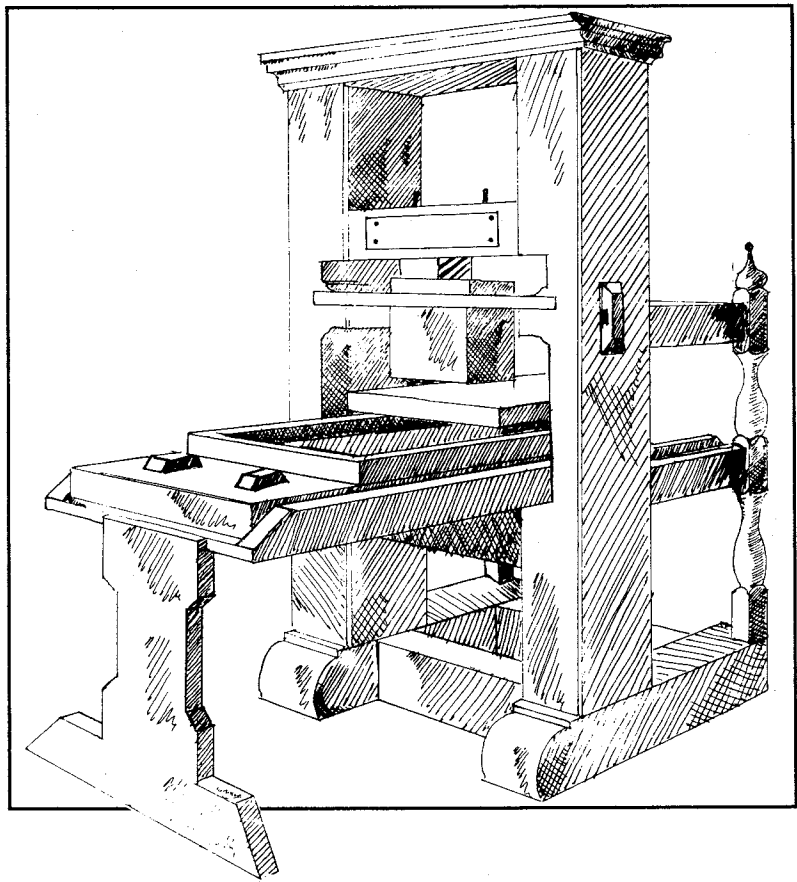


TUGBOAT

The Communications of the T_EX Users Group



Volume 15, Number 1, March 1994

TeX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published quarterly by the TeX Users Group, Balboa Building, Room 307, 735 State Street, Santa Barbara, CA 93101, U.S.A.

1994 dues for individual members are as follows:

- Ordinary members: \$60
- Students: \$30

Membership in the TeX Users Group is for the calendar year, and includes all issues of *TUGboat* and *TeX and TUG NEWS* 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 the annual election. A membership form is provided on page 77.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: North America \$60 a year; all other countries, ordinary delivery \$60, air mail delivery \$80.

Second-class postage paid at Santa Barbara, CA, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TeX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, 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.

TUGboat © Copyright 1994, TeX Users Group

Permission is granted to make and distribute verbatim copies of this publication or of individual items from this publication provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this publication or of individual items from this publication under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this publication or of individual items from this publication into another language, under the above conditions for modified versions, except that this permission notice may be included in translations approved by the TeX Users Group instead of in the original English.

Some individual authors may wish to retain traditional copyright rights to their own articles. Such articles can be identified by the presence of a copyright notice thereon.

Printed in U.S.A.

Board of Directors

Donald Knuth, *Grand Wizard of TeX-arcana*[†]
Christina Thiele, *President**
Michel Goossens*, *Vice President*
George Greenwade*, *Treasurer*
Peter Flynn*, *Secretary*
Barbara Beeton
Mimi Burbank
Alain Cousquer, *Special Director for GUTenberg*
Jackie Damrau
Luzia Dietsche
Michael Doob
Michael Ferguson
Roswitha Graham, *Special Director for the Nordic countries*
Yannis Haralambous
Kees van der Laan, *Special Director for NTG*
Joachim Lammarsch, *Special Director for DANTE*
Nico Poppelier
Jon Radel
Sebastian Rahtz
Tom Rokicki
Chris Rowley, *Special Director for UKTeXUG*
Raymond Goucher, *Founding Executive Director*[†]
Hermann Zapf, *Wizard of Fonts*[†]

* member of executive committee

[†] honorary

Addresses

All correspondence,
payments, etc.

TeX Users Group
P. O. Box 869
Santa Barbara,
CA 93102-0869 USA

Parcel post,
delivery services:

TeX Users Group
Balboa Building
Room 307
735 State Street
Santa Barbara, CA 93101
USA

Telephone

805-963-1338

Fax

805-963-8358

Electronic Mail

(Internet)

General correspondence:

TUG@tug.org

Submissions to *TUGboat*:

TUGboat@Math.AMS.org

TeX is a trademark of the American Mathematical Society.

In this pursuit of quantity, with which we seem obsessed, what place remains for what we used to call typographic quality? Is it still there to ensure that the author's thoughts are well transmitted, or has it become a useless luxury belonging to the past?

Ladislav Mandel

Developing an awareness of
typographic letterforms,
in *Electronic Publishing —
Origination, Dissemination and
Design* (Volume 6, Number 1,
March 1993)

TUGBOAT

COMMUNICATIONS OF THE T_EX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 15, NUMBER 1

SANTA BARBARA

CALIFORNIA

MARCH 1994

U.S.A.

TUGboat

During 1994, the communications of the TeX Users Group will be published in four issues. One issue (Vol. 15, No. 3) will contain the Proceedings of the 1994 TUG Annual Meeting.

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

The next regular issue will be Vol. 15, No. 2; deadlines for that issue will have passed by the time this issue is mailed. Deadlines for Vol. 15, No. 4 are August 17, 1994, for technical items, and September 14, 1994, for reports and similar items. Mailing dates for these two issues are scheduled for June and December. Deadlines for future issues are listed in the Calendar, page 66.

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, Barbara Beeton (see address on p. 3).

Contributions in electronic form are encouraged, via electronic mail, on magnetic tape or diskette, or transferred directly to the American Mathematical Society's computer; contributions in the form of camera copy are also accepted. The *TUGboat* "style files", for use with either plain TeX or LaTeX, are available "on all good archives". For authors who have no access to a network, they will be sent on request; please specify which is preferred. For instructions, write or call the TUG office.

An address has been set up on the AMS computer for receipt of contributions sent via electronic mail: TUGboat@math.ams.org on the Internet.

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@math.ams.org or to the Editor, Barbara Beeton (see address on p. 3).

TUGboat Editorial Board

Barbara Beeton, *Editor*
Victor Eijkhout, *Associate Editor, Macros*
Jackie Damrau, *Associate Editor, LaTeX*
Alan Hoenig, *Associate Editor, Typesetting on Personal Computers*

See page 3 for addresses.

Other TUG Publications

TUG publishes the series *TeXniques*, in which have appeared reference materials and user manuals for macro packages and TeX-related software, as well as the Proceedings of the 1987 and 1988 Annual Meetings. Other publications on TeXnical subjects also appear from time to time.

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 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.

APS $\mu 5$ is a trademark of Autologic, Inc.

DOS and MS/DOS are trademarks of MicroSoft Corporation

METAFONT is a trademark of Addison-Wesley Inc.
PC TeX is a registered trademark of Personal TeX, Inc.

PostScript is a trademark of Adobe Systems, Inc.
TeX and AMS-TeX are trademarks of the American Mathematical Society.

Textures is a trademark of Blue Sky Research.

UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

Addresses

Note: Unless otherwise specified, network addresses (shown in typewriter font) are on the Internet.

T_EX Users Group Office
P. O. Box 869
Santa Barbara, CA 93102-0869 U.S.A.
or

Balboa Building, Room 307
735 State Street
Santa Barbara, CA 93101 U.S.A.
805-963-1338
Fax: 805-963-8358
tug@tug.org

Donald Arseneau
Triumf
4004 Wesbrook Mall
Vancouver, BC Canada V6T 1Z6
asnd@erich.triumf.ca

Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
I-10129 Turin, Italy
beccari@polito.it

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

John Berlin
T_EX Users Group
P. O. Box 869
Santa Barbara, CA 93102-0869 U.S.A.
805-963-1338
john@tug.org

Mimi R. Burbank
Supercomputer Computations
Research Institute
B-186, 400 Science Center Library
Florida State University
Tallahassee, FL 32306-4052 U.S.A.
904-644-2440
mimi@scri.fsu.edu

Raymond Chen
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399 U.S.A.
raymond@microsoft.com

Alain Cousquer
LIFL – Université de Lille I
F-59655 Villeneuve d'Ascq Cedex
France
(33) 20 43 47 15
Fax: (33) 20 43 65 66
cousquer@LIFL.FR

Jackie Damrau
P. O. Box 875
Red Oak, TX 75154-0875, U.S.A.
214-617-2323
damrau@amber.unm.edu

Christine Detig
Technical University of Darmstadt
WG Systems Programming
Alexanderstraße 10
D-64283 Darmstadt, Germany
detig@iti.informatik.
th-darmstadt.de

Luzia Dietsche
Universität Heidelberg
Im Neuenheimer Feld 293
D-69120 Heidelberg, Germany
x68@vm.urz.uni-heidelberg.de

Michael Doob
Department of Math & Astronomy
University of Manitoba
341 Machray Hall
Winnipeg R3T 2N2
Manitoba, Canada
204-474-9796
mdoob@ccu.umanitoba.ca

Victor Eijkhout
Department of Computer Science
107 Ayres Hall
University of Tennessee at Knoxville
Knoxville, TN 37996-1301, U.S.A.
eijkhout@cs.utk.edu

Michael J. Ferguson
INRS – Télécommunications
Université du Québec
16 Place du Commerce
Verdun H3E 1H6, Québec Canada
514-765-7834
mike@inrs-telecom.quebec.ca

Jonathan Fine
203 Coldhams Lane
Cambridge CB1 3HY, England
J.Fine@pmms.cam.ac.uk

Peter Flynn
Computer Center
University College
Cork, Ireland
+353 21 276871 x2609
cbts8001@iruccvax.ucc.ie

Robert Fuster
Universitat Politècnica de València
Departament de Matemàtica Aplicada
Carni de Vera, 14
E-46071 València, Spain
mat5rfc@ccci.upv.es

Michel Goossens
Text Processing Section
MI Group – AS Division
CERN
CH-1211 Geneva 23, Switzerland
goossens@cern.ch

Roswitha Graham
K.T.H. Royal Institute of Technology
DAB
100 44 Stockholm, Sweden
46 (08) 7906525
uucp: roswitha@admin.kth.se

George D. Greenwade
Department of Economics and
Business Analysis
College of Business Administration
P. O. Box 2118
Sam Houston State University
Huntsville, TX 77341-2118 U.S.A.
bed_gdg@shsu.edu

Yannis Haralambous
187, rue Nationale
59000 Lille, France
Fax: (33) 20.91.05.64
yannis@gat.univ-lille1.fr

Matthew N. Hendryx
P. O. Box 218
North Manchester, IN 46962 U.S.A.

