def\bb#1{\def\savedargone{{#1}}%
  \futurelet\next\bbX}
```

After that, we proceed for a while as before

```
\def\bbX{%
  \ifx[\next \expandafter\bbXX
  \else \expandafter\bbXXX \fi}
\def\bbXXX#1{\bbXX[#1]{#1}}
```

and then we insert the saved argument in between the final macro and the the remaining arguments:

```
\def\bbXX{\expandafter\bbY\savedargone}
\def\bbY#1[#2]#3{
  One: [#1] Opt: [#2] Req: [#3]\par}
```

The `\expandafter` in `\bbXX` turns the sequence

```
\bbXX <arg 2><arg 3>
```

which first becomes

```
\expandafter\bbY\savedargone
               <arg 2><arg 3>
```

into

```
\bb <arg1><arg 2><arg 3>
```

Well, there you have it. All the ingredients for writing macros with optional arguments.

So why isn't this article over? Well, after writing macros like this becomes a second nature to you, you might start wondering if there isn't a way to automate this rather repetitive process. And of course there is. But it is a bit of work. In fact, this may well be the most mind-bending macro I have ever written.

A small device to save us some typing:

```
\let\expa\expandafter
\let\noex\noexpand
```

We now set ourselves the goal of writing a macro `\defoptargcomm`—'define an optional argument command'—that allows us to write

```
\defoptargcomm\def\aa[#1]#2{%
  Opt: [#1] Req: [#2]\par}
```

so that again, as above,

```
\aa [1]2
\aa 2
```

gives the right result.

I will give the macro in increments. First of all, the name of the macro to be defined is changed from a control sequence into a string of characters, so that we can base other macro names on it:

```
\def\defoptargcomm#1#2{%
  \edef\bnon{\stringcsnoescape#2}%
```

The auxiliary macro `\stringcsnoescape` is given below, as will be all further auxiliaries.

Next we define the macro that peeks at a possible square bracket. To get the effect of

```
\def\aa{\futurelet\next\aaX}
```

we write (where `\nxarg` and `\nxname` are auxiliaries; see below)

```
  \edef\anon{\nxarg#1{\bnon}{%
    \futurelet\noex\next
    \nxname{\bnon X}}}\anon
```

This is a good trick: since we will have to form some new control sequences, we build the define statement inside the `\edef` of an otherwise unimportant macro. Calling this macro will then execute the definition. (The auxiliary `\nxname` serves to build a control sequence and further prevent it from being expanded. See the end of this article.)

We use this trick again, this time to define the macro that will decide, based on the presence or not of a square bracket, which further macro to call. For

```
\def\aaX{%
  \ifx[\next \expa\aaXX
  \else \expa\aaXXX}\fi}
```

we write

```
  \edef\anon{\nxarg#1{\bnon X}{%
    \noex\ifx[\noex\next
      \noex\expa
        \nxname{\bnon XX}%
    \noex\else
      \noex\expa
        \nxname{\bnon XXX}%
    \noex\fi}}\anon
```

Here is the macro that duplicates the first argument if there is no optional argument. For the equivalent of

```
\def\aaXXX#1{\aaXX[#1]{#1}}
```

we write

```
  \edef\anon{%
   \nxarg#1{\bnon XXX}####1{%
    \nxname{\bnon XX}[####1]{####1}}%
           }\anon
```

And now we would have to do the actual definition of the macro with optional argument, as we did in the first part of the article. However, we can skip this, as the definition was already in the input stream, so we conclude the definition of `\defoptargcomm` with the `\def` control sequence (argument `#1`) and the name with two Xs attached. For

```
\def\aaXX   ...
  % .. after this come the parameters
  % .. and definition
```

we conclude with

```
  \arg#1{\bnon XX}}
```

Phew.

What? You want more?

Well, the assumption that an omitted optional argument should take on the value of the first present argument is a bit limiting. You may want it to take on some default value. For instance, the syntax

```
\defoptargcomm[4]\def\bb[#1]#2{%
  Opt: [#1] Req: [#2]\par}
```

would mean that the value taken in absence of an optional argument is '4'. The input

```
\bb [1]2

\bb 2
```

then gives

```
Opt: [1] Req: [2]
Opt: [2] Req: [2]
```

That is not very hard to do: we need yet another application of `\futurelet`.

```
\def\defoptargcomm{%
  \futurelet\next\defoptargcommX}
```

```
\def\defoptargcommX{%
  \ifx[\next
    \expandafter\defoptargcommXX
  \else \def\optarg{[########1]}%
    \expandafter\defoptargcommXXX
  \fi}
\def\defoptargcommXX[#1]{%
 \def\optarg{[#1]}\defoptargcommXXX}
```

We are thus saving the value of the optional argument in a control sequence `\optarg`. Note the sequence of eight hash characters, which I will not further explain[1].

Now the macro `\defoptargcommXXX` has to use the value of `\optarg`. For this we change only a small part. For the equivalent of

```
\def\bbXXX#1{%
  \bbXX[optarg]{#1}}
```

```
  \edef\anon{%
   \nxarg#1{\bnon XXX}####1{%
     \nxname{\bnon XX}\optarg{####1}}%
            }\anon
```

Tada!

Now, if you've followed this exposition carefully, you'll have noticed that this ultra-powerful macro can still not do something that we could do by hand: let any argument be optional, not just the first. Let us say that we want to write

```
\PACdefoptargncomm3%
 \def\cc#1#2[#3]#4{First: [#1,#2]
    Opt: [#3] Req: [#4]\par}
\cc 12[3]4
\cc 124
```

and get

```
First: [1,2] Opt: [3] Req: [4]
First: [1,2] Opt: [4] Req: [4]
```

Deep breath. Here comes the final version of our macro for defining macros with optional arguments.

I will explain this one bottom-up, instead of top-down. Our first problem is that we need to get the first couple of fixed arguments out of the way before we can look at the optional argument. Suppose we have a macro `\firstarg` that expands to the arguments before the optional one, in this case `#1#2`, and a macro `\savedarg` with the same, but in braces: `{#1}{#2}`, then to get the equivalent of

```
\def\cc#1#2{\def\ccsaved{{#1}{#2}}%
    \futurelet\next\ccX}
```

we write — and compare this with the above —

```
 \edef\anon{\nxarg#1{\bnon}\firstarg
     {\def\nxname{\bnon saved}%
           {\savedarg}%
       \futurelet\noex\next
       \nxname{\bnon X}}}\anon
```

Now that we have the first arguments set aside, we can look for the square bracket. If is is not there, we have to call a macro that duplicates the next argument; if is is there, we re-insert the saved arguments, and call the final macro. This would read

```
\expa\ccXX\ccsaved
```

but because it occurs in a conditional it becomes

```
\expa\expa\expa\ccXX\expa\ccsaved
             \else
```

and because it happens inside an `\edef` there are `\noexpand`s interspersed everywhere:

```
  \edef\anon{\nxarg#1{\bnon X}{%
    \noex\if[\noex\next
      \noex\expa\noex\expa\noex\expa
        \nxname{\bnon XX}%
          \noex\expa\nxname
                  {\bnon saved}%
    \noex\else
      \noex\expa\nxname{\bnon XXX}%
    \noex\fi}}\anon
```

By comparison, the macro to duplicate the argument after the omitted optional argument is child's play:

```
  \edef\anon{%
     \nxarg#1{\bnon XXX}####1{%
      \noex\expa\nxname{\bnon XX}%
       \nxname{\bnon saved}%
        [\optarg]{####1}}}\anon
```

Note the `\optarg`, which contains either the tokens `####1`, or a default value.

All this is inside a macro

```
\def\defoptargcommXXX#1#2{%
  \def\protect{}%
  \edef\bnon{\stringcsnoescape#2}%
  < ... the above ... >
  \arg#1{\bnon XX}}
```

This is the macro that handles the explicit default value:

```
\def\defoptargcommXX[#1]{%
  \def\optarg{#1}\defoptargcommXXX}
```

[1] Okay, just a little bit then. If TEX sees one hash character followed by a letter — which can only happen in a macro — it replaces it by the corresponding macro argument. Two hash characters in a row are replaced by a single, which is further left untouched. Unless, that is, it is scanned again. Since in the end the sequence here will be scanned three times we need to write eight hash characters in order to get #1 in the input stream.

We're getting close to the interesting bits. This the macro that tests for a default value:

```
\def\defoptargcommX{%
 \ifx[\next
   \expandafter\defoptargcommXX
 \else
  \edef\anon
   {\def\noex\optarg{\protect\hash1}%
   }\anon
  \expandafter\PACdefoptargcommXXX
 \fi}
```

Note the occurrence of a macro `\hash`, which we will define in a minute. Here is the old macro for an optional first argument:

```
\def\PACdefoptargcomm{%
  \def\PACfirstarg{}\def\PACsavedarg{}%
  \def\protect{\noex\protect\noex}%
  \def\hash{########}%
  \futurelet\next\defoptargcommX}
```

This is the nasty one: the macro that accepts the location of the optional argument and builds the `\firstarg`, `\savedarg` macros. We use two token lists, which gradually get build inside a loop.

```
\def\defoptargncomm#1{%
 \toksa={}\toksb={}\counta=#1\relax
 \def\protect{\noex\protect\noex}%
 \def\hash{########}%
 {\count1=1 \count2=#1
  \loop
   \edef\PACanon{
     \global\PACtoksa={\the\PACtoksa
       \protect\hash\number\count1}%
     \global\PACtoksb={\the\PACtoksb
      {\protect\hash\number\count1}}%
               }\PACanon
   \advance\count1 by 1\relax
  \ifnum\count1<\count2 \repeat
 }%
 \edef\anon{\def\noex\firstarg
             {\the\toksa}}\PACanon
 \edef\anon{\def\noex\savedarg
             {\the\PACtoksb}}\PACanon
 \futurelet\next\defoptargcommX}
```

And that's it. You can get this monster from CTAN as `PAC_utils.tex`; you also need `CS_auxs.tex`.

Finally, here are the auxiliary macros. I will leave out mentioning various conditions on the functioning of these macros; normally they will be satisfied. To convert a control sequence to a string of characters, purely by expansion:

```
\let\expa\expandafter
\let\noex\noexpand
\def\stringcsnoescape#1{%
  \expa\gobbleescape\string#1}
{\escapechar-1
\expa\expa\expa\gdef
  \expa\expa\expa\CSgobblearrow
    \expa\string
      \csname macro:->\endcsname{}
}
\def\gobbleescape#1{%
  \ifnum`\\=`#1 \else #1\fi}
```

Here are various macros to build a control sequence out of a string of characters, and subsequently to protect the control sequence from further expansion:

```
\def\name#1{\csname#1\endcsname}
\def\arg#1#2{%
  \expa#1\csname#2\endcsname}
\def\csarg#1#2{%
  \name{#1\expa}\csname#2\endcsname}
\def\nxarg#1#2{%
  \expa#1\expa\noex
    \csname#2\endcsname}
\def\nxname#1{%
  \expa\noex\csname#1\endcsname}
```

◇ Victor Eijkhout
Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301 U.S.A.
victor@eijkhout.net

Drawing Message Sequence Charts with LaTeX

Sjouke Mauw and Victor Bos

Abstract

The MSC macro package facilitates LaTeX users to easily include Message Sequence Charts in their texts. This article describes the motivation for developing the MSC macro package, the features of the MSC macro package, and the design of the MSC macro package.

1 Introduction

The Message Sequence Chart (MSC) language is a visual formalism to describe interaction between components of a system. The language is standardized by the ITU (International Telecommunication Union) in Recommendation Z.120 [4]. An introductory text on MSC can be found in [5]. MSCs have a wide application domain, ranging from requirements specification to testing and documentation.

An example of a Message Sequence Chart is given in Figure 1. The MSC shows an ftp login session to a CTAN archive. Three players, called *instances*, are involved in the session: *User*, *ftp client*, and *CTAN* at location *ftp.tex.ac.uk*. The instances are denoted by vertical lines. Interaction between instances is denoted by labeled arrows. For instance, the arrow *ftp.tex.ac.uk* is a message from *User* to *ftp client*. Sending and receiving of a message are special types of *events*; each message has a *send* event and a *receive* event. Later we will see other types of events. Events occur on instance lines. Events are ordered in time and for each instance, time is supposed to run from top to bottom. Furthermore, the send event of a message never occurs after the receive event of the message. For example, from Figure 1, we can derive that the *ftp.tex.ac.uk* message occurs before the *connect* message, because the receive event of the first message occurs before the send event of the second message.

In order to include MSCs in LaTeX documents, we have developed the MSC macro package. The current version of the MSC macro package supports almost the full MSC language as defined in the standard. In this article we will describe the motivation of the MSC macro package, the features of the MSC macro package, the design of the MSC macro package, and the limitations of the MSC macro package. This paper does not describe all features of the MSC macro package. For a thorough treatment of the MSC macro package we refer to the user manual [2].

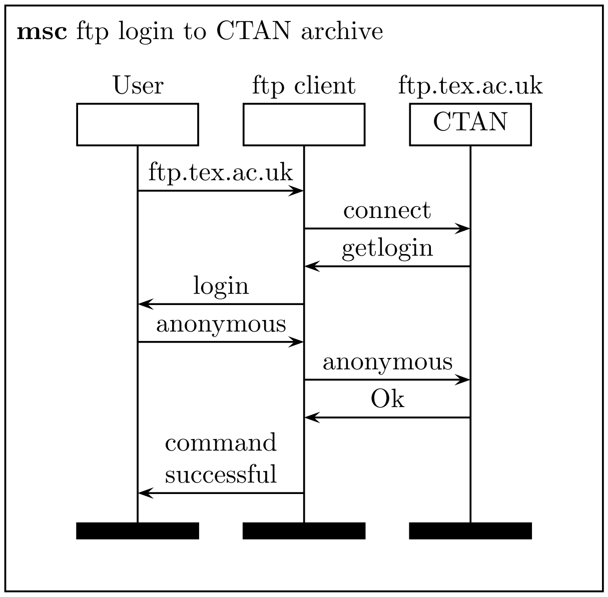


Figure 1: An ftp login to the *CTAN* at *ftp.tex.ac.uk*.

2 Motivation

Several commercial and non-commercial tools are available, which support drawing or generating Message Sequence Charts. However, these tools are in general not freely available and often not flexible enough to satisfy all users' wishes with respect to the layout and graphical appearance of an MSC. Furthermore, they often do not allow the user to include LaTeX code in the MSCs. Another drawback of these tools is that quite often they restructure MSCs automatically. Though for simple MSCs this might be what the user wants, for more complex MSCs the result of automatic restructuring is usually not desired.

Therefore, people often use general drawing tools, such as *xfig* (see `http://www.xfig.org/`) to draw MSCs. However flexible this approach is, it has some drawbacks. First of all, general drawing tools have (and should have) a low level of abstraction; their interface is defined in terms of *coordinates*, *points*, *lines*, *polygons*, etc. To draw MSCs, the user would probably be more comfortable if the interface was defined in terms of *instances*, *messages*, *actions*, etc. For example, if you are drawing a message in an MSC using a general drawing tool, you would probably have to draw a *line* with an arrow head from a *position* (x_0, y_0) to a *position* (x_1, y_1), instead of drawing a *message* from an *instance* i_0 to an *instance* i_1 of the MSC.

Another drawback of using general drawing tools is that they usually do not provide libraries of MSC symbols. Therefore, if you have to draw many MSCs, it will take much effort to get a set of consistent looking MSCs. Furthermore, if you want to change a parameter of the MSCs, e.g., the width of the instance head symbols, you would probably have to edit all MSCs manually.

For these reasons, we developed the MSC macro package for LaTeX. The macros in the package enable a textual representation of an MSC in a LaTeX source document. By compiling the LaTeX document into PostScript, a graphical representation of the MSC is generated.

The design requirements for the MSC macro package were:

1. The package should follow the ITU standard with respect to shape and placement of the symbols of an MSC.
2. The interface of the package should be at the right level of abstraction.
3. There should only be a limited amount of automatic restructuring and layout of the MSCs.
4. The appearance of (sets of) MSCs should be configurable by an appropriate set of parameters.
5. The MSC macro package should run on standard LaTeX distributions.

3 User interface

In this section we will briefly describe the user interface of the MSC macro package. We will do this by giving examples and showing the LaTeX code that produced the examples.

MSC environment MSCs are drawn in the `msc` environment. The syntax of this environment is `\begin{msc}[titlepos]{title} ... \end{msc}`. The title of the MSC is defined by the `title` parameter. The optional parameter `titlepos` determines the position of the title. By default it is `l` (left aligned). Other possible values are `c` (centered) and `r` (right aligned).

Instances Instances are declared with the

`\declinst[*]{nn}{an}{in}`

command. The starred version produces a *fat instance* which will not be discussed in this paper. The `nn` parameter defines a *nickname* of the instance. Nicknames identify instances and are used to draw messages and events. The `an` parameter defines the *above name* of the instance. This is the text to be placed above the instance head symbol (the rectangle at the top of an instance). The `in` parameter defines the *inside name* of the instance. This is the text to be placed inside the instance head symbol. Both the inside name and the above name may be empty.

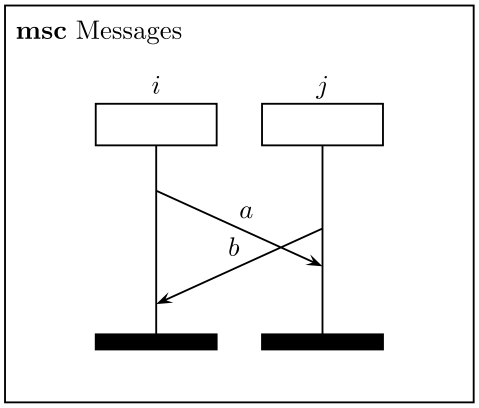
Messages Messages are drawn with the

`\mess[pos]{txt}{s}{r}[offset]`

command. The optional `pos` parameter defines the position of so-called *self messages*: messages from an instance to itself. The default value of `pos` is `l` (to the left of the instance) and another possible value is `r` (to the right of the instance). The `txt` parameter defines the label of the arrow representing the message. The `s` parameter is the nickname of the instance on which the send event occurs, i.e., the nickname of the sender. The `r` parameter is the nickname of the instance on which the receive event occurs, i.e., the nickname of the receiver. The optional parameter `offset` defines the number of levels the receive event is shifted vertically with respect to the send event. Levels are discussed in the next paragraph. Offsets are useful if two instances send messages to each other and then wait for the messages to be received. For example, Figure 2 shows messages a and b between instances i and j. The receive event of message a occurs after the send event of message b and vice versa. Both messages have `offset` $= 2$ in order to place the receive events two levels below the send events.

Levels The height of an `msc` environment is determined by the number of *levels* and a fixed amount of vertical space above and below the first and last level, respectively. Levels are created by the `\nextlevel[num]` command. The optional parameter denotes the number of levels to be added; its default value is 1. Levels are used to order events in time. Recall that time runs from top to bottom, i.e., it runs from higher levels to lower levels. Events in the same level are drawn at equal vertical distance from the top of the MSC. The send event of a message will always be drawn in the current level. The receive event of a message can be drawn in another level using the `offset` parameter of the `\mess` command. Note that levels are not part of the MSC language, they are just an implementation means to draw MSCs.

Using the commands described so far, we can generate the MSC of Figure 1. The LaTeX input to generate that MSC is given below. The length `\instdist`, used in the last `\mess` command, defines the distance between instances of an MSC and is one of the parameters to configure the MSC macro package. Here, it is used to create a `\parbox` that is



```
\begin{msc}{Messages}
\declinst{i}{$i$}{}
\declinst{j}{$j$}{}
\mess{$a$}{i}{j}[2]
\nextlevel
\mess{$b$}{j}{i}[2]
\nextlevel[2]
\end{msc}
```

Figure 2: Using non-zero message offsets.

15% smaller than the distance between the instances of the MSC.

```
\begin{figure}[htb]
\begin{center}

\begin{msc}{ftp login to CTAN archive}
\declinst{usr}{User}{}
\declinst{ftp}{ftp client}{}
\declinst{ctan}{ftp.tex.ac.uk}{CTAN}

\mess{ftp.tex.ac.uk}{usr}{ftp}
\nextlevel
\mess{connect}{ftp}{ctan}
\nextlevel
\mess{getlogin}{ctan}{ftp}
\nextlevel
\mess{login}{ftp}{usr}
\nextlevel
\mess{anonymous}{usr}{ftp}
\nextlevel
\mess{anonymous}{ftp}{ctan}
\nextlevel
\mess{Ok}{ctan}{ftp}
\nextlevel[2]
\mess{\parbox[b]{.85\instdist}
  {\centering command successful}}{ftp}{usr}

\end{msc}
\end{center}
\end{figure}
```

Actions Actions are events that can be used to model internal activity of a particular instance. Actions are defined with the `\action{txt}{nn}` command. The `txt` parameter defines the text to be placed inside the action symbol. The `nn` parameter is the nickname of the instance that executes the action. The action will be drawn at the current level with its top aligned with send events at the same level.

For example, suppose *CTAN* has to do some computations in order to determine if the anonymous login is allowed. The computation could be modeled by a *check* action, as depicted in Figure 3. The LaTeX code for the MSC of Figure 3 is:

```
\begin{msc}{Action}
\declinst{ftp}{ftp client}{}
\declinst{ctan}{ftp.tex.ac.uk}{CTAN}
\nextlevel
\mess{anonymous}{ftp}{ctan}
\nextlevel
\action{Check}{ctan}
\nextlevel[2]
\mess{Ok}{ctan}{ftp}
\end{msc}
```

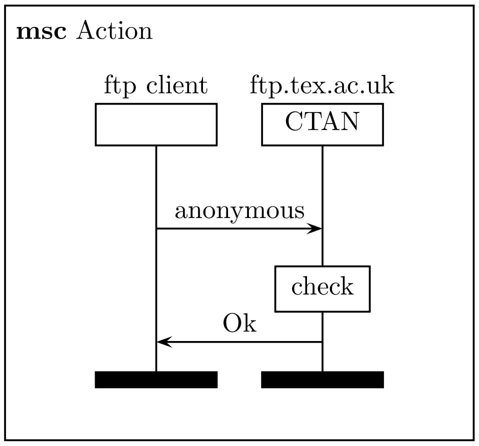


Figure 3: An MSC with an action.

Regions Another way to model internal activity, or inactivity, is by using *regions*. Regions are defined by the `\regionstart{regtype}{nn}` and the `\regionend{nn}` commands. The `regtype` parameter defines the type of the region: `activation`, `coregion` (which will not be discussed in this paper), or `suspension`. The `nn` parameter is the nickname of the instance on which the region should be drawn. If an instance is active, e.g., doing some computations, this can be modeled by an *activation region*. If an instance is inactive, e.g., waiting for results, this can be modeled by a *suspension region*. For example, the computation of the *CTAN* could be modeled by an activation region. Furthermore, the *ftp client* is inactive during this computation, which could be modeled by a suspension region. Figure 4 shows the resulting MSC. The LaTeX code for the MSC of Figure 4 is:

```
\begin{msc}{Regions}
\declinst{usr}{User}{}
\declinst{ftp}{ftp client}{}
\declinst{ctan}{ftp.tex.ac.uk}{CTAN}
\regionstart{activation}{ftp}
\mess{anonymous}{usr}{ftp}
\nextlevel
\regionstart{suspension}{ftp}
\regionstart{activation}{ctan}
\mess{anonymous}{ftp}{ctan}
\nextlevel[2]
\mess{Ok}{ctan}{ftp}
\regionend{ctan}
\regionstart{activation}{ftp}
\nextlevel
\mess{\parbox[b]{.85\instdist}
  {\centering command successful}}{ftp}{usr}
\regionend{ftp}
\end{msc}
```

Note that the space between the activation region of the *ftp client* and the *anonymous* message from the *ftp client* to *CTAN* is very small. In the next paragraph we will show how redefining one of the MSC *parameters* can increase this space.

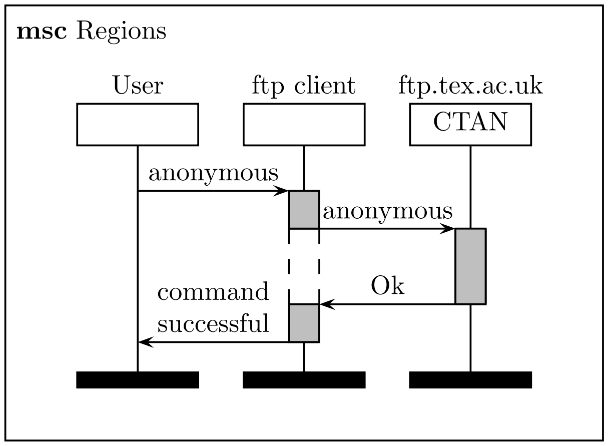


Figure 4: An MSC with activation and suspension regions.

MSC parameters The MSC macro package has almost 30 parameters to change the layout of MSCs. For example, the width of instances, the distance between instances, the distance between the head symbols and the MSC frame, and the width and height of action symbols can all be changed. These parameters are represented by the LaTeX lengths `\instwidth`, `\instdist`, `\topheaddist`, `\actionwidth`, `\actionheight`, respectively. For instance, in the MSC of Figure 4, the distance between the instances should be slightly bigger, in order to increase the space between the activation region of the *ftp client* and the *anonymous* message from the *ftp client* to *CTAN*. Figure 5 shows the same MSC, but now the distance between instances is increased by 10%. The LaTeX code for this MSC is the code for Figure 4 in which just after the line `\begin{msc}{regions}` the line `\setlength{\instdist}{1.1\instdist}` is included.

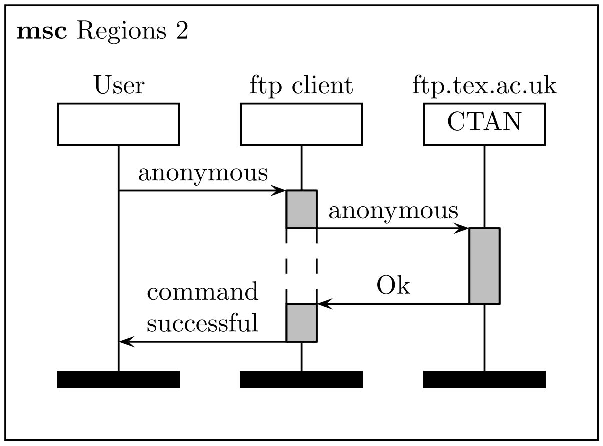


Figure 5: An MSC with larger distance between instances.

The location where an MSC parameter is changed in the LaTeX source document determines its effect. Since the MSC parameters are normal LaTeX macros or LaTeX lengths, the normal LaTeX scoping rules for these entities apply. For example, if a length parameter is changed outside any LaTeX environment, its effect is visible for all `msc` environments defined after the change. However, if it is changed inside an `msc` environment, its effect is only visible for that MSC.

Since there are many parameters to configure the MSC macro package, there are three predefined parameter settings to generate small, normal, or large MSCs. The command `\setmscvalues{parset}` can be used to change the selected parameter settings. The `parset` parameter should be `small`, `normal`, or `large`. The default setting is `normal`.

4 Implementation

In this section we will describe some aspects of the implementation of the MSC macro package.

Drawing MSCs In general, and as shown by the examples of the previous sections, an MSC consists of a number of vertically oriented instances that are connected by horizontally oriented messages. So, the width of an MSC is related to the number of instances and the height of an MSC is related to the number of (ordered) messages. Based on this observation, there are several implementations possible.

To define the width of an MSC, we could use an additional parameter of the `msc` environment. However, this strategy has some drawbacks. First of all, an extra parameter, the horizontal position, is required to declare instances. Furthermore, this parameter probably changes whenever a new instance is added to the left of an existing instance. Finally, the user should calculate the value of this parameter carefully in order to get evenly spaced instances.

Therefore, we chose to compute the width of an MSC based on the number of instances declared by the user and the, user definable, `\instdist` length that defines the distance between instances. This decision does not violate requirement 3 of Section 2, no automatic structuring and layout, since the number of instances is under control of the user. Furthermore, the user can adjust the space to the left of the first instance and the space to the right of the last instance by redefining the length parameter `\envinstdist`.

The messages are partially ordered based on the relative position of their send and receive events on instances. We could have decided to provide commands to order the events and then let the package compute the final layout of the MSC. However, apart from the fact that this computation is not trivial, this strategy fails with respect to requirement 3: no automatic structuring and layout.

Another strategy is to use an extra parameter of the `msc` environment to define the vertical size of an MSC. There are several drawbacks to this approach. First of all, the vertical size has to be computed. Secondly, commands to draw messages, actions, regions, etc., should have one or more additional parameter to indicate the vertical position at which they should be drawn. Finally, if a new message is to be added somewhere in the MSC, the vertical placement parameter of commands below the new message should probably be updated.

Therefore, we chose to only provide a command, `\nextlevel`, to advance the current height of the MSC. By increasing the current height between two messages, the partial order can be defined. Furthermore, one can easily add new messages to the MSC at any vertical position without having to change parameters of existing messages.

These decisions resulted in an `msc` environment in which the MSC is drawn in a top-left bottom-right fashion.

Nicknames As explained above, the MSC macro package uses nicknames to identify instances. If an instance is declared, the following attributes are associated to its nickname:

- The inside name,
- The above name,
- The width of the instance line,
- A flag indicating if it is a normal or a fat instance,
- The left, center, and right x-position of the instance,
- The y-position from which this instance still has to be drawn,
- The style of the instance line, and
- The style of the region of the instance.

The `\declinst` command defines the attributes using the following TeX code pattern:
`\expandafter\def`
 `\csname inst`⟨*attrnickname*⟩`\endcsname`
 `{`⟨*value*⟩`}` where ⟨*attrnickname*⟩ is the concatenation of the attribute, e.g., `abname` (above name), and the nickname and where ⟨*value*⟩ is the value of the attribute. For instance, the declaration