Alan Hoenig
17 Bay Avenue
Huntington, NY 11743 U.S.A.
516-385-0736
ajhjj@cunyv.cuny.edu

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

Kees van der Laan
Hunzeweg 57
9893 PB Garnwerd (Gr.)
The Netherlands
cgl@rug.nl

Joachim Lammarsch
Computing Center
University of Heidelberg
Im Neuenheimer Feld 293
D-69120 Heidelberg, Germany
x92@vm.urz.uni-heidelberg.de

Hans van der Meer
 University of Amsterdam
 Faculty of Mathematics and
 Computer Science
 Plantage Muidergracht 24
 1018 TV Amsterdam
 Netherlands
 hansm@fwi.uva.nl

Pat Monohon
 T_EX Users Group
 P. O. Box 869
 Santa Barbara, CA 93102-0869 U.S.A.
 805-963-1338
 monohon@tug.org

T. L. (Frank) Pappas
 338 Francis Drive
 Havertown, PA 19083
 fpappas@mcimail.com

Nico A. F. M. Poppelier
 Elsevier Science Publishers BV
 Academic Publishing Division
 Information Technology Development
 Molenwerf 1
 1014 AG Amsterdam,
 The Netherlands
 n.poppelier@elsevier.nl

Jon Radel
 P. O. Box 2276
 Reston, VA 22090-0276 U.S.A.
 jon@radel.com

Sebastian Rahtz
 ArchaeoInformatica
 12 Cynet Street
 York YO2 1AG, U.K.
 spqr@ftp.tex.ac.uk

David Rhead
 University of Nottingham
 Cripps Computing Centre
 University Park
 Nottingham NG7 2RD England, U.K.
 David_Rhead@vme.ccc.nottingham.ac.uk

Tomas Rokicki
 Box 2081
 Stanford, CA 94309 U.S.A.
 415-322-6442
 rokicki@cs.stanford.edu

Chris Rowley
 Open University
 Walton Hall
 Milton Keynes MK7 6AA
 United Kingdom
 c.a.rowley@vax.acs.open.ac.uk

David Salomon
 Computer Science Department
 California State University
 Northridge, CA 91330-8281 U.S.A.
 dxs@ms.secs.csun.edu

Merry Obrecht Sawdey
 3532 Bryant Ave So.
 Apt. 112
 Minneapolis, MN 55408 U.S.A.
 sawdey@denali.ee.umn.edu

Joachim Schrod
 Technical University of Darmstadt
 WG Systems Programming
 Alexanderstraße 10
 D-64283 Darmstadt, Germany
 schrod@iti.informatik.
 th-darmstadt.de

Janet Sullivan
 T_EX Users Group
 P. O. Box 869
 Santa Barbara, CA 93102-0869 U.S.A.
 805-963-1338
 janet@tug.org

Christina Thiele
 5 Homestead Street
 Nepean K2E 7N9, Ontario Canada
 cthiele@ccs.carleton.ca

Gabriel Valiente Feruglio
 Universitat de les Illes Balears
 Departament de Ciències
 Matemàtiques i Informàtica
 E-07071 Palma de Mallorca, Spain
 valiente@ipc4.uib.es

Hermann Zapf
 Seitersweg 35
 D-64287 Darmstadt, Germany

General Delivery

Opening words

Christina Thiele
President, T_EX Users Group

Welcome to 1994, our 15th anniversary year. Nice to see all of you who have renewed your TUG memberships; and it's also nice to welcome new members who are seeing what TUG and *TUGboat* are all about. Hope you enjoy the issue!

1 Board news

Some changes in the board should be noted here — they were too late for inclusion in *TTN* 3,1.

For last fall's board election, as reported in *TTN* 2,4:21, there were only 11 valid nominations for the 15 positions available on the board. By acclamation, these eleven people were named to the board, and it became my task to fill the remaining four positions. I'd like to welcome three new members to our board: Michael Doob, Michel Goossens, and Tom Rokicki have graciously accepted to join TUG's board for a term of one and a half years.

The board then had to select two new officers, Vice President and Treasurer: these are Michel Goossens and George Greenwade, respectively. Peter Flynn will continue on as Secretary.

As well, Chris Rowley takes over from Peter Abbott as chair of the UK T_EX Users Group, and hence as their representative on TUG's board.

2 The annual meeting

The annual meeting in Santa Barbara promises to be as big as any we've had yet. The focus is on innovations, and colour certainly is a big one. A preliminary program has now been mailed to all members, and also appears later in this issue. Updates on the meeting will be appearing over the next several months, so keep an eye out.

Check out the program. Check out your funds. Start saving! And plan to come out to the west coast to find out about some very interesting T_EX applications. Come to renew old acquaintances, and to meet new members. Not to mention the fact that you'll also find some sun, some sand, some ocean, and some very mild temperatures. At this time of year, it all sounds just too hard to resist!

3 T_EX in new places

In my last column (*TUGboat* 14, no. 4, p. 371) I mentioned the Linguistic Society of America's an-

nual meeting in Boston, for January 1994, where I'd be doing a poster session on T_EX and linguistics. Just before the meeting, in late December, I also started up a new discussion list, *ling-tex*, and the response was tremendous. Much of what was posted to the list ended up being reported and shown at the LSA meeting: macros, fonts, archives, information on user groups around the world, and so on. There were over 1,000 people in attendance for the 4-day meeting, in spite of a severe snowstorm that delayed travel for hours. I must confess to having been surprised at the wide-spread familiarity with T_EX in general, even if its application to linguistics was not as well-known. Many of us who had met via the *ling-tex* list were delighted to meet one another in person.

I was fortunate in having considerable assistance from Ed Baker of EBTS, who helped make the two-hour poster session very informative for everyone who stopped by. If anyone is considering presenting such a demonstration session, you really do need two people to do it properly!

As well, Robert Harris of MicroSystems Inc., a long-time member of TUG and an active vendor at our own meetings, had a booth in the massive book-display hall. Here again, there was a lot of interest shown during the meeting, as people stopped to ask about T_EX, about books and software, about user groups and archives, and just to find out more about how T_EX might be a useful means of preparing linguistics documentation.

In all, the exposure which T_EX received at the LSA meeting via the poster session and the vendor's booth was tremendous. I'd like to thank Ed and Bob for their contributions to the effort. Next year's meeting, in New Orleans, will hopefully again see a display or demonstration of T_EX and linguistics, with perhaps some of the *ling-tex* readers participating. Hint, hint!

And now to look forward ... coming up very shortly (April 11–15) is the massive 4-way joint conference of RIDT94, EP94, TEP94, and PODP94 in Darmstadt. If any of you are going to attend, please consider writing up a short report for us: get in touch with either Barbara or myself.

A little later on, the annual Society for Scholarly Publishing (SSP) meeting is in San Francisco this year (June 8–10). If there's anyone in the San Francisco area who is planning to attend this meeting on scholarly publishing, or who would be willing to lend a hand in case we get a TUG booth there, please get in touch with me.

And if you know of any conferences where a T_EX or TUG presence would be useful, do let us know.

4 Announcing Euro \TeX '94

This September, on an island off the coast of Gdansk in Poland, the 1994 Euro \TeX meeting will be held, hosted by the Polish \TeX Users Group, GUST.¹ Abstracts should be sent by May 1st, final papers by August 15; the meeting will take place September 26–30. Information appears elsewhere in this issue of *TUGboat* with the details.

5 Office updates

By this time you should have received the first announcements regarding the annual meeting in Santa Barbara. Included in that packet was — **your membership ID number!** This will be your number for as long as you're a TUG member. Please have it handy when you call the office for technical support, and of course include it whenever you write to us, or when you order something from the product store — it guarantees your 10% discount as a TUG member.

The office is also pleased to announce that the following new publications are now available:

- *The L \TeX E Companion*, by Goossens, Mittelbach and Samarin
- *\TeX in Practice*, by Stephan von Bechtolsheim
- *\TeX and L \TeX E: Drawing and Literate Programming*, by Eitan Gurari
- *A User's Guide for \TeX Help; The On-Line \TeX Handbook*, by Borde and Rokicki

They expect also to have Norm Walsh's new book, *Making \TeX Work*, very shortly. If you plan on coming to the meeting this summer, better bring an extra strong book bag! Which reminds me ... I'm also going to be looking for a replacement mug for the one I broke last year. Pat says there are lots of mugs from past meetings — check your collection to see if you need any new ones too!