```
\declinst{usr}{User}{}
```

defines the following commands:
`\instabnameusr`, `\instinnameusr`, `\instbarwidthusr`, `\instisfatusr`, `\instxposusr`, `\instlxposusr`, `\instrxposusr`, `\instyposusr`, `\instlinestyleusr`, and `\instregionstyleusr`.

For each instance attribute, there is an internal command to read the value of the attribute. For example, to read the value of the above name of instance `usr`, one should use `\msc@instabname{usr}`. For some attributes, like the current y-position, there is a command to change the value of the attributes. For example, to change the y-position of instance `usr` to the value `y`, one could use `\msc@setinstypos{usr}{y}`.

Underlying drawing engine The MSC macro package uses the pstricks package, see [6] or Chapter 4 of [3], to draw lines, arrows, and frames. This package is now commonly available in LaTeX distributions, so relying on this package does not violate requirement 5. A drawback of pstricks is that it is incompatible with PDFLaTeX. Consequently, our MSC macro package is incompatible with PDFLaTeX, too. However, there are other ways to generate pdf from LaTeX documents. One option is to first convert the dvi file into PostScript, e.g., using *dvips*, and then convert the PostScript file into pdf, e.g., using the *ps2pdf* utility included in ghostscript distributions (`http://www.cs.wisc.edu/~ghost/`).

5 Availability

The MSC macro package is freely available at CTAN, see directory `macros/latex/contrib/supported/msc`, and at `http://www.win.tue.nl/~sjouke/mscpackage.html`. It is distributed under the *LaTeX Project Public License*, see `http://www.latex-project.org/lppl.txt`. Documentation of the package consists of a user manual [2] and a reference manual [1]. These documents are included in the distribution.

6 Conclusions

The MSC macro package enables users to include MSCs in LaTeX documents. Furthermore, the MSCs have a consistent layout that can be configured by an appropriate set of parameters. The package supports almost the complete ITU standard of the MSC language, including MSC documents and high level MSCs (which were not discussed in this paper).

1. The abstraction level of the MSC macro package is as desired.
2. The user has full control over the relative position of instances, messages, etc.
3. Changing MSCs, e.g., adding extra instances or messages, is easy and does not require computations by the user.
4. The MSC macro package is highly configurable. There are about 30 user definable length parameters and a small number of text parameters.

The developers of the MSC macro package consider the package more or less complete. Therefore, the only changes to the package will be bug fixes and/or code documentation.

References

[1] Victor Bos and Sjouke Mauw. *A LaTeX macro package for Message Sequence Charts—Reference Manual—Describing MSC macro package version 1.5*, April 2002. Included in MSC macro package distribution.

[2] Victor Bos and Sjouke Mauw. *A LaTeX macro package for Message Sequence Charts—User Manual—Describing MSC macro package version 1.5*, April 2002. Included in MSC macro package distribution.

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, 1994.

[4] ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC).* ITU-TS, Geneva, 2001.

[5] E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on message sequence charts (msc'96). In *FORTE*, 1996.

[6] Timothy van Zandt. Pstricks, PostScript macros for Generic TeX. User's Guide, available at every CTAN site, (`CTAN:graphics/pstricks/`), 1993.

◇ Sjouke Mauw
Computing Science Department
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB, Eindhoven
The Netherlands
`sjouke@win.tue.nl`

◇ Victor Bos
Software Construction Laboratory
Turku Centre for Computer Science
Lemminkäisenkatu 14 A
FIN-20520, Turku
Finland
`v.bos@abo.fi`

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 :-). If 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 ε-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 `\tracingassigns`, which will show you changes to register values and changes to control sequences when they happen, and `\tracinggroups`, 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

```
\tracingassigns=1\tracinggroups=1\tracingall
```

That in itself is already a nuisance (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
{\relax}

\setlength  ->\calc@assign@skip

\calc@assign@skip ->\calc@assign@generic \calc@Askip \calc@Bskip

\calc@assign@generic #1#2#3#4->\let \calc@A #1\let \calc@B #2\expandafter \calc
@open \expandafter (#4!\global \calc@A \calc@B \endgroup #3\calc@B
#1<-\calc@Askip
#2<-\calc@Bskip
#3<-\linewidth
#4<-1cm
{\let}
{\let}
{\expandafter}
{\expandafter}

\calc@open (->\begingroup \aftergroup \calc@initB \begingroup \aftergroup \calc
@initB \calc@pre@scan
{\begingroup}
{\aftergroup}
{\begingroup}
{\aftergroup}
```

[*]This file has version number 1.0a trace LaTeX code, last revised 2000/02/16.

```
\calc@pre@scan #1->\ifx (#1\expandafter \calc@open \else \ifx \widthof #1\expan
dafter \expandafter \expandafter \calc@textsize \else \calc@numeric \fi \fi #1
#1<-1
{\ifx}
{false}
{\ifx}
{false}

\calc@numeric ->\afterassignment \calc@post@scan \global \calc@A
{\afterassignment}
{\global}
{\fi}
{\fi}

\calc@post@scan #1->\ifx #1!\let \calc@next \endgroup \else \ifx #1+\let \calc@
next \calc@add \else \ifx #1-\let \calc@next \calc@subtract \else \ifx #1*\let
\calc@next \calc@multiplyx \else \ifx #1/\let \calc@next \calc@dividex \else \i
fx #1)\let \calc@next \calc@close \else \calc@error #1\fi \fi \fi \fi \fi \fi \
calc@next
#1<-!
{\ifx}
{true}
{\let}
{\else}
{\endgroup}
{restoring \calc@next=undefined}

\calc@initB ->\calc@B \calc@A
{\skip44}
{\global}
{\endgroup}
{restoring \skip44=0.0pt}

\calc@initB ->\calc@B \calc@A
{\skip44}
{\dimen27}
```

Do you still remember what I was talking about?

No? We're trying to find a problem in macro code without having to scan too many uninteresting lines. To make this possible we have to redefine a number of key commands to turn tracing off temporarily in the hope that this will reduce the amount of noise during the trace. For example, if we change one of the `calc` internals slightly, the above tracing output can be reduced to:

```
\setlength ->\protect \setlength
{\relax}

\setlength  ->\calc@assign@skip

\calc@assign@skip ->\calc@assign@generic \calc@Askip \calc@Bskip

\calc@assign@generic #1#2#3#4->\let \calc@A #1\let \calc@B #2\expandafter \calc
@open \expandafter (#4!\global \calc@A \calc@B \endgroup #3\calc@B
#1<-\calc@Askip
#2<-\calc@Bskip
#3<-\linewidth
#4<-1cm
{\let}
{\let}
{\expandafter}
{\expandafter}

\calc@open (->\begingroup \conditionally@traceoff \aftergroup \calc@initB \begi
ngroup \aftergroup \calc@initB \calc@pre@scan

\conditionally@traceoff ->\tracingrestores \z@ \tracingcommands \z@ \tracingpag
es \z@ \tracingmacros \z@ \tracingparagraphs \z@
```

```
{\tracingrestores}
{\tracingcommands}
{restoring \tracingrestores=1}

\calc@initB ->\calc@B \calc@A
{\skip44}
{\dimen27}
```

Still a lot of noise but definitely preferable to the original case.

I redefined those internals that I found most annoyingly noisy. There are probably many others that could be treated in a similar fashion, so if you think you found one worth adding please drop me a short note.

* * *

`\traceon` `\traceoff` The package defines the two macros `\traceon` and `\traceoff` to unconditionally turn tracing on or off, respectively. `\traceon` is like `\tracingall` but additionally adds `\tracingassigns` and `\tracinggroups` if the ε-TEX program (in extended mode) is used. And `\traceoff` will turn tracing off again, a command which is already badly missing in plain TEX, since it is often not desirable to restrict the tracing using extra groups in the document.

`\conditionally@traceon` `\conditionally@traceoff` There are also two internal macros that turn tracing on and off, but only if the user requested tracing in the first place. These are the ones that are used internally within the code below.

Since the package overwrites some internals of other packages you should load it as the last package in your preamble using `\usepackage{trace}`.

A sample file

The following small test file shows the benefits of the `trace` package. If one uncomments the line loading the package, the amount of tracing data will be drastically reduced. Without the `trace` package we get 6573 lines in the log file; adding the package will reduce this to 1593 lines.

```
\documentclass{article}
\usepackage{calc}
%\usepackage{trace} % uncomment to see difference

\begin{document}
\ifx\traceon\undefined \tracingall \else \traceon \fi

\setlength\linewidth{1cm}

$foo=\bar a$

\small \texttt{\$}  \stop
```

Implementation

This package is for use with LATEX (though something similar could be produced for other formats).

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[1998/12/01]
```

`\if@tracing` We need a switch to determine if we want any tracing at all. Otherwise, if we use `\traceoff`...`\traceon` internally, we would unconditionally turn on tracing even when no tracing was asked for in the first place.

```
\newif\if@tracing
```

`\traceon`
`\conditionally@traceoff`

As stated in the introduction, the amount of tracing being done should depend on the formatter we use. So we first test if we are running with ε-TEX in extended mode. In the latter csse the command `\tracinggroups` is defined.[1]

```
\ifx\tracinggroups\undefined
```

If we are using standard TEX then `\traceon` is more or less another name for `\tracingall`. The only differences are that we set the above `@tracing` switch to true and reorder the assignments within it somewhat so that it will output no tracing information about itself. In contrast, `\tracingall` itself produces

```
{vertical mode: \tracingstats}
{\tracingpages}
{\tracinglostchars}
{\tracingmacros}
{\tracingparagraphs}
{\tracingrestores}
{\errorcontextlines}

\showoutput ->\tracingoutput \@ne \showboxbreadth \maxdimen \showboxdepth \maxd
imen \errorstopmode \showoverfull
{\tracingoutput}
{\showboxbreadth}
{\showboxdepth}
{\errorstopmode}


\showoverfull ->\tracingonline \@ne
{\tracingonline}
```

Which is quite a lot given that none of it is of any help to the task at hand. In contrast `\traceon` will produce nothing whatsoever since the noise generating switches are set at the very end.

```
  \def\traceon{%
```

We start by setting the `@tracing` switch to signal that tracing is asked for. This is then followed by setting the various tracing primitives of TEX.

```
    \@tracingtrue
    \tracingstats\tw@
    \tracingpages\@ne
    \tracinglostchars\@ne
    \tracingparagraphs\@ne
    \errorcontextlines\maxdimen\showoutput
    \tracingmacros\tw@
    \tracingrestores\@ne
    \tracingcommands\tw@
  }
```

Now what should `\conditionally@traceoff` do in this case? Should it revert all settings changed by `\traceon`? It should not, since our goal is to shorten the trace output, thus setting all of the uninteresting values back makes the output unnecessarily longer. Therefore we restrict ourself to those `\tracing...` internals that really contribute to listings like the above.

And one additional point is worth mentioning. The order in which we turn the tracing internals off has effects on the output we see. So what needs to be turned off first? Either `\tracingrestores` or `\tracingcommands`; it makes no difference which, as long as they both come first. This is because those two are the only tracing switches that produce output while tracing the command `\conditionally@traceoff` itself (see example on page 95).

[1] If some package writer has defined that command name for some reason—too bad—then we make the wrong deduction from this fact and as a result the package will fail.

In principle we would need to test the `@tracing` switch to see if there is anything to turn off; after all, this is the conditional trace off. However this would lead to extra output if we are currently tracing so we skip the test and instead accept that in case we are not doing any tracing we unnecessarily set the tracing primitives back to zero (i.e., the value they already have).

```
\def\conditionally@traceoff{%
  \tracingrestores\z@
  \tracingcommands\z@
  \tracingpages\z@
  \tracingmacros\z@
  \tracingparagraphs\z@
```

As remarked above there are more tracing switches set by `\traceon`, however there is no point in resetting `\tracingstats` or `\tracinglostchars` so we leave them alone.

```
% \tracingstats\z@
% \tracinglostchars\z@
```

Since this is the command that only conditionally turns off tracing we do not touch the `@tracing` switch. This way a `\conditionally@traceon` will be able to turn the tracing on again.

```
    }
```

That covers the case for the standard TEX program. If `\tracingsgroups` was defined we assume that we are running with ε-TEX in extended mode.

```
\else
```

In that case `\traceon` does more than `\tracingall`: it also turns on tracing of assignments and tracing of grouping.[2] To keep tracing at a minimum `\tracingassigns` should be turned on last (in fact like before we disassemble `\tracingall` and reorder it partially).