◊ Christina Thiele
5 Homestead Street
Nepean, Ontario
K2E 7N9 Canada
cthiele@ccs.carleton.ca

¹ Note that the last Euro \TeX meeting was held in 1992 in Prague, reported in TTN 1,3:22–23.

Editorial Comments

Barbara Beeton

Welcome to a new year of TUG and *TUGboat*. With the end of the old year, there have been some changes in faces on the TUG Board of Directors. Christina has already mentioned this in her “Opening words”. I'd just like to add that the Board members are your representatives, and need your input to do their job properly. All the members of the Board are listed on the inside front cover of every issue of *TUGboat*, and their addresses appear in the address list at the beginning of every regular issue (not the annual Proceedings). Make yourselves known; let us know about your concerns for TUG.

1 *TUGboat* wish list

The “wish list” published in *TUGboat* 14, no. 4, p. 372, has yielded some additional suggestions and some new volunteers, but by no means as many as one might have hoped. Perhaps that's because it arrived around the end of the year, always a busy time.

Okay, readers! You've had enough time to think about this — now's the time to act! Go back and read the list again, and if you have any comments or suggestions, no matter how grand or trivial, send them in!

One suggestion that is being investigated is to have occasional “theme” issues, edited by a volunteer with recognized experience in that area. Suggestions for topics and editors are welcome — send them to me in care of TUGboat@math.ams.org, or to the postal address that appears at the end of this column.

2 Another award for DEK

I received the following note from Don Knuth on February 24:

Yesterday's mail brought the good news that the Royal Swedish Academy of Sciences decided to award me the Adelsköld Medal for my work on *The Art of Computer Programming* and \TeX . This medal is presented only once every 10 years, for “an innovation within technical sciences.”

They say I will receive the medal “from the hands of H. M. King Carl XVI Gustaf” at the academy's annual meeting on March 24. (He is the pleasant gentleman who can be seen on TV these days escorting Sweden's magnificent Queen at the Olympics.)

As with previous awards, this is certainly well deserved. Congratulations, Don!

3 New magazine on type and typography

Don Hosek, an occasional contributor to this publication, has announced his own entry into the field of typographic periodicals.

Serif, a new quarterly publication on type and typography, will begin publication in the fourth quarter of 1994, and then continue regularly every third month beginning March 1995.

Authors are invited to write articles on all aspects of type and typography; submissions will be paid for at market rates, depending on length and content. Interested parties should request an author's guide by writing to the address below, and including their *postal* address.

Serif
 Quixote Digital Typography
 555 Guilford
 Claremont, CA 91711
 Fax: 909-625-1342
 E-mail: clement!dhosek@netcom.com

Inquiries are also invited from potential advertisers and subscribers.

4 Hyphenation and exceptions

The list of (U.S. English) hyphenation exceptions hasn't appeared for over a year, so it's time to start polishing it up again for publication in the fall. If you have encountered any words that \TeX doesn't hyphenate properly, check the last edition and send in anything new; see *TUGboat* 13, no. 4, p. 452, or retrieve it from a CTAN site, where it is filed as

.../digests/tugboat/tb0hyf.tex

Please remember that the authority for the U.S. patterns is *Webster's Third New International Dictionary*; you may not always agree with it (I don't), but it's the resource that has been adopted.

I'm also trying to collect information on who is maintaining similar lists for other sets of hyphenation patterns, for publication either in *TUGboat* or in a future resource directory. If you are maintaining such a collection, or know who is doing so, please send me the following information:

- the name and address (e-mail preferred) of the person maintaining the collection;
- the language for which the patterns are used;
- the creator of the patterns and the name of the patterns file.

I will forward this information to the Technical Working Groups concerned with language matters.

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

The \TeX Hierarchy

Donald Arseneau, Raymond Chen and
 Victor Eijkhout

Introduction

For the UNIX operating system, a list of characterizations exists describing what constitutes a novice, a user, a guru, ... Here we give a similar list for users of \TeX . The reader is kindly asked to take this purely in a humorous vein.

The name

Novice says 'tecks'.

User says 'tecchhh' but still moistens the screen doing it.

Programmer correctly pronounces ' \TeX '.

Wizard has made at least one bad pun on the name \TeX .

Guru knows that even Knuth says 'tek'.

The manual

Novice owns *The \TeX book*.

Programmer has just made a first correction to the text.

Hacker has formatted `texbook.tex` and knows about Knuth's 'little joke'.

Wizard is thinking of ways to supply the missing 'tactile and olfactory sensations' of \TeX .

Guru thinks ' \TeX : the program' is more useful.

The index of *The \TeX book*

Novice is confused by the number of references for each entry, has laughed at ' \TeX : bad puns on the name', and has counted the number of middle names of Barbara Beeton.

User knows about Bo Derek (in *The \TeX book*), Jill Knuth (in *The METAFONTbook*), and Ellen Gilkerson (in the \LaTeX manual).

Wizard knows why some entries are italicized or underlined.

Guru knows to look up Bourbaki for smart line breaks in paragraphs.

The system

Novice has found many bugs in \TeX .

User has learned that there are no bugs in \TeX , but doesn't understand why 'it doesn't work!'.

Guru has actually found bugs in \TeX ; frames the check from DEK.

Guru extraordinaire cashes checks from DEK.

Famous people

Novice is not sure whether Leslie Lamport is a man or a woman.

User knows not to capitalize ‘barbara beeton’.

Wizard knows how to pronounce ‘Knuth’ and ‘Eijkhout’.

Guru Knuth has asked about their middle name(s).

Programming style

Novice uses grouping without knowing why.

User writes `\bf{ ... }` and doesn’t understand what went wrong.

Programmer writes `\def\bold#1{{\bf #1}}`.

Programmer first class writes `\long\def\bold#1{{\bf #1}}`.

Hacker writes `\def\beginbold{\bgroup\bf}`
`\def\endbold{\egroup}`

Wizard writes `\def\bold{\bgroup\bf\let\next=}`.

Guru writes `\def\bold#1{\bgroup\bf\let\next=}`.

Style (cont’d)

Novice has heard of ties.

User inserts ties and writes ‘dr.\ ’.

Hacker writes ‘dr.\ ’, except in bibliographies where `frenchspacing` is in effect.

Guru makes ‘.’ an active character in bibliographies so that ‘D.E. Knuth’ means ‘D.\,E.\penalty 300\ Knuth’.

Errors

Pre-novice wonders why ‘Q’ takes so long to quit.

Novice will exit on the first ‘error’, even if the message starts with ‘OK’.

User keeps pressing return to scroll past errors, until that gets into an infinite loop.

Guru, having written the input file with ‘cat >’ in the first place, will type ‘i’ at an error, correcting all typos and supplying all missing macros interactively, thereby successfully completing the formatting in the first run.

Capacity Exceeded ...

Novice constantly runs into the ‘TeX capacity exceeded’ error and asks the admin to build a larger version.

User knows how to find unbalanced curly braces.

Hacker occasionally runs into ‘TeX capacity exceeded’ errors and usually finds a way around them.

Wizard knows how to increase TeX’s capacity, taking care to read DEK’s warnings about setting the values too high.

Guru ignores DEK’s warnings.

Printing and previewing

Novice prints the whole document after each run of TeX.

User knows of previewers.

Programmer knows at least two previewers and vigorously argues why one is utter garbage.

Wizard thinks that

```
\tracingoutput=1
\showboxdepth\maxdimen
\showboxbreadth\maxdimen
```

is the best previewer.

Macros

Novice has heard of macros, but has never seen one.

User writes macros that are used once, and that are longer than the code they replace.

Programmer, having been bitten by unwanted spaces, writes macros that don’t contain spaces, and every line ends with a ‘%’.

Hacker has written self-modifying macros, writes `\endlinechar=-1` or `\catcode‘\~M=9` to prevent having to put %’s at the end of lines in macros.

Guru has written macros containing #####, more than 3 `\expandafter`’s in a row, and the sequence `\expandafter\endcsname`.

Fossil still has macros written in TeX78.

Macros (cont’d)

Novice has written a macro `\box` to draw a box.

User has renamed it to `\boxit`.

Wizard has redefined `\mbox` so that it can have `\verb` in its arguments.

L^AT_EX

Novice uses L^AT_EX because colleagues and friends do.

User uses L^AT_EX, even though colleagues and friends use Microsoft Word or WordPerfect.

Wizard uses L^AT_EX for journal and conference submissions, but homegrown macros when working alone.

L^AT_EX errors

Novice actually takes the manual when it says ‘LaTeX error. See LaTeX manual for explanation.’

User knows what the relevant bits of L^AT_EX error messages are.

Programmer knows what to type at the question mark when L^AT_EX reports ‘\begin{document} ended by \end{itemize}’.

Wizard doesn't make errors in L^AT_EX, and answers questions about L^AT_EX by editing `latex.tex`.
Guru knows whether to edit `latex.tex`, `lplain.tex`, `article.sty`, or `art10.sty`.

L^AT_EX style

Novice types `a$_{1}$` because the error in `a_{1}` occurred on the `'_'`.

User types `a_{1}` because Leslie Lamport says so.

Other packages

Novice could do more in Pagemaker.

User doesn't see the difference between T_EX macros and WordPerfect macros.

Hacker writes macros to make T_EX look more like `troff`.

Wizard types `\input troff` to process old `troff` files.

Guru types

```
\input txtmacros
\input text.txt
to format plain text.
```

Life, everything

Novice thinks that learning T_EX will take a long time.

User realises that it wasn't so bad after all.

Programmer tries to convince himself that the next macro is really going to save time in the future.

Wizard daydreams idly about how much he could have done with his life if he had never heard of T_EX.

Guru realises that a life without T_EX is not worth living.

(Also thanks to Barbara Beeton, Tim Chow, Denys Duchier, Dan Ellard, Michael Sofka.)

- ◊ Donald Arseneau
Triumf
4004 Wesbrook Mall
Vancouver, BC
Canada V6T 1Z6
Internet: `asnd@erich.triumf.ca`
- ◊ Raymond Chen
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
Internet: `raymond@microsoft.com`
- ◊ Victor Eijkhout
Department of Computer Science
University of Tennessee at Knoxville
Knoxville, TN 37996-1301
Internet: `eijkhout@cs.utk.edu`

Philology

Some remarks on typesetting classical Latin

Claudio Beccari

Abstract

Besides requiring special fonts and/or hyphenation patterns, typesetting of ancient languages, in particular classical Latin, requires that some stylistic points should be taken into consideration; for instance, medieval codices and Renaissance books should *not* be taken as models, but, if an old style flavor is desired, books printed in the late XVII century should be imitated. Particular attention is given to the issue of the letters 'u' and 'v'.

1 Introduction

The excellent paper by Yannis Haralambous [1] on hyphenation of ancient Greek and Latin published in *TUGboat* 13.4 gives me the opportunity of expressing my ideas about the style of composition of ancient languages with particular reference to classical Latin. Having prepared the hyphenation patterns for modern Latin [4], I was very pleased to find Haralambous's work on ancient Latin and ancient Greek and to see how he solved the difficulty of preparing hyphenation tables that allow to deal with prefixes that are so common in both languages.

Haralambous cites a Latin example from [7], having taken into account the Chicago manual of style [8], where: a) upper-case 'V' and its corresponding lower-case 'u' are used; b) the ligature 'æ' (which implies also œ, Æ, Œ) is used. The Chicago manual of style, in practice, suggests to set Latin according to what the scholars call the *restituta* [*lectio*], that is in a way that supposedly imitates the original setting.

In this paper I will try to prove that the *restituta* in reality imitates the medieval codices and the first printings, not the original way of writing Latin by the Roman themselves, so that the *restituta* should be avoided in favor of a more modern way of setting classical Latin.

2 The Latin script

We are all aware that ancient Romans used 'V' for indicating several different sounds, one of which was the back closed vowel /u/, another was the closed bi-labial vowel /y/ (same as the Greek Υ from which the Roman glyph 'V' derives), but certainly also the voiced labio-dental fricative consonant /v/, especially when it was in intervocalic position (how

would they have pronounced the word *VVFLA* otherwise?), or at the beginning of a word when it was followed by a vocalic ‘V’ as in *VVLGVVS*, *VVLT*, ... The fact that the consonantic value of the letter ‘V’ is maintained consistently in all the Romance languages (with possible alterations into a bi-labial fricative or a bi-labial plosive) confirms this value. I came across the works of Quintilianus [14] where he complains about the poverty of the Latin alphabet (of his time) that does not allow to distinguish the three sounds represented by the same glyph; Fig. 1 shows a page of a XVI century book where his complaints are reported. Besides Quintilianus’ complaints, Fig. 1 gives an example of classical Latin typeset according to the habits of the early printings.

We are also aware of the fact that twenty centuries ago our Roman and Greek ancestors did not use lower-case letters; these are a medieval variation of the uncial script of either language; such variation was substantially complete in the eighth or ninth century, while the complex system of Greek diacritics (see the fonts produced by Silvio Levy [5] or by Mylonas and Whitney [6]) was complete around the seventh century. Also the punctuation varied a lot (that is, it was either completely absent or reduced to very simple marks) and it was settled down just during the Renaissance, in practice, with the advent of printing.

Fig. 2 shows a page of one of the last codices that was composed for the Duke Federico of Urbino [16]; the script is defined *calligrafia umanistica libraria o tonda* (book or round humanistic script) and is particularly easy to read.¹ The use of capitalization (“lucas” [Luke] and “dei” [God’s] in lower case, for example), abbreviations, ligatures, punctuation, accents, is very different from what we use today; ‘u’ is regularly used in lower case, except in one case where ‘v’ is used (... *env/merare longissimum est.*), and ‘V’ is used in small caps, especially after ‘Q’. Abbreviations such as ‘Q̇’ or ‘q̇’ for ‘qui’, or ‘ṗ’ for ‘prae’, make this text difficult to understand for readers not acquainted with paleography even if the lettering is very clear.

¹ A similar script defined *calligrafia umanistica diritta* (straight humanistic script) was used by the engravers working for Manunzio as a model for producing what now we call “roman type”; the *calligrafia umanistica inclinata o corsiva* (slanted or cursive humanistic script) was the model for designing what now we call “italics”. In Italian still nowadays these font shapes are called *tondo* and *corsivo* instead of “romano” and “italico” respectively.

When in the fifteenth century Gutenberg, Manunzio and the other prototypographers designed the glyphs for use in printing, they imitated the three current Latin handwritten styles (Texture, Roman and Italic), and these did not contain upper- and lower-case ‘V’ and ‘U’; in printing they preserved the manuscript tradition of using ‘V’ for the upper-case and ‘u’ for the lower case letter *independently from the language* in use. I have seen books in Latin, Italian, French, English, Spanish, German printed in the XV, XVI and XVII centuries, where this habit was preserved. Sometimes in the initial position a lower case ‘v’ was used independently of the consonantic or vocalic function of the letter, while in the 42-line Bible by Gutenberg (at least in the sample page reproduced in [10]) ‘u’ and ‘v’ are correctly used but only at the beginning of the words.

Sporadic attempts to eliminate this anomaly were made by many grammarians, for example Trissino for Italian [12], but they remained *vox clamans in deserto* till the second half of the XVIII century. Fig. 4 shows a couple of facing pages from a book by Trissino printed in 1547 [13], where he uses the phonetic alphabet he had proposed in [12] for the Italian language: it includes two glyphs for the two sounds of each of the letters ‘e’, ‘o’, ‘s’, and ‘z’, it uses ‘u’ and ‘v’ correctly even in capitalized titles, and uses ‘k’ instead of ‘ch’ (not always) and ‘lj’ instead of ‘gli’; there are no unusual abbreviations, the ligatures concern only the letter ‘s’ followed by another ‘s’ or by ‘t’ and the spelling is unusually modern, except perhaps for an excessive use of ‘h’ compared to modern usage.

According to my sources [11], it was the Dutch printer Elsevier that eventually succeeded in doing away with this confusion and used the proper letter for the proper sound; Fig. 5 shows a couple of pages of a book printed by Elsevier in 1649, where in the body of the text ‘u’ and ‘v’ are used according to the new style, while in the titles set in capitals or caps-and-small-caps the old style is preserved and the glyph ‘V’ is used throughout. In the XVIII century the new style of using ‘v’ and ‘u’ in the proper places had become almost universally accepted, so that you can recognize a two century old book from other elements (language style, font design, ligatures, page graphic layout, ...), not from the use of ‘u’ and ‘v’.

Before the age of printing the lower-case letter ‘i’ was dotless in the humanistic straight and cursive scripts (see Fig. 2); the dot was introduced with the

inducium est. Quo quidem ita seuerè sunt usi ueteres Grammatici, ut non uersus modò censoria quadam uirgula notare, & libros qui falsò uiderentur inscripti, tanquam subditios sumu mouere familia permisissent sibi: sed autores alios in ordinem redegerint, alios omnino exenerint numero. Nec Poetas legisse satis est. excutiendum omne scriptorum genus: non propter historiarum modò, sed uerba, quæ frequenter ius ab autoribus sumunt. Tum nec citra Musicen Grammaticæ potest esse perfecta, cum ei de metris rhythmisq; dicendum sit. Nec si rationem syderum ignoret, Poetas intelligat: qui (ut alia mittam) toties ortu occasuq; signorum in declarandis temporibus utuntur. Nec ignara Philosophiæ, cum propter plurimos in omnibus ferè carminibus locos ex intima quæstionum naturalium subtilitate repetitos: tum uel propter Empedoclem in Græcis, Varronem ac Lucretium in Latinis: qui præcepta sapientiæ uersibus tradiderunt. Eloquentia quoque non mediocri est opus, ut de unaquaque earum quas demonstrauimus rerum dicat proprie & copiose.

Quo minus sunt ferendi, qui hanc artem ut tenuem ac icinam cauillantur: quæ nisi Oratori futuro fundamenta fideliter iecerit, quicquid superstruxerit, corruet: necessaria pueris, iucunda senibus, dulcis secretorum: comes, & quæ uel sola omni studiorum genere plus habet operis quam ostentationis. Ne quis igitur tanquam parua fastidiat Grammaticæ elementa: non quia magnæ sit operæ, Consonantes à Vocalibus discernere, ipsasq; eas in Seniuocalium numerum, Mutarumq; partiri: sed quia interiora uelut sacri huius aduentibus, apparebit multa rerum subtilitas, quæ non modo acere ingenia puerilia, sed exercere altissimam quoque eruditionem ac scientiam possit. An cuiuslibet auris est exigere Literarum sonos? Non hercule magis quam neruorum. At Grammatici saltem omnes in hanc descendunt rerum tenuitatem, desint

De literis & earum potestate.

desint ne aliquæ nobis necessariae literæ, non cum Græca scribimus (tum enim ab iisdem duas mutuamur) sed proprie in x & z. Latinis, ut in his seruus & uulgus, Aeolicum digamma desideratur. Et medius est quidam, u, & i, literæ sonus. Non enim sic optimum dicimus, ut optimum. Et in here, neque, e, plane, neque, i, auditur. An rursus aliæ redundant, præter illam aspirationis I: quæ si necessaria est, etiam contrariam sibi poscit, I. Et K, quæ & ipsa quorundam nominum nota est. Et Q, cuius simul effectus species que, nisi quòd paulum à nostris obliquatur. Kappa apud Græcos, nunc tantum in numero manet. Et nostrarum ultima X, qua tamen carere posuimus, si non quæsissemus. Atque etiam in ipsis Vocalibus Grammatici est uidere, an aliquas pro Consonantibus usus acceperit, quia iam sicut tam scribitur, & uos ut eos. At quæ ut uocales iunguntur, aut unam longam faciunt, ut ueteres scripsere, qui geminatione earum uelut apicem utebantur: aut duas: nisi quis pucat etiam ex tribus uocalibus syllabam, quod nequit fieri, si non aliquæ officio consonantium fungantur.