```
\def\traceon{%
  \@tracingtrue
  \tracingstats\tw@
  \tracingpages\@ne
  \tracinglostchars\@ne
  \tracingparagraphs\@ne
  \errorcontextlines\maxdimen\showoutput
  \tracingmacros\tw@
  \tracinggroups\@ne
  \tracingrestores\@ne
  \tracingcommands\tw@
  \tracingassigns\@ne
}
```

When turning tracing off again we now also have to turn off those additional tracing switches. But what to turn off in what order? Since `\tracingassigns` is quite noisy (two lines of output per assignment) and the whole command expansion consists of assignments, we had best start with this switch and follow it again by `\tracingrestores` and `\tracingcommands`. The rest can be in any order, it doesn't make a difference.

With the same reasoning as before we omit testing for the `@tracing` switch and always set the primitives back to zero.

```
\def\conditionally@traceoff{%
  \tracingassigns\z@
  \tracingrestores\z@
```

[2] These are my personal preference settings; ε-TEX does in fact offer some more tracing switches and perhaps one or or more of them should be added here as well.

```
    \tracingcommands\z@
    \tracingpages\z@
    \tracingmacros\z@
    \tracingparagraphs\z@
    \tracinggroups\z@
  }
```

This concludes the part that depends on the formatter being used.

```
\fi
```

`\traceoff` `\conditionally@traceon` Above we have defined `\conditionally@traceoff` and `\traceon` so now we have to define their counterparts.

To stop tracing unconditionally we call `\conditionally@traceoff` (which in reality is far from conditional except for not setting the `@tracing` switch :-) and then reset the `@tracing` switch to false.

```
\def\traceoff{\conditionally@traceoff \@tracingfalse}
```

Now the `\conditionally@traceon` command will look at the `@tracing` switch and if it is true it will call `\traceon` to restart tracing (note that the latter command unnecessarily sets the switch to true as well). The reason for the `\expandafter` is to get rid of the `\fi` primitive which would otherwise show up in the tracing output (and perhaps puzzle somebody).

```
\def\conditionally@traceon{\if@tracing \expandafter \traceon \fi}
```

The rest of the package now consists of redefinitions of certain commands to make use of `\conditionally@traceoff`.

Taming `calc`

`\calc@open` Near the start of parsing a calc expression the macro `\calc@open` is called. Since it already involves a group it is perfectly suitable for our task—we don't even have to restart the tracing as this is done automatically for us.

```
\def\calc@open({\begingroup
   \conditionally@traceoff
   \aftergroup\calc@initB
   \begingroup\aftergroup\calc@initB
   \calc@pre@scan}
```

Making NFSS less noisy

`\define@newfont` Whenever NFSS determines that the font currently asked for is not already loaded, it will start looking through font definition files and then load the font. This results in a very large number of tracing lines which are not normally of interest (unless there is a bug in that area—something we hope should have been found by now). Again the code already contains its own group so we only have to turn the tracing off.

```
\def\define@newfont{%
  \begingroup
    \conditionally@traceoff
    \let\typeout\@font@info
    \escapechar\m@ne
    \expandafter\expandafter\expandafter
       \split@name\expandafter\string\font@name\@nil
      \try@load@fontshape % try always
    \expandafter\ifx
       \csname\curr@fontshape\endcsname \relax
      \wrong@fontshape\else
      \extract@font\fi
  \endgroup}
```

`\frozen@everymath`
`\frozen@everydisplay`

At the beginning of every math formula NFSS will check whether or not the math fonts are properly set up and if not will load whatever is needed. So we surround that part of the code with `\conditionally@traceoff` and `\conditionally@traceon` thereby avoiding all this uninteresting output.

```
\frozen@everymath =
   {\conditionally@traceoff \check@mathfonts \conditionally@traceon
    \the\everymath}
\frozen@everydisplay =
   {\conditionally@traceoff \check@mathfonts \conditionally@traceon
    \the\everydisplay}
```

Checking for italic corrections

`\maybe@ic@`

When executing `\textit` or its friends, LaTeX looks ahead to determine whether or not to add an italic correction at the end. This involves looping through the `\nocorrlist` which outputs a lot of tracing lines we are normally not interested in. So we disable tracing for this part of the processing.