Quæret etiam hoc, Quomodo duabus demum uocalibus in seipsas coeundi natura sit, cum consonantium coæt nulla, nisi alteram frangat. Atqui litera, i, sibi insidit. Coniicit enim est ab illo iacit. Et, u, quomodo nunc scribitur uulgus & seruus. Sciat etiam Cicroni placuisse Ajo Maijamq; gemi= In Oratore. nata, i, scribere. Quod si est, etiam iungetur ut consonans.

Quare discat puer, quid in literis proprium, quid commune, quæ cum quibus cognatio. Nec miretur cur ex scamno fiat Scabellum: aut à pinna (quod est acutum) securis utrinque habens aciem, Bipennis: ne illorum sequatur errorem, qui quia Bipennis, à pennis duabus hoc esse nomen existimant, Pinnas autem dici uolunt. Neque has modò nouerit mutationes, quas afferrunt declinatio, aut prepositio, ut secat secuit, cedit excidit, cedit cecidit, calcat excultat: & sic à lauando lotus, & inde

Figure 1: The grammarian Quintilianus (I century A.D.) discusses the pronunciation of Latin and complains that ‘V’ (‘u’ in print) is supposed to represent several sounds, one of which is the *Aeolicum digamma*. This page reports part of the section “De literis & earum potestate” (The letters and their value): [...] *aliquæ nobis necessariae literæ, non cum Græca scribimus, ([...]) sed proprie in Latinis, ut in his [“seruus & uulgus”], Aeolicum digamma desideratur. Et medius est quidam, u, & i, literæ sonus. Non enim sic [“optimum”] dicimus, ut [“optimum”].*

German scripts from which Texture is derived. Besides a number of ligatures², some of which survived till to day, there were two different glyphs for the letter ‘s’, one for the end of the words and one for internal or initial positions. The latter closely resembled an ‘f’, the difference being that the tie did not cross the stem of the letter (see again Fig. 2); the ligature of the latter glyph with a regular ‘s’ gave rise to the ‘ß’ glyph. Among these ligatures there are ‘æ’, ‘Æ’, ‘œ’, and ‘Œ’ that were totally unknown twenty centuries ago. Furthermore many shorthand notations, abbreviations, substitutions of ‘n’ and ‘m’ with a

² In a recent issue of *TTN* [17] Peter Flynn asks if “someone would like to try faking (*sic*) up the ct and st ligatures”; I like these ligatures that were so frequent in XVIII century books and I admit that sometimes such graphic devices are useful for giving “that particular flavor” to the printed page.

tilde accent on the preceding vowel (always Fig. 2), occasional accents on the desinence of the ablatives (even in printing), etc. etc., were such that a modern unskilled reader may find it difficult to read a XV or XVI century book.

These are the main reasons why I think medieval manuscripts and early printings should not be taken as a model for setting classical Latin into type. Nor should they be taken as a model *today* for setting into type the works of the medieval writers themselves; would you set *The Canterbury tales*, or *El cantar de mio Cid*, or *Le romans de la rose*, or *Il decameron* making use only of ‘V’ and ‘u’, and using all the other abbreviations, ligatures, unusual glyphs, diacritics, and the like? The only reason for doing so might be for reproducing those masterpieces with the look they had their days, but this would be useful only for scholars, and I doubt that

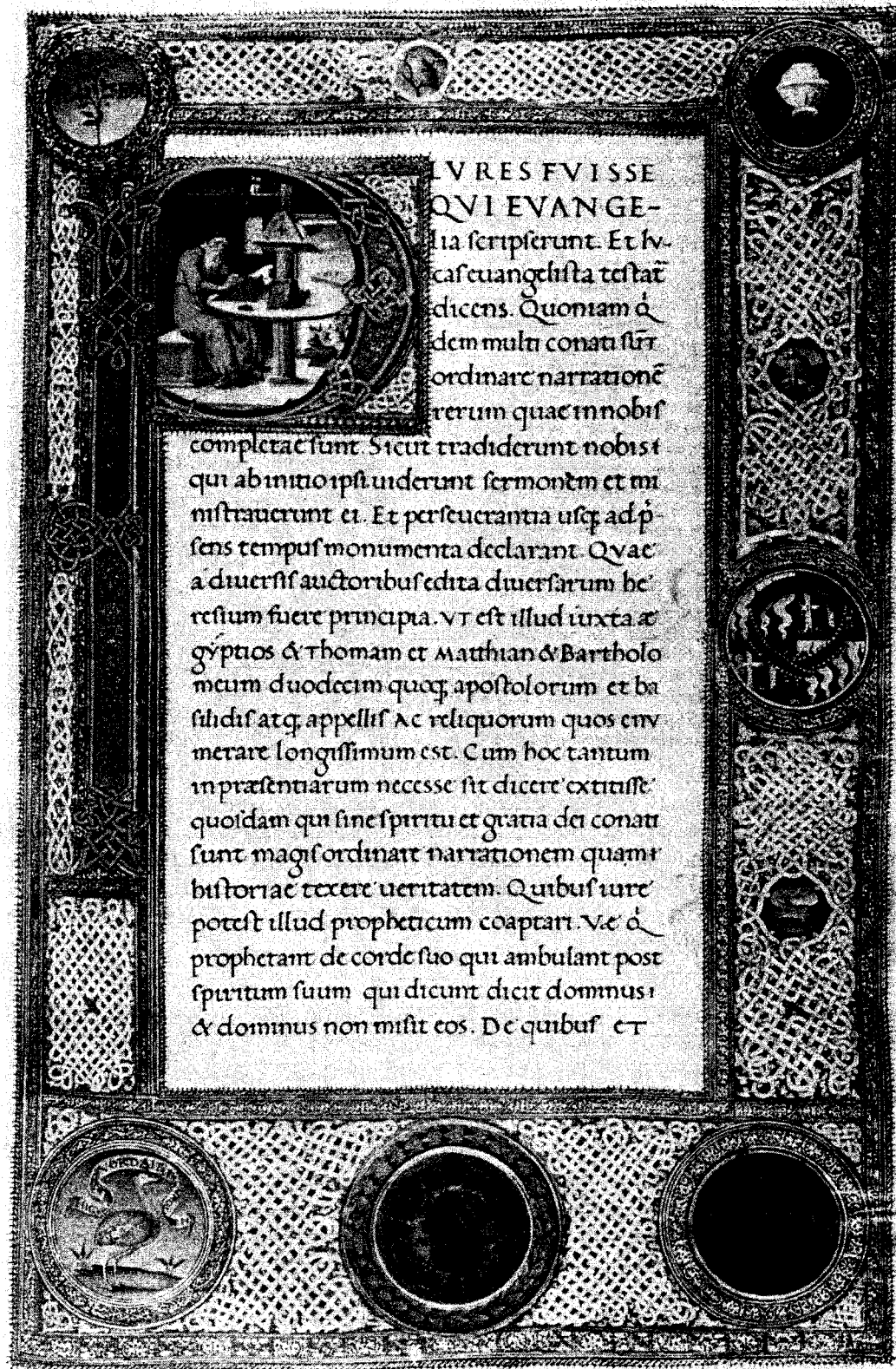


Figure 2: The *Codex Urbinas* containing the Latin version of the New Testament translated by Jerolamus. Reproduced from [9, page 153] by courtesy of Mr Ghorzo, president of the publishing house.

CORNELIVS·LVCIVS·SCIPIO BARBATVS·GNAIVOD·PATRE
 PROGNATUS·FORTIS·VIR·SAPIENSQVE – QVOIVS·FORMA·VIRTVTEI·PARISVMA
 FVIT – CONSOL·CENSOR·AIDILIS·QVEI·FVIT·APVD·VOS – TAVRASIA·CISAVNA
 SAMNIO·CEPIT – SVBIGIT·OMNE·LOVCANA – OPSIDESQVE ABDOVCIT

Figure 3: Inscription on the sarcophagus side of L. Cornelius Scipio Barbatus (259 B.C.). Vatican Museum. Archaic lapidarian script.

IL SESTO LIBRO
 DELLA ITALIA LIBERATA
 DA GOTTHI.

Il sestō muove il campō, e fà il gran vallon

NEL Tempō, che si stava entr'a le mura
 Il Capitaniō, a far ripari, e fossi,
 E che quei cavalier, ch'avean piljatō
 Paulo, eran iti a liberare Areta,
 I buon legati cō i tribuni insieme,
 Che si trovav nel'adunato stuolō,
 Facevano exercitar tutte le genti;
 Tal, che i tirani almen due volte al giornō
 Si riducevano sopra la quintana,
 Et imparavan quivi a fare il passō
 Pare di tempō, e di lungheza eguale,
 Da gir con esso almen tre milja a l'hora.
 Poi si davano al corsō, et al saltare
 Saraje, e fossi, et a natar ne l'onde;
 E dopō questō, ivano contra un palō
 Nadofo, e grossō, e di robustō legnō,
 Ch'avanzava sei pie sopra la terra,
 E con un scudō grave, et una maza,
 Ch'era di peso doppio d'una spada,
 Combattean seco, e come a un lor nimico
 Tentavan di ferirlo hor ne la gola,
 Hore ne i fianchi, et hora ne la faccia;

SESTO

96

Ne hī menavan mai se non di punta.
 Erano anchor quei giovinetti intenti
 A tirar haste, e trar balestre, et archi,
 Et a saltar sopra cavai di legnō,
 E destramente maneggiarsi in essi.
 Et imparavan anchor a portar pesi
 A cavar fossi, e far tutti i ripari,
 Ch'eran mestieri a circondare il vallon.
 Onde venendo Belisario il grande
 Una mattina nel spuntar de l'alba
 A riveder come si stava il campo,
 Per farlo caminar verso Tarento,
 Il vecchio Paulo se hī fece incontra
 Et in tal modo a lui parlando disse.
 Illustre Capitan, luce del mondo,
 Divisi havem hī alloggiamenti tutti,
 Et havem posto ogni centuria insieme
 Sotto il suo contestabile, che stansi
 A mangiare, e dormire sempre in un loco.
 Et ordinato havem, che ogni pramosso
 Habbia i suoi fanti, e stian presso al sergente;
 E che i sergenti stian cō i caporali,
 E quei cō i loro iconomi, e squadrieri;
 Tenendo sempre i consueti luochi.
 Et io hī faccio stare in questa forma;
 Acciō, che meglio si conoscan tutti
 L'un l'altro, e cerchi ognun di farsi onore,

Figure 4: Two facing pages of a XVI century Italian book set in print with a phonetic alphabet adapted to the sounds of the Italian language; this phonetic alphabet was partially used until the end of the XIX century, but the regular use of 'u' and 'v' for representing their own sound was not adopted until the second half of the XVII century.



Figure 5: Two facing pages of a Latin book printed by Elsevier in the second half of the XVII century where, at least in plain text, 'u' and 'v' are used consistently with their value. The original is set with a type size of 6.52 pt, so that reading requires a pretty good eyesight.

such a modern imitation of the past would really be appreciated by the scholars themselves.

To give an ancient look to ancient texts, while maintaining readability for modern readers, I'd suggest imitating the typesetting style of the little book reproduced in Fig. 5: titles have a pretty old fashioned look, the text is easily readable, the proper choice of fonts can add a lot, and the page's graphic design gives the final touch; I'd suggest reviving the "two-diagonal" method for positioning the facing pages one in front of the other. The little book of Fig. 5 gives many hints in this respect, although it must have been a sort of pocket handbook, that is a booklet without a pretentious look.

3 Ancient writing style

From Roman times we have actual specimens of marble, stone and clay inscriptions, besides a few papyri; before the Augustean period the glyphs were simple and without serifs. In Fig. 3 there is a transcription of a tomb epitaph, where I deliberately used sans serif fonts so as to imitate the original script.

The different uses of the glyph 'V' can be readily seen; among the others, the fact that the diphthong 'OV' is sometimes used as in Greek for the sound /u/, while the word *PARISVMA* implies the sound /y/: in the classical times the same word became in fact *PARISSIMA*. The diphthong 'AE' (or the ligature 'Æ') is missing and is still written as the Greek diphthong 'AI', where it comes from; the Greek diphthongs 'OI' and 'EI' still appear in that III century B.C. specimen, while such diphthongs will not be used any more in the classical age, from the I century B.C. onwards.

C·TREBIVS·L·F·LONGVVS
VETERANVS·COHORTIS
SECVNDAE·PRAETORIAE

Figure 6: Funerary inscription of the Augustean age. (City Museum of Bologna) Square lapidarian script.

Another example comes from an Augustean marble post, reset in Fig. 6 in Roman capitals; the original is engraved with the square lapidarian capitals that were used as a model to design most modern Roman upper case glyphs. It can be noticed that the ligatures 'Æ' are completely absent although there are three instances of the diphthong 'AE'.

4 Modern style for classical Latin

Classical Latin could be set in Roman capitals or, may be, in Roman small caps, only in case one wants

to give the flavor of classical inscriptions or handwritten codices; in such cases I'd rather use only the glyph 'V'.

The Latin text cited by Haralambous would turn out this way:

FLVMEN EST ARAR, QVOD PER FINES HAE-
DVORVM ET SEQVANORVM IN RHODANVM
INFLVIT, INCREDIBILI LENITATE, ITA VT
OCVLIS IN VTRAM PARTEM FLVAT IVDICARI
NON POSSIT. ID HELVETII RATIBVS AC
LINTRIBVS IVNCTIS TRANSIBANT. VBI PER
EXPLORATOIRES CAESAR CERTIOR FACTVS
EST TRES IAM PARTES COPIARVM HELVETIOS
ID FLVMEN TRADVXISSE, QVARTAM FERE
PARTEM CITRA FLVMEN ARARIM RELIQVAM
ESSE, DE TERTIA VIGILIA CVM LEGIONI-
BVS TRIBVS E CASTRIS PROPECTVS EST AD
EAM PARTEM PERVENIT QVAE NONDVM FLV-
MEN TRANSIERAT. EOS IMPEDITOS ET IN-
OPINANTES ADGRESSVS MAGNAM PARTEM
EORVM CONCIDIT: RELIQVI SESE FVGAE
MANDARVNT ATQVE IN PROXIMAS SILVAS
ABDIDERVNT.

But the reading of a long text set only in capitals is tiresome, so that common lower-case Roman or, sometimes, Italic type is more adequate for longer texts; in any case I find no reason for using just the glyphs 'V' and 'u', as done until the XVII century, because that is a bad habit that was done away with in all other modern languages which, nevertheless, up to that century were handwritten and printed with that curious anomaly: three glyphs to render the voiceless guttural consonant /k/, namely 'c', 'k' and 'q', and one glyph to render two different sounds as /u/ and /v/.

In passing, it may be interesting to compare the hyphenation produced by my modern Latin hyphenation patterns with those produced by the patterns created by Haralambous for medieval Latin. The same text, written in a modern way with the criteria I discussed above gets the following hyphens:

Flu-men est Arar, quod per fi-nes Hae-duo-
rum et Se-qua-no-rum in Rho-da-num in-
fluit, in-cre-di-bi-li le-ni-ta-te, ita ut ocu-lis
in utram par-tem fluat iu-di-ca-ri non pos-sit.
Id Hel-ve-tii ra-ti-bus ac lin-tri-bus iunc-tis
trans-ibant. Ubi per ex-plo-ra-to-res Caes-
sar cer-tior fac-tus est tres iam par-tes co-
pia-rum Hel-ve-tios id flu-men tra-du-xis-se,
quar-tam fe-re par-tem ci-tra flu-men Ara-
rim re-li-quam es-se, de ter-tia vi-gi-lia cum
le-gio-ni-bus tri-bus e ca-stris pro-fec-tus est

ad eam par-tem per-ve-nit quae non-dum flu-
men trans-÷-ie-rat. Eos im-pe-di-tos et ino-pi-
nan-tes ad-gres-sus ma-gnam par-tem eo-rum
con-ci-dit: re-li-qui se-se fu-gae man-da-runt
atque in pro-xi-mas sil-vas ab-di-de-runt.

Manual separation of prefixes by means of the underscore definition explained in [4] was used; in practice it was used only to separate the prefix trans-, and it is marked with a ÷ mark in the above text. In this respect Haralambous's patterns are far superior; of course for using Haralambous patterns it is necessary to `\lccode` and `\uccode` properly the letters 'u' and 'V' since they correspond to one another in passing from upper case to lower case and vice versa.

5 Conclusion

Several arguments have been set forth for explaining why classical Latin (and other ancient languages as well) should not be set according to the typesetting style used in the early age of printing and in the medieval codices; although the *restituta* version of Latin texts is enjoying a certain popularity among the scholars, the *restituta* "gives back" the appearance of writing and printing of the first centuries of this millennium, not the appearance of the original script of twenty centuries ago.

Although I believe in what I claimed in this paper, I might be wrong or miss some point; therefore I'd like to invite the readers of *TUGboat* to a broader debate on matters concerning the typesetting of old texts. Haralambous has already given fundamental contributions to this debate, not only with the paper that originated this comment of mine, but also with his many fonts for unusual languages; among the others let me draw attention to his paper [2] concerning the typesetting of old German, where he explains the motivations that pushed him to design his beautiful Schwabacher fonts. In [3] he also contributed, among others, the ancient Greek epigraphical characters and the rules for setting Greek epigraphs. There is enough material already, but except for [2], I believe most of us miss the aesthetic viewpoint.

References

- [1] Y. Haralambous, "Hyphenation patterns for ancient Greek and Latin", *TUGboat*, vol. 13, n. 4, pp. 457-469 (1992), resubmitted in Greek to the Academy of Athens with the title
Γιάννης Χαραλάμπους, "Συλλαβισμός των αρχαίων και νέων ελληνικών μέσω Η/Υ (Σύστημα ΤΕΧ)" Πρακτικά της Ακαδημίας Αθηνών (κατατεθέν για δημοσίευση)
- [2] Y. Haralambous, "Typesetting old German —

Fraktur, Schwabacher, Gotisch, and Initials", *TUGboat*, vol. 12, no. 1 (special issue with the Proceedings of T_EX90), pp. 129-137 (1991)

- [3] Y. Haralambous, "T_EX and those other languages", *TUGboat*, vol. 12, no. 4, pp. 539-548 (1991)
- [4] C. Beccari, "Computer aided hyphenation for Italian and modern Latin", *TUGboat*, vol. 13, no. 1, pp. 23-33 (1992)
- [5] S. Levy, "Using Greek fonts in T_EX", *TUGboat*, vol. 9, no. 1, pp. 20-24 (1988)
- [6] C. Mylonas and R. Whitney, "Complete Greek with adjunct fonts", *TUGboat*, vol. 13, no. 1, pp. 39-50 (1992)
- [7] César, *Guerre des Gaules*, transl. by L.-A. Constans, Les belles lettres, Paris, 1984
- [8] *The Chicago manual of style*, The University of Chicago Press, Chicago and London, 1982
- [9] *GRAFICA - Scienza, tecnologia ed arte della stampa*, Arti Poligrafiche Europee di Antonio Ghiorzo, Milano, 1984
- [10] *Ibidem*, Insert at page 250
- [11] B. Migliorini, "Breve profilo storico dell'ortografia italiana", *Ibidem*
- [12] G.G. Trissino, *Epistola de le lettere nuovamente aggiunte ne la lingua italiana*, Roma, 1524 (cited by B. Migliorini in [11])
- [13] G.G. Trissino, *La Italia liberata da Gotthi*, Roma, presso Valerio e Luigi Diorici, 1547
- [14] M. Fabii Quintiliani, *Institutionum oratoriarum libri XII*, Lugduni, apud Seb. Gryphium, 1544
- [15] *C. Cornelius Tacitus cum optimis exemplaribus collatus*, Amstelodami, Typis Ludovici Elzevirii, 1649
- [16] *Codex Urbinas latinus no. 10*, Manuscript written and decorated between 1474 and 1482 for the Duke Federico of Montefeltro (Urbino), now at the Biblioteca Vaticana, Vatican City
- [17] P. Flynn, "Typographer's Inn", *T_EX and TUG NEWS*, vol. 2, no. 1, pp. 3-5, 1993

◇ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy
beccari@polito.it

Comments on the paper “Typesetting Catalan texts with T_EX”

The interesting paper by G. Valiente Feruglio and R. Fuster on typesetting Catalan (*TUGboat* 14, no. 3, pp. 252–259) brings forth some spicy information about a language that is not widely known but is a legitimate member of the Latin family.