```
\def \maybe@ic@ {%
  \ifdim \fontdimen\@ne\font>\z@
  \else
    \conditionally@traceoff
    \@tempswatrue
    \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
        \nocorrlist
    \do \t@st@ic
    \if@tempswa \sw@slant \fi
    \conditionally@traceon
  \fi
}
```

◇ Frank Mittelbach

Abstracts

Les Cahiers GUTenberg
Contents of Issue 35/36 (May 2000) and Issue 37/38 (December 2000)

Double issue 35/36 (May 2000): Proceedings of GUT 2000 “LaTeX and XML: Cooperation on the Internet”

Martial Chartoire, Éditorial : LaTeX et XML : coopération pour l’internet; pp. 3–4

The subject is the theme of the GUTenberg 2000 annual meeting in Toulouse. This issue of the *Cahiers* includes a good number of conference papers, which demonstrate that this cooperation has become quite effective.

The editor muses about how far things have come from the 1994 GUTenberg meeting, which had as its theme “Distribution of electronic documents”: from the advent of the Web and HTML, and now XML and XSL, which bring screen output closer to that which LaTeX can produce. XML has been adopted so quickly that four articles in this issue are already discussing its application to various projects.

The rest of the editorial touches on the various articles in the issue, and closes with words of thanks to those who had helped organize the 2000 conference in Toulouse.

Jacques André and Pascale Laurent, Publications scientifiques électroniques : quoi et comment ? (résumé étendu) [Electronic scientific publication: what and how? (extended summary)]; pp. 5–13

This is a survey about scientific electronic journals and some current international experiments, the results of which too often remain within the domain of computer and documentation specialists.

[Based on authors’ résumé and abstract]

Frédéric Boulanger and Yolaine Bourda, Documentation de projets en XML [Documenting XML Projects]; pp. 15–23

The lack of documentation in most software projects, and particularly in our students’ projects, led us to develop a way to document software by annotating it in comments. We generalize this approach to be able to document code in any language and to create documentation in any format. However, XML is our preferred choice since this work is part of a larger document processing project.

[Authors’ abstract]

Viviane Boulétreau and Jean-Paul Ducasse, La production de documents électroniques structurés à grande échelle : la diffusion électronique des thèse universitaires [Large-scale production of structured electronic documents: electronic distribution of university theses]; pp. 25–35

The Université Lumière Lyon 2 has been working for a year on a project to distribute theses electronically. This requires that the document format meet three crucial criteria: long-term availability, efficient distribution, and ease of access. The article provides an overview of the current project, plans for future development (in the short term), and pointers to the long-term role of such documents in information exchange.

[Based on authors’ introduction]

Yolaine Bourda and Marc Hélier, Métadonnées, RDF et documents pédagogiques [Metadata, RDF and teaching documents]; pp. 37–52

In many fields, such as education, electronic documents do not pay for themselves as they should (reused, found ...). One possible solution is to rely on metadata, RDF and XML. The aim of this paper is to present the idea of metadata and to emphasize the importance of standardization. For a given set of metadata, many implementations using XML are possible. This multiplicity has its drawbacks. A unique implementation may be obtained by means of RDF. The Dublin Core initiative and the Learning Objects, which are under construction by the IEEE, are given as examples of this process. [Authors’ abstract]

Philip Taylor, Jiří Zlatuška and Karel Skoupý, The NTS Project: From conception to implementation; pp. 53–77

This 25-page article provides an overview of the NTS project: its history, development approaches and choices, current status, and impending completion.

[Based on authors’ introduction]

David Carlisle, Michel Goossens and Sebastian Rahtz, De XML à PDF via `xmltex`, XSLT et PassiveTeX [From XML to PDF via `xmltex`, XSLT and PassiveTeX]; pp. 79–114

This article introduces `xmltex`, a TeX macro package that parses an XML document and typesets it under the control of configuration files. We also discuss PassiveTeX, a library of TeX macros based on `xmltex`, that processes XML documents containing XSL formatting objects and generates PDF or DVI output. We compare these two approaches with a direct translation of the XML source file into LaTeX. We show examples of these techniques for the TEI, DocBook and MathML DTDs. The appendix gives details about the `xmltex` commands.

[Authors' abstract]

Frank Mittelbach, David Carlisle and Chris Rowley, New Interfaces for LaTeX Class Design; pp. 115–120

Traditional LaTeX class files typically implement one fixed design via ad hoc, and often low-level, (LA)TeX code. This style of implementation makes it much harder than is either desirable or necessary to produce classes that implement a specific visual design. This article introduces some extensions to LaTeX that will help to provide a new, more declarative interface that can be used in class files. It is based on the idea of a *template*, which describes how to carry out some action but which provides some flexibility since its code uses the values of a set of named (keyword) parameters.

[Based on authors' introduction]

Benjamin Bayart, Nouvelles pistes pour une distribution de TeX [New approaches for TeX distributions]; pp. 121–132

We begin with a quick overview of the situation which led to the idea of a new type of TeX distribution. Based on quite different problems, a very old discussion about defining a TPM (TeX Package Manager) had quickly led to quite similar conclusions. The basic principles behind FDNTeX (FDN = French Data Network) will be presented and explained in detail.

[Translation of French résumé]

Michel Cubero-Castan, PolyDoc : un exemple d'application XML pour la création personnalisée de polycopiés [PolyDoc: Example of an XML application for creating customized copies]; pp. 133–155

This article presents PolyDoc, a Java application based on the W3C's Document Object Model (DOM), which allows translation of a document from XML to HTML, LaTeX, Open e-Book, ..., or again into XML (with a different DTD). Using PolyDoc, we describe a document production process with three stages: contents written in XML, global customized formatting in Java, production of the result via HTML, LaTeX, ...

[Based on French résumé]

Roberta Faggian, Integration of resources on the World Wide Web using XML; pp. 157–167

An initiative to explain high energy physics to the general public has been started at CERN. The use of the Web has been identified as crucial to the success of this initiative. An integral part of this project is the construction of a Web-based information system that collects many different resources on the Web (information published by many European and US particle physics institutes). This paper proposes a solution to the problem of integration and reuse of heterogeneous information by enriching existing content semantic with metadata in order to improve understanding and discovery. The main part of the work is the study of the RDF standard for representing metadata, and its implementation using XML syntax. [Author's abstract]

Jean-Michel Hufflen, Typographie : les conventions, la tradition, les goûts, ... et LaTeX [Typography: Conventions, traditions, tastes ... and LaTeX]; pp. 169–214

This article is a transcript of a tutorial designed to show that learning typographic rules, even learning both French and English rules, is not that difficult. The article also provides some starting points for using the french and babel packages, the one for writing in French, the other for dealing with most other languages in a relatively homogeneous fashion. The article then shows how to organize a new class file as well as a new multilingual bibliography style.

[Translation of French résumé]

Christian Rossi, Le CTAN Navigator [The CTAN Navigator]; pp. 215–221

The CTAN Navigator (`http://ctan.loria.fr`) is a Web server that provides a set of tools to facilitate the search, transfer, and installation of files available from CTAN (the Comprehensive TeX Archive Network).

[Translation of French résumé]

– – ∗ – –

Double issue 37/38 (December 2000)

Jacques André, Éditorial; pp. 3–4

Jacques André cites a passage from the GUTenberg statutes, one which states that the association's aims include bringing together French-language TeX users, encouraging technical exchanges to promote the printing and distribution of scientific publications, and offering its members a specific number of services. Those aims are all represented in the various articles included in this issue.

And as do all editors, Jacques laments the fact that articles seem so rarely to arrive unsolicited — and hopes that the new millenium will see this change.

Fabrice Popineau, Affichez vos documents LaTeX sur le Web avec TeX4ht [Post your LaTeX documents on the Web with TeX4ht]; pp. 5–43

Eitan Gurari is the author of TeX4ht, a clever tool which allows TeX and LaTeX documents to be translated into HTML and XML. I'd like to show here that TeX4ht is simple to use, powerful and extensible. Let's have a look at its features.

[Author's abstract]

Hermann Zapf, Typographie des caractères romains et de la Renaissance [Typography of Roman characters and the Renaissance]; pp. 44–52

This paper is the French translation of a German paper written by Hermann Zapf in 1953, dedicated to the Renaissance humanists who defined Roman Capitals with the use of compass and rules. Zapf shows that the design of characters is not just a matter of geometry and that the re-design of a classic character must take into account the original drawings. [Author's abstract]

The article was originally published in German as "Vom Formgesetz der Renaissance Antiqua", in the 1953 issue of the annual series *Gutenberg-Jahrbuch*. The "Editorial Note" accompanying the article gives more details.

Pierre Attar and Bruno Chatel, État des recommandations XML dans le domaine documentaire [Status of XML recommendations for documents]; pp. 53–85

The purpose of this report is to present the current state of the XML standard, its power and its limits in addressing the needs of documentation applications. As well, it looks at XML parsers, to try and define their quality and efficiency.

[Translation of French résumé (abbreviated)]

Frank Mittelbach, Formater des documents ayant des flottants : un nouvel algorithme pour LaTeX 2_ε* [Formatting documents with floats: A new algorithm for LaTeX 2_ε*]; pp. 86–108

This paper describes an approach to placement of floats in multicolumn documents. The current version of LaTeX was originally written for single-column documents and extended to support two-column documents by essentially building each column independently from the other. As a result, the current system shows severe limitations in two-column mode, such as the fact that spanning floats are always deferred to at least the next page or that numbering between column floats and spanning floats can get out of sequence.

The new algorithm is intended to overcome these limitations and at the same time extend the supported class of document layouts to multiple columns with floats spanning an arbitrary number of columns. [Author's abstract]

This paper was also also presented at TUG 2000 in Oxford, and appears in *TUGboat* 21, no. 3 (2000), pp. 278–290.

– – ∗ – –

Articles from *Cahiers* issues can be found in PDF format at the GUTenberg site:

`http://www.gutenberg.eu.org/pub/gut/publications`

[Compiled by Christina Thiele]

Calendar

2001

Feb 28 – Mar 3	DANTE 2001, 24th meeting, Fachhochschule Rosenheim, Germany. For information, visit `http://www.dante.de/dante2001/`.
Mar 26 – 28	XML World Europe, Amsterdam, Netherlands. For information, visit `http://www.xmlworld.org/`.
Apr 1 – Jun 15	The Best of the Best: A traveling juried exhibition of books by members of the Guild of Book Workers. Ohio State University Library, Athens, Ohio. Sites and dates are listed at `http://palimpsest.stanford.edu/byorg/gbw`.
Apr 9 – 13	Seybold Boston, Boston, Massachusetts. For information, visit `http://www.key3media.com/seyboldseminars/boston2001/`.
Apr 29 – May 2	BachoTeX 2001, 9th annual meeting of the Polish TeX Users' Group (GUST), "Contemporary publishing TeXnology", Bachotek, Brodnica Lake District, Poland. For information, visit `http://www.gust.org.pl/BachoTeX/`.
May 14 – 17	Congrès GUTenberg 2001, "Le document au XXIe Siècle", Metz, France. For information, visit `http://www.gutenberg.eu.org/manif/gut2001/`.
Jun 6 – 8	Society for Scholarly Publishing, 23rd annual meeting, San Francisco, California. For information, visit `http://www.sspnet.org`.
Jun 13 – 17	ACH/ALLC 2001: Joint International Conference of the Association for Computers and the Humanities, and Association for Literary and Linguistic Computing, New York University, New York. For information, visit `http://www.nyu.edu/its/humanities/ach_allc2001/`.