This paper sets forth another problem that is going to be more and more important as multilanguage facilities become available for more and more T_EX users: we all are attracted to make statements concerning languages for which we do not have sufficient supporting evidence or a specific competence (... starting with myself, since I am not immune from this “weakness”).

Valiente Feruglio and Fuster make two statements concerning italian that are wrong:

a) Note 4 on page 255: “... It differs from Spanish and Italian: in these two languages all combinations of ‘i’ or ‘u’ with another vowel are diphthongs.”

b) First paragraph on page 257: “... For instance, Latin INTELLIGENTIA derives into French *intelligence* and Italian *intelligenza*, while Latin SELLA derives into French *selle* and Italian *sella*. Then these two languages use the same orthography for two different phonemes.”

Statement a) can be easily corrected for what concerns italian by saying that “all combinations of *unstressed* ‘i’ or ‘u’ with another vowel are diphthongs.” T_EX does not know anything about stressing, especially in italian where stress accents are not compulsory, so that the imprecision of statement a) has no consequences for T_EX.

Statement b) is definitely wrong for what concerns italian, and I’d say also for what concerns french, but I leave the french issue to French speakers, since, although I speak fluent french, I am not a French speaker.

Italian orthography is a phonetic one and almost perfectly matches the phonemes of the language; although stress (tonic) accents are not compulsory and phonic accents are optional and very seldom used, although the two variants (voiced and unvoiced) of the letters ‘s’ and ‘z’ are not distinguished with different glyphs or graphemes, I’d say that italian spelling is perfectly adherent to the semantic value of the various phonemes. In other words the two variants of ‘s’ and ‘z’ don’t change the meaning of a word, just reveal the regional origin of the speaker. In any case these points have nothing to do with the “long l” phoneme (|ll|) and the “lateral palatal l” phoneme (|λ|): in italian the former is spelled ‘ll’ and corresponds to the catalan ‘ll’, while the latter is spelled ‘gli’ and corresponds

to the catalan ‘ll’. In conclusion the double ‘l’ in *intelligenza* and *sella* are pronounced exactly the same in italian, and it is not true that the same orthography is used for two different phonemes.

Ironically the typical italian trigraph ‘gli’ is present in Valiente Feruglio’s second family name; I would not be surprised if he discovered some Italian ancestors in his maternal genealogy. I found half a dozen Feruglio entries in the telephone directory of my city!

Of course these remarks do not invalidate the excellent paper by Valiente Feruglio and Fuster, and I warmly thank them for disseminating information on the catalan language that, apparently, received the attention it deserves only in this century.

Claudio Beccari
beccari@polito.it

Comments on the comments: Typesetting Catalan texts with T_EX

*O Dio, la Chiesa Romana
in mani dei catalani!*
Pietro Bembo (secolo XV-XVI)

We appreciate the remarks by C. Beccari on our paper “Typesetting Catalan Texts with T_EX” (*TUGboat* 14, no. 3, pp. 252–259), and take the opportunity to make the statements therein more precise.

We agree with Beccari’s first statement in that all combinations of unstressed ‘i’ or ‘u’ with another vowel are diphthongs in Spanish and Italian. As a matter of fact, unstressed ‘i’ and ‘u’, in contact with a vowel, are semivowels (or semiconsonants) but they become full vowels when stressed, and therefore there is no diphthong when a stressed ‘i’ or ‘u’ is combined with a vowel.

The second statement made by Beccari, however, needs more clarification. Our statement that Italian uses the same orthography (ll) for two different phonemes (|ll| and |λλ|) is not well posed. It does not refer to the current italian spelling and pronunciation alone but in the context of its relationship to the evolution from Latin to modern languages.

Romanic languages differ in the way they spell and pronounce words derived from Latin, depending on whether the words derive from Classical Latin or from vulgar Latin. The solutions adopted by the

different romanian languages are not much different, although they are not identical.

A usual phenomenon in this sense, at least in the case of Catalan and Spanish, is the *palatalization* into $|\lambda\lambda|$ of words derived from vulgar Latin. This palatalization is represented by ‘ll’ in Catalan and Spanish and by ‘gli’ in Italian. This phenomenon, however, does not occur in words of Classical Latin origin, although in some languages (Catalan and Italian, among others) there is a duplication or *gemination* which is what is represented by ‘ll’ in Catalan, while it is represented by ‘ll’ in Italian. These words are also written with ‘ll’ in French, although there seems to be no difference in pronunciation, while in Spanish it is written ‘l’ and pronounced $|\lambda|$.

For instance, the word INTELLIGENTIA, which is of Classical Latin origin, derives into Catalan *intelligència*, pronounced $|\lambda\lambda|$, into Italian *intelligenza*, pronounced $|\lambda\lambda|$, into French *intelligence*, pronounced $|\lambda|$, and into Spanish *inteligencia*, pronounced $|\lambda|$.

We hope to have clarified our statements with this discussion. Although Valiente Feruglio’s second family name contains the trigraph ‘gli,’ which corresponds to the $|\lambda\lambda\lambda|$ phoneme, there is no record of Roman ancestors with that family name known to the authors, while Valiente Feruglio’s last Italian ancestor was born in Ramanzacco, in the province of Udine, in 1861, and died in Santa Fe (Argentina) in 1937. As it turns out, however, having Italian ancestors does not guarantee a good knowledge of the Italian language, for Valiente Feruglio does not speak fluent Italian... yet.

- ◊ Gabriel Valiente Feruglio
valiente@ipc4.uib.es
- ◊ Robert Fuster
mat5rfc@cci.upv.es

Book Reviews

Book review: *Math into T_EX*

Nico Poppelier

George Grätzer, *Math into T_EX*, A simple introduction to $\mathcal{A}\mathcal{M}\mathcal{S}$ -L $\text{T}_{\text{E}}\text{X}$. Birkhäuser 1993, 294 pages (including indexes) ISBN 0-8176-3637-4 (includes diskette).

Even though document-preparation packages of the WYSIWYG type become better and better every year, the majority of mathematicians, computer scientists and physicists still use T_EX, in any of its many flavours. In 1982 the American Mathematical Society released $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX, which has certainly contributed to the popularity of T_EX in mathematician’s circles. Around 1990, the macros of $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX were made available for the expanding community of L $\text{T}_{\text{E}}\text{X}$ users in the form of $\mathcal{A}\mathcal{M}\mathcal{S}$ -L $\text{T}_{\text{E}}\text{X}$. And now, finally, there is a book that describes this useful but complex extension to L $\text{T}_{\text{E}}\text{X}$ for beginners.

George Grätzer teaches mathematics at the University of Manitoba (Canada), and has tried to write a book “from a user’s point of view”. His book consists of three parts. In part I, *A short course*, he explains how to install $\mathcal{A}\mathcal{M}\mathcal{S}$ -L $\text{T}_{\text{E}}\text{X}$ on an IBM-compatible PC under DOS, and on an Apple Macintosh, and then explains the basics of $\mathcal{A}\mathcal{M}\mathcal{S}$ -L $\text{T}_{\text{E}}\text{X}$. In part II, *A leisurely course*, he goes back to the fundamentals of typing text and formulas in T_EX, and then explains all a mathematician, or engineer or scientist, needs to know about $\mathcal{A}\mathcal{M}\mathcal{S}$ -L $\text{T}_{\text{E}}\text{X}$. Finally, part III is about customizing.

Math into T_EX was written with the basic idea behind the design of L $\text{T}_{\text{E}}\text{X}$ nestled firmly in the mind of the author. Mr. Grätzer emphasizes proper usage of L $\text{T}_{\text{E}}\text{X}$ while writing papers and books. His approach is didactically very good, he takes his time explaining things, and gives enough examples. *Math into T_EX* is not a book about L $\text{T}_{\text{E}}\text{X}$ itself, so the author does not cover all of L $\text{T}_{\text{E}}\text{X}$. Nevertheless, he treats tables (briefly) and BibT_EX, which makes the book a useful introductory text as well as a handy reference for authors who do not want to know more about T_EX and its flavours than is necessary for writing a research paper.

Even though many people, including several well versed in T_EX and L $\text{T}_{\text{E}}\text{X}$, read draft versions of *Math into T_EX*, the book contains a lot of errors. One of the more interesting ones is this one: in section 3-4.8, on hyphenation, the author gives the following example

`\hyphenation{data-base Birkh-h\"auser}`

I often wish that \TeX could do tricks like this, but alas!