Jul 13 – 15	TypeCon 2001, Rochester, New York. For information, visit `http://www.typecon2001.com`.

TUG 2001
University of Delaware, Newark, Delaware.
For information, visit `http://www.tug.org/tug2001/`.

Aug 6 – 10	Intermediate/Advanced LaTeX training class.
Aug 12 – 16	The 22nd annual meeting of the TeX Users Group, "2001: A TeX Odyssey".

Aug 12 – 17	Extreme Markup Languages 2001: "There's Nothing so Practical as a Good Theory", Montréal, Canada. For information, visit `http://www.gca.org`.
Aug 12 – 17	SIGGRAPH 2001, Los Angeles, California. For information, visit `http://www.siggraph.org/s2001/`.
Sep 8	WDA'2001: First International Workshop on Web Document Analysis, Seattle, Washington. For information, visit `http://www.csc.liv.ac.uk/~wda2001`.
Sep 17 – 20	XML World 2001, San Francisco, California. For information, visit `http://www.xmlworld.org/`.
Sep 23 – 27	EuroTeX 2001, "TeX and Meta: the Good, the Bad and the Ugly Bits", Kerkrade, Netherlands. For information, visit `http://www.ntg.nl/eurotex/`.
Sep 29	29th Annual General Meeting of the Danish TeX Users Group (DK-TUG), Århus, Denmark. For information, visit `http://sunsite.dk/dk-tug/`.
Oct 24 – 26	4th International Conference on The Electronic Document, Toulouse, France. For information, visit `http://www.irit.fr/CIDE2001/`.

Status as of 1 July 2002

For additional information on TUG-sponsored events listed above, contact the TUG office (+1 503 223-9994, fax: +1 503 223-3960, e-mail: `office@tug.org`). For events sponsored by other organizations, please use the contact address provided.

Additional type-related events and news items are listed in the Sans Serif Web pages, at `http://news.serifmagazine.com/`.

Owing to the lateness of this issue, please consider that all events shown for 2001 are included only "for the record".

Nov 9 – 10	ACM Symposium on Document Engineering, Atlanta, Georgia. For information, visit `http://www.documentengineering.org`.

2002

Jan 16 – Feb 20	The Best of the Best: A traveling juried exhibition of books by members of the Guild of Book Workers. Swarthmore College, Swarthmore, Pennsylvania. Sites and dates are listed at `http://palimpsest.stanford.edu/byorg/gbw`.
Feb 19 – 22	Seybold New York, New York City. For information, visit `http://www.key3media.com/seyboldseminars/ny2002/`.
Feb 20 – 23	DANTE 2002, 26th meeting, Universität Erlangen-Nürnberg, Germany. For information, visit `http://www.dante.de/dante2002/`.
Mar 13 – Apr 23	The Best of the Best: A traveling juried exhibition of books by members of the Guild of Book Workers. San Diego State University Malcolm A. Love Library, San Diego, California. Sites and dates are listed at `http://palimpsest.stanford.edu/byorg/gbw`.
Apr 29 – May 3	EuroBachoTEX 2002, 13th meeting of European TEX Users and 10th annual meeting of the Polish TEX Users' Group (GUST), "TEX and beyond", Bachotek, Brodnica Lake District, Poland. For information, visit `http://www.gust.org.pl/BachoTeX/2002/`.
May 7 – Jun 27	The Best of the Best: A traveling juried exhibition of books by members of the Guild of Book Workers. San Francisco Public Library, San Francisco, California. San Diego, California. Sites and dates are listed at `http://palimpsest.stanford.edu/byorg/gbw`.
May 29	Journée GUTenberg, "Distributions", Paris, France. For information, visit `http://www.gutenberg.eu.org/`.
May 29 – 31	Society for Scholarly Publishing, 24th annual meeting, Boston, Massachusetts. For information, visit `http://www.sspnet.org`.
Jul 12 – 14	TypeCon 2002, Toronto, Canada. For information, visit `http://www.typecon2002.com`.
Jul 21 – 26	SIGGRAPH 2002, San Antonio, Texas. For information, visit `http://www.siggraph.org/calendar/`.

TUG 2002
International Convention Centre, Trivandrum, India. For information, visit `http://www.tug.org.in/tug2002/`.

Sep 1 – 3	Tutorials: LATEX; LATEX to XML; METAPOST; the Text Encoding Initiative; TEX macro expansion.
Sep 4 – 7	The 23rd annual meeting of the TEX Users Group, "Stand up and be proud of TEX!". For information, visit `http://www.tug.org.in/tug2002/`.

Sep 19 – 22	Association Typographique Internationale (ATypI) annual conference, Rome, Italy. For information, visit `http://www.atypi.org/rome2002/`.
Sep 24 – 25	First Annual Conference, Friends of St. Bride Printing Library, London, England. For information, visit `http://www.stbride.org/conference.htm`
Oct 12	UK TUG Autumn meeting, Nottingham University. For information, contact Dick Nickalls, `dicknickalls@compuserve.com`.
Oct 14 – 17	Book History Workshop, Institue d'histoire du livre, Lyons, France. For information, visit `http://ihl.enssib.fr`.
Nov 8 – 9	ACM Symposium on Document Engineering, McLean, Virginia. For information, visit `http://www.sdml.cs.kent.edu/doceng2002/`.

2003

TUG 2003
Outrigger Waikoloa Beach Resort, Big Island, Hawai'i.

Jul 20 – 24	The 24th annual meeting of the TEX Users Group. For information, visit `http://www.tug.org.in/tug2003/`.

Sep 4 – 7	the TEX Users Group, "Stand up and be proud of TEX!", Trivandrum, Kerala, India. For information, visit `http://www.tug.org.in/tug2002/`.
Jul 27 – Aug 1	SIGGRAPH 2003, San Diego, California. For information, visit `http://www.siggraph.org/calendar/`.

- PLEASE POST -

Ahoy! Come join us for the 22nd Annual Meeting & Conference of the TeX Users Group

For more information see:
http://www.tug.org/tug2001

August 12-16, 2001

UNIVERSITY OF DELAWARE
Clayton Hall Conference Center
Newark, Delaware, USA
[nearest airports PHL or BWI]

REGISTER NOW!

ACCOMMODATION [ask for Extended Stay or University of Delaware rates]

Embassy Suites, Newark-Wilmington South, DE
tel: +1 302-368-8000; fax +1-302-368-8975
Sleep Inn, Newark, DE
tel: +1 302-453-1700; fax +1-302-453-1710

REGISTRATION ON-LINE [$350 (members)/$425 (non-members) USD]

TEX USERS GROUP
TUG office
P.O. Box 2311 • Portland • OR 97208-2311 • USA
email: office@tug.org
tel: +1-503-223-9994; fax: +1-503-223-3960

PRE-CONFERENCE: Intermediate/Advanced LaTeX Classes; Aug 6-10, 2001 [$325 (members)/$350 (non-members) USD]

ORGANIZERS: Sue DeMeritt, Stephanie Hogue, Wendy McKay, Patricia Monohon, Heidi Seistrich, Anita Schwartz • **tug2001@tug.org**

TUG Business

Minutes of TEX Users Group Annual General Meeting held on 15 August 2000; Oxford, England

Susan DeMeritt, for
Arthur Ogawa, Secretary

Call to Order

The meeting was called to order at 2:30 p.m. by Mimi Jett, President of TUG.

Reading of Last Year's Minutes

Susan DeMeritt read the minutes of the 1999 Annual General Meeting held in Vancouver, B.C., Canada.

Board of Directors Report

- Robin Laakso presented membership statistics, explained the tasks involved in her job as office manager and presented the office budget information.
- Jonathan Fine stated that he feels that information regarding the budget is not readily available. Robin Fairbairns countered that he likes the way the Board is being run and that he does not want to know the day-to-day dealings, he wants to know the bottom line. Robin Laakso expressed gratitude for Robin Fairbairns' comments and stated that she enjoys working with TUG.
- Barbara Beeton reported that next year is an election year. Five board positions, plus that of President, will be open. Three of the five board positions will be open because terms will end for Arthur Ogawa, Patricia Monohon, and Petr Sojka. Formal announcements will be sent out on October 1 with a mid-January deadline for receipt of nominations. Barbara Beeton discussed the possibility of electronic balloting.
- Susan DeMeritt reported that the 2001 meeting will be held at the University of Delaware, in Newark, Delaware. Discussion was started about the 2002 meeting: proposals have been received for Ireland and India. Susan DeMeritt will put something on the web site where people can let us know whether they would be able to attend if it were held in India.
- Kaja Christiansen reported the the TUG Web server will be moving to Aarhus, Denmark. The transition should be smooth. Three new systems are going to be purchased this year: one for the Web server, one for CTAN and one for the office.

Training and Education

Mimi Jett reported that TUG is once again going to offer training, seminars, and workshops. These will be held twice a year in the US, and a site is also being sought in the UK.

Promotional Materials

TUG will collaborate with other TEX groups on putting out new promotional materials.

At this point, Jonathan Fine stood up and stated a list of complaints that he wanted to have heard.

Financial Report

Don DeLand gave an overview of the budget, with details on the LATEX3 fund and *TUGboat* expenses. [See the next page for the report.]

Mimi Jett announced that $5,000 would be given to the NTS project.

Don DeLand gave an overview of current budget info.

Mimi Jett announced the gift of Metafog to CyrTUG and GUST.

Old Business

There was no old business to discuss.

New Business

Jonathan Fine asked if a membership list could be created. Don DeLand explained that because of privacy laws, especially in Europe, we may not be able to make the list generally available. Sebastian Rahtz suggested that a member should take on that task, not a Board member. Ahmed Hindawi volunteered.

Nelson Beebe asked about the backlog of *TUGboat*. Mimi Jett mentioned that part of the problem was not enough articles to print and encouraged the membership to start writing more.

Jonathan Fine asked if it would be possible to get *TUGboat* without being a member. It was stated that *TUGboat* is a benefit of membership.

Once membership increases with training, conferences, and seminars the readership will increase as well.

The meeting was closed at 2:40 p.m. by Mimi Jett.

◇ Susan DeMeritt, for
Arthur Ogawa, Secretary

Financial Statement, 2000

Donald DeLand

Report from the TUG Treasurer

The TeX Users Group is a 501(C)(6) not-for-profit corporation under the Internal Revenue code. The association is funded primarily by annual dues from members and to a lesser degree by product sales and interest income. Membership dues of $132,470 are 4.5% higher than they were in 1999. Cost of goods expense was about 72% less than in 1999. The drop is due in large part to the late production/delivery (and absence of the accrued expense) for *TUGboat*. (1999 cost of goods was also higher than normal.) TUG bank accounts grew from $90,324 at year end 1999 to $125,100 at year end 2000, a 28% increase.

If *TUGboat* expenses for 2000 were included below it would have reduced the bank account and bottom line by approximately $25K.

As always, the accounts have been reviewed by TUG's accountant but they have not been audited or verified. We are working toward a more consistent method of record-keeping so that the comparison from year to year provides a more useful analysis.

◇ Donald DeLand
Integre Technical Publishing Co.
4015 Carlisle NE, Suite A
Albuquerque, NM 87107
`don.deland@tug.org`

Dec 31, 2000

ASSETS	
Current Assets	
Checking/Savings	
BofA CD 21204-07032	$ 36,181.91
BofA Checking 21203-10859	90.07
BofA MMkt Bursry 21204-11698	46.51
BofA Maximizer 21203-18374	88,766.90
Petty Cash	15.41
Total Checking/Savings	125,100.80
Accounts Receivable	1,519.15
Total Other Current Assets	1,010.00
Fixed Assets	
Equipment	42,211.33
Accumulated Depreciation	− 31,515.42
Total Fixed Assets	10,695.91
TOTAL ASSETS	**$ 138,325.86**
LIABILITIES & EQUITY	
Liabilities	
AMS Prepaid Memberships	$ 1,875.00
Total Payroll Liabilities	1,467.40
Total Liabilities	3,342.40
Equity	
Perm Restricted – LaTeX	1,467.50
Perm Restricted – Bursary	46.51
Unrestricted	79,786.06
Chg in Restrict Acct 2000	1,744.81
Net Income	51,938.58
Total Equity	134,983.46
TOTAL LIABILITIES & EQUITY	**$ 138,325.86**

Profit & Loss
Jan–Dec 2000

Income	
Membership Dues	$ 132,470.00
Advertising	100.00
Conference	0.00
Product Sales	1,669.00
Contributions/Interest	11,462.00
Merchant charges and bank fees	− 3,816.00
Total Income	**141,885.00**
Cost of Goods Sold	
TUGboat	6,615.00
CD ROMs	9,791.00
Conference	2,100.00
Delivery	3,068.00
Miscellaneous	634.00
Total Cost of Goods Sold	**20,563.00**
Fund Disbursements	
Bursary Fund	1,390.00
LaTeX3	455.00
Total Fund Disbursements	**3,490.00**
Expense	
Gross Wages	42,241.00
Equipment	502.00
Operations	17,593.00
Other	5,558.00
Total Expense	**65,894.00**
Net Income	**$ 51,938.00**

2003 TeX Users Group Election

Arthur Ogawa
for the Elections Committee

The positions of TUG President and of 11 members of the Board of Directors will be open as of the 2003 Annual Board Meeting, which will take place in conjunction with the 24th Annual TUG Meeting to be held in July 2003 at the Outrigger Waikoloa Beach Resort in Hawaiʻi.

The current President, Mimi Jett, has stated that she will not stand for re-election. The terms of the eleven directors whose terms will expire in 2003 are Barbara Beeton, Karl Berry, Kaja Christiansen, Don DeLand, Susan DeMeritt, Stephanie Hogue, Judy Johnson, Ross Moore, Cheryl Ponchin, Kristoffer Rose, and Philip Taylor. Continuing directors, with terms ending in 2005, are Wendy McKay, Arthur Ogawa, Patricia Monohon and Michael Sofka.

The election to choose the new President and Board members will be held in Spring of 2003. Nominations for these openings are now being invited.

The Bylaws provide that "Any member may be nominated for election to the office of TUG President/to the Board by submitting a nomination petition in accordance with the TUG Election Procedures. Election ... shall be by written mail ballot of the entire membership, carried out in accordance with those same Procedures." The term of President is two years.

The name of any member may be placed in nomination for election to one of the open offices by submission of a petition, signed by two other members in good standing, to the TUG office at least two weeks (14 days) prior to the mailing of ballots. (A candidate's membership dues for 2003 will be expected to be paid by the nomination deadline.) The term of a member of the TUG Board is four years.

A nomination form follows this announcement; forms may also be obtained from the TUG office, or via the TUG Web pages at `http://www.tug.org`.

Along with a nomination form, each candidate is asked to supply a passport-size photograph, a short biography, and a statement of intent to be included with the ballot; the biography and statement of intent together may not exceed 400 words. The deadline for receipt at the TUG office of nomination forms and ballot information is **1 February 2003**.

Ballots will be mailed to all members within 30 days after the close of nominations. Marked ballots must be returned no more than six (6) weeks following the mailing; the exact dates will be noted on the ballots.

Ballots will be counted by a disinterested party not part of the TUG organization. The results of the election should be available by early June, and will be announced in a future issue of *TUGboat* as well as through various TeX-related electronic lists.

2003 TUG Election — Nomination Form

Only TUG members whose dues have been paid for 2003 will be eligible to participate in the election. The signatures of two (2) members in good standing at the time they sign the nomination form are required in addition to that of the nominee. **Type or print** names clearly, using the name by which you are known to TUG. Names that cannot be identified from the TUG membership records will not be accepted as valid.

The undersigned TUG members propose the nomination of:

Name of Nominee: ____________________

Signature: ____________________

Date: ____________________

for the position of (check one):

☐ **TUG President**

☐ **Member of the TUG Board of Directors**

for a term beginning with the 2003 Annual Meeting, **July 2003**.

Members supporting this nomination:

1. ____________________
(please print)

____________________ ____________
(signature) (date)

2. ____________________
(please print)

____________________ ____________
(signature) (date)

Return this nomination form to the TUG office (FAXed forms will be accepted). Nomination forms and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received in the TUG office no later than **1 February 2003**.[1] It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will incomplete applications be accepted.

☐ nomination form

☐ photograph

☐ biography/personal statement

TeX Users Group **FAX:** +1 503 223-3960
Nominations for 2003 Election
1466 NW Naito Parkway, Suite 3141
Portland, OR 97209-2820
U.S.A.

[1] Supplementary material may be sent separately from the form, and supporting signatures need not all appear on one form.

Institutional Members

American Mathematical Society, *Providence, Rhode Island*

Center for Computing Science, *Bowie, Maryland*

Cessna Aircraft Company, *Wichita, Kansas*

The Clarinda Company, *Clarinda, Iowa*

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 Advanced Study, *Princeton, New Jersey*

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

Iowa State University, Computation Center, *Ames, Iowa*

Kluwer Academic Publishers, *Dordrecht, The Netherlands*

KTH Royal Institute of Technology, *Stockholm, Sweden*

Marquette University, Department of Mathematics, Statistics and Computer Science, *Milwaukee, Wisconsin*

Masaryk University, Faculty of Informatics, *Brno, Czechoslovakia*

Max Planck Institut für Mathematik, *Bonn, Germany*

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

Princeton University, Department of Mathematics, *Princeton, New Jersey*

Springer-Verlag Heidelberg, *Heidelberg, Germany*

Stanford Linear Accelerator Center (SLAC), *Stanford, California*

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

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

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

University of Delaware, Computing and Network Services, *Newark, Delaware*

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

Università degli Studi di Trieste, *Trieste, Italy*

Vanderbilt University, *Nashville, Tennessee*

TEX USERS GROUP

Promoting the use of TEX throughout the world

mailing address:
P.O. Box 2311
Portland, OR 97208–2311 USA

shipping address:
1466 NW Naito Parkway,
Suite 3141
Portland, OR 97209–2820 USA

Phone:	+1 503 223–9994
Fax:	+1 503 223–3960
Email:	`office@tug.org`
WWW:	`www.tug.org`

2002 TUG Membership Form

Rates for TUG membership and TUGboat subscription are listed below. Please check the appropriate boxes and mail payment (in US dollars, drawn on a United States bank) along with a copy of this form. If paying by credit card, you may fax the completed form to the number at left.

• 2002 TUGboat (Volume 23)

• 2002 CD-ROMs include TEX Live 7 (1 disk) and Dante's CTAN 2002 (3 disk set).

• *Multi-year orders:* You may use this year's rate to pay for more than one year of membership.

• Orders received after 30 April, 2002: please add $10 to cover the additional expense of shipping back numbers of TUGboat and CD-ROMs.

		Rate	Amount
Annual membership for 2002 (TUGboat and CD-ROMs)	☐	$65	_____
Student/Senior membership for 2002 (TUGboat, CD-ROMs)*	☐	$35	_____
Subscription for 2002 (TUGboat and CD-ROMs) (Non-voting)	☐	$75	_____
Shipping charge if after 30 April, 2002	☐	$10	_____
Materials for 2001‡			
TUGboat Volume 22	☐	$45	_____
TEX Live 6 CD-ROM	☐	$5	_____
2001 CTAN CD-ROMs	☐	$10	_____
Voluntary donations			
General TUG contribution	☐		_____
Contribution to Bursary Fund†	☐		_____
		Total $	_____

Payment (check one) ☐ Payment enclosed ☐ Charge Visa/Mastercard/AmEx

Account Number: ______________________________

Exp. date: ________ Signature: ______________________________

* Please attach photocopy of (if student) 2002 student ID or (if senior) ID showing age 65 years or older.

† The Bursary Fund provides financial assistance for attendance at the TUG Annual Meeting.

‡ If you were not a TUG member in 2001 and wish to receive TEX Live and CTAN CDs right away, please order the desired item(s) along with your 2002 membership.

Information for TUG membership list

TUG uses the information you provide to mail you products, publications, notices, and (for voting members) official ballots, or in a printed or electronic membership list, available to TUG members only.

Note: TUG neither sells its membership list nor provides it to anyone outside of its own membership.

Allowing TUG to send you notices electronically will generally ensure that you receive them much earlier than the notice in printed form. However, if you would rather not receive TUG notices via electronic mail, please check the appropriate box.

Do not send me TUG notices via email ☐.

TUG plans to prepare a printed or electronic membership list, available to TUG members only. If you would like a listing in such a publication, please check the appropriate box.

Please do include my information in a published members-only TUG directory ☐.

Name: ______________________________

Department: ______________________________

Institution: ______________________________

Address: ______________________________

____

____

Phone: ____________________ Fax: ____________________

Email address: ______________________________

Position: ____________________ Affiliation: ____________________

TEX Consulting & Production Services

Information about these services can be obtained from:

TEX Users Group
1466 NW Naito Parkway, Suite 3141
Portland, OR 97209-2820, U.S.A.
Phone: +1 503 223-9994

Fax: +1 503 223-3960
Email: `office@tug.org`
URL: `http://www.tug.org/consultants.html`

North America

Loew, Elizabeth
President, TEXniques, Inc.,
675 Massachusetts Avenue, 6th Floor,
Cambridge, MA 02139;
(617) 876-2333; Fax: (781) 344-8158
Email: `loew@texniques.com`
Complete book and journal production in the areas of mathematics, physics, engineering, and biology. Services include copyediting, layout, art sizing, preparation of electronic figures; we keyboard from raw manuscript or tweak TEX files.

Ogawa, Arthur
40453 Cherokee Oaks Drive,
Three Rivers, CA 93271-9743;
(209) 561-4585
Email: `Ogawa@teleport.com`
Bookbuilding services, including design, copyedit, art, and composition; color is my speciality. Custom TEX macros and LATEX2ε document classes and packages. Instruction, support, and consultation for workgroups and authors. Application development in LATEX, TEX, SGML, PostScript, Java, and ßC++. Database and corporate publishing. Extensive references.

Veytsman, Boris
2239 Double Eagle Ct.
Reston, VA 20191;
(703) 860-0013
Email: `boris@lk.net`
I provide training, consulting, software design and implementation for Unix, Perl, SQL, TEX, and LATEX. I have authored several popular packages for LATEX and `latelx2html`. I have contributed to several web-based projects for generating and typesetting reports.
For more information please visit my web page: `http://users.lk.net/ borisv`.

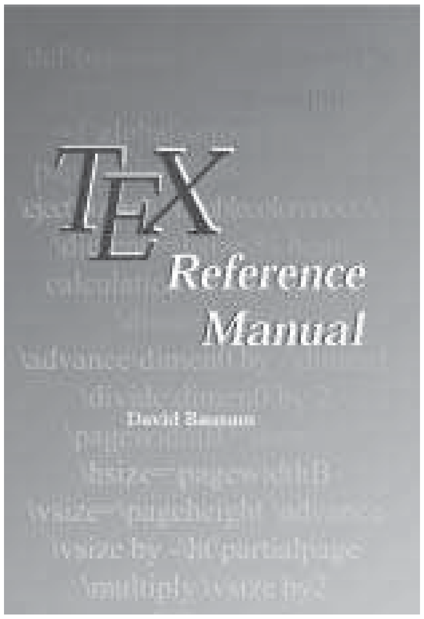
The Unicorn Collaborative, Inc, Ted Zajdel
115 Aspen Drive, Suite K
Pacheco, CA 94553
(925) 689-7442
Email: `contact@unicorn-collab.com`
We are a technical documentation company, initiated in 1990, which time, strives for error free, seamless documentation, delivered on time, and within budget. We provide high quality documentation services such as document design, graphic design and copy editing. We have extensive experience using tools such as FrameMaker, TEX, LATEX, Word, Acrobat, and many graphics programs. One of our specialties is producing technical manuals and books using LATEX and TEX. Our experienced staff can be trained to use any tool required to meet your needs. We can help you develop, rewrite, or simply copy-edit your documentation. Our broad experience with different industries allows us to handle many types of documentation including, but not limited to, software and hardware systems, communications, scientific instrumentation, engineering, physics, astronomy, chemistry, pharmaceuticals, biotechnology, semiconductor technology, manufacturing and control systems. For more information see our web page `http://www.unicorn-collab.com`.

Outside North America

DocuTEXing: TEX Typesetting Facility
43 Ibn Kotaiba Street,
Nasr City, Cairo 11471, Egypt
+20 2 4034178; Fax: +20 2 4034178
Email: `main-office@DocuTeXing.com`
DocuTEXing provides high-quality TEX and LATEX typesetting services to authors, editors, and publishers. Our services extend from simple typesetting and technical illustrations to full production of electronic journals. For more information, samples, and references, please visit our web site: `http://www.DocuTeXing.com` or contact us by e-mail.

kluwer
the language of science

Just Published



TEX *Reference Manual*

David Bausum, *Lighthouse & Associates, Beloit, WI, USA*

The *TeX Reference Manual* is the first comprehensive reference manual written by a programmer for programmers. It contains reference pages for each of TeX's 325 primitive control sequences. Over 80% of its reference pages contain examples that range from simple to challenging. Each example is typeset verbatim in a style which is easy to read and experiment with. *TeX Reference Manual* also just typesets the example, so you can see what it makes, and explains how the example works. The description on each primitive's reference page is an annotated discussion of *The TeXbook's* treatment of the primitive. That means a TeX user will find it natural to move back and forth between the two books. One of *TeX Reference Manual's* innovative features is families. They simplify the search for the primitive which performs a particular task.

Primitive Control Sequences

Family Name		Type	Description
Box (29)	Logic (20)	C	Command (163)
Character (16)	Macro (20)	D	Derived Command (17)
Debugging (25)	Marks (4)	IQ	Internal Quantity (42)
File I/O (13)	Math (69)	PI	Parameter (integer) (55)
Fonts (5)	Page (13)	PD	Parameter (dimen) (21)
Glue (12)	Paragraph (30)	PG	Parameter (glue) (15)
Hyphenation (11)	Penalties (12)	PM	Parameter (muglue) (3)
Inserts (8)	Registers (11)	PT	Parameter (token) (9)
Job (11)	Tables (9)		
Kern (7)			

TeX Reference Manual has appendices which provide a comprehensive discussion of: verbatim material, PostScript fonts, and two-column material. In particular, one word describes its font macros, elegant. The *TeX Reference Manual* is an invaluable tool for both the experienced and new users of TeX.

CONTENTS

February 2002 Hardbound, ISBN 0-7923-7673-0
390 pp. EUR 108.00 / USD 99.00 / GBP 68.00

ORDER TODAY!

ONLINE: WWW.WKAP.NL

Fax your order:
USA: 781-681-9045
Rest of World: +31 78 6546 474

Phone:
USA: +781-871-6600
Rest of World: +31 78 6392 392

Email:
USA: kluwer@wkap.com
Rest of World: services@wkap.nl