These errors will be removed in a second, revised edition I assume (if enough people buy the book), and then I can say in all truth that *Math into \TeX* is a welcome addition to the growing collection of books about \TeX and related topics.

- ◊ Nico Poppelier
Elsevier Science Publishers
Amsterdam
The Netherlands
Internet:
n.poppelier@elsevier.nl

Book review: *\TeX in Practice*

T. L. (Frank) Pappas

Stephan von Bechtolsheim, *\TeX in Practice*. Springer-Verlag, New York, 1993. Four volumes: ISBN 0-387-97595-0, 0-387-97596-9, 0-387-97597-7, 0-387-97598-5. Price: Four-volume set is \$169, individual volumes \$49 each. Source for the *TIP* macros is available for downloading.

My first encounter with *\TeX in Practice* was more than five years ago when I purchased a preliminary draft. Although it was far from complete—many sections were “to be completed”—*TIP* seemed to promise a complete introduction to \TeX . Although I found *TIP* unusable at the time, I assumed its shortcomings were due to its preliminary status.

My second encounter with *TIP* occurred more than a year ago when I received a copy from von Bechtolsheim. With just a few “minor” changes, this was to be the camera-ready copy that his publisher, Springer-Verlag, would receive. I found the copy extremely difficult to handle since about 1,000 of the pages were printed one to an 8.5×11 sheet. Still, I scanned through the material and again came away with the impression that *TIP* was going to play a significant role in making \TeX more accessible. Although I was less enthused than the first time, I again assumed that my discomfort with *TIP* would go away when I could look at the published result.

My third encounter with *\TeX in Practice* occurred this past Fall, when I received a copy directly from Springer-Verlag. *TIP* is published as an 1800+ page, four volume set: Volume I: Basics; Volume II:

Paragraphs, Math, and Fonts; Volume III: Tokens, Macros; and Volume IV: Output Routines, Tables.

A word from the publisher

Now that I have seen the final product, my impression of *TIP* has changed drastically, but before explaining why, let me share with you the following comp.text.tex post from Dave Rogers that appeared on 27 Oct 1993:

As some of you have perhaps noted, I am the Editor of the *Monographs in Visual Communication Series* for Springer-Verlag which includes *\TeX in Practice* by Stephan von Bechtolsheim. The forward [*sic*] in the volume is **not** what I wrote. It was modified by Stephan without my concurrence. The unmodified version is given below. I think the second paragraph is particularly interesting as I have noticed a significant dichotomy in the way different people approach \TeX .

Further, I take **no** responsibility for the quality of the typesetting of the book nor for the quality of the English or the proofreading. I consider the book a prime example of a very poor design and typesetting job. The English is atrocious and the proofreading is nearly non-existent. Both the editorial and production departments at Springer-Verlag and I tried to get these defects corrected but with little success.

Having said that why did we publish the book? Basically because it contains very valuable information about the use of \TeX . Information that the \TeX community very much needs. After all, the fundamental purpose of a book is to convey information. So the decision was made to ignore the defects and publish it anyway.

I trust that you can ignore the presentation defects in the book and concentrate on the information.

Dave Rogers

Blame the publisher!

Although I agree with Dave Rogers' criticism of *TIP*, I think von Bechtolsheim is less to blame in this than Springer-Verlag. Rogers' writes, “After all, the fundamental purpose of a book is to convey information. So the decision was made to ignore the defects and publish it anyway.” While this is true, Springer-Verlag already has a means of publishing such material in its lecture notes series such as *Lecture Notes in Computer Science*. With that series readers know they are purchasing books that

may lack polish but are willing to accept that in exchange for timely access to material. That is the premise that the lecture notes series is based on.

With regard to *TIP*, Springer-Verlag should have insisted on proofreading the material, told von Bechtolsheim what changes needed to be made, making it clear that *TIP* would not be published unless these changes were made. By not doing so, Springer-Verlag has embarrassed itself. Springer-Verlag should also be embarrassed to have allowed such poor quality design and typesetting to be associated with any book it publishes.

Without doubt, von Bechtolsheim made fundamental and significant mistakes in preparing this series, but a publisher owes it to the author and the author's readers to ensure that such mistakes do not occur. Springer-Verlag failed to do this and is thus ultimately responsible for the poor quality of *TIP*.

One of \TeX 's virtues is that an author can produce a high-quality manuscript, produce camera-ready copy, and then deliver it to a publisher. *TIP* shows just how badly this process breaks down if the publisher shirks its responsibility.

Who is this book for?

Here is the second paragraph Rogers mentioned in his comp.text.tex post and which von Bechtolsheim removed from the foreword.

\TeX itself can be considered from at least two significant and quite different viewpoints. The first is as a typesetting *system* in which the typesetter has precise control of the placement of characters and white space, the design and make-up of lines, equations, paragraphs, and pages. The second is as a macro-extensible *programming* language. Fundamentally, *TeX in Practice* addresses \TeX from the latter viewpoint.

I agree with these remarks.

When I looked at the second draft of *TIP* I assumed that typesetting was covered but that I just couldn't find it in the 1000+ sheets of that draft, especially since this version of *TIP* did not yet have an index. It seemed reasonable to make this assumption since the name of the set is *TeX in Practice*—a better name would be *TeX Macros in Practice*.

Without doubt, *TIP* is for someone who wants to learn how to write \TeX macros. Beyond that it is unclear who should read *TIP*.

The preface and the first 10 pages or so explain what \TeX is, why you need a good text editor, the files that \TeX uses, and assorted other introductory information. They give the impression that *TIP* is an introduction to \TeX . To the contrary, anyone

using *TIP* as an introduction would be scared away from \TeX forever!

TIP requires more than a passing knowledge of \TeX since it freely uses \TeX concepts without first explaining them. Sometimes von Bechtolsheim attempts to get around this by giving a cross-reference to the volume and section where the concept is defined. Sometimes he doesn't.

For example, von Bechtolsheim starts using `\catcode` fairly early in *TIP* so that he can use @ as a letter in the macro examples. However, no discussion of `\catcode`—not even a cross-reference—appears until volume III.

On the subject of cross-references, in the first 100 pages I had to constantly refer to volume III. Of course, these cross references lead to others, and so on. In the length of time I spent going through the first 100 pages, I should have gotten through the entire first volume.

As for the macro examples, they also rely on cross-references to macros described in later volumes. In some cases the examples uses macros that are not described in the book—I had to look at the actual macro files on at least one occasion.

For example, the following appears on page 73 of Volume I:

```
\NameDef{@TheC-#1}{%
  \expandafter\expandafter\expandafter#2%
  \expandafter{\csname @C-#1\endcsname}%
}
```

Up to this point no explanation is given as to why @ can be used as a letter. The control sequences `\expandafter`, `\csname` and `\endcsname` have not been described up to this point either, nor is their use in the example explained.

`\NameDef` is a *TIP* macro that is not defined until volume III. `\NameDef{X,Y}` creates a macro `\@TheC-X`, that expands to the current value of the counter `\X`, formatted using macro `\Y`. It is certainly a useful macro, but even its use is not explained.

The choice of macro examples is hard to understand. It almost seems to be based on the approach "Why use a simple example to make things clear when a complex, poorly and incompletely explained, heavily cross-referenced example, whose true usefulness won't be explained until a later volume, will confuse and frustrate the reader."

Summing up

The lack of style, polish, and proofreading, combined with von Bechtolsheim's difficulties with the English language, make for brutal reading. It is not a book for casual reading, nor is it a book for novice

\TeX users. Only \TeX users who know how to program (in a programming language or in \TeX macros) should attempt to read this book, and then such users should already be familiar with most of \TeX 's advanced features.

However, to be fair, *TIP* covers just about every aspect of writing \TeX macros you will ever need to know. Most of the *TIP* macros can be used with \LaTeX as well as plain \TeX and the macros do provide substantial capabilities.

Perhaps the best way to look at *TIP* is as an outstanding collection of macros that happen to come with four volumes of documentation.

◊ T. L. (Frank) Pappas
338 Francis Drive
Havertown, PA 19083
fpappas@mcimail.com

Book review: *Il \TeX – Introduzione al linguaggio e complementi avanzati*

Claudio Beccari

Gianni Gilardi, *Il \TeX – Introduzione al linguaggio e complementi avanzati*. Zanichelli/Decibel, Bologna 1993, 226 pages, hardbound. ISBN 88-08-10860-0, 36 000 Lit \approx 21 US\$.

Another book on \TeX was published in 1993 in Italy; the publisher Zanichelli is one of the leading scientific publishers in this country and in addition to the many valuable university books (remember: the University of Bologna was founded in 1088 AD) is also publishing a series on typography, book design, desktop publishing, etc.; Zanichelli is one of the few publishers that accepts (\LaTeX) manuscripts from authors.

With the cooperation of the publishing company Decibel Editrice of Padua, this book on \TeX is perfectly set with \TeX in 12pt (with a tight baseline skip of 13pt as is customary in Italy) and done with a high resolution phototypesetter so that the traditional cm fonts have a marvelous look and excellent readability.

The page design looks like the work of a book designer, although no explicit information is given in this regard on the colophon page; both headers show the section title filled with an `\hrulefill` so as to be aligned with the external margin, while the left foot shows the chapter ordinal and the right foot the chapter title; both footers have the page number

aligned with the external margin and the copyright notice and the ISBN number aligned with the internal margin. Many other design details are properly chosen but this is not the place to discuss such fine points. The quality of paper and the hard cover complete the book in such a way that one has the impression of a lasting work; the price is fully affordable and lower than one might expect for such a high quality technical product.

The book is divided in four large chapters: 1) \TeX and the world of text, 2) \TeX and the world of mathematics, 3) \TeX further on, 4) \TeX beyond survival. The first two chapters are at a beginner's level and describe in a very simple way how to set text or math. There are plenty of examples and the various control sequences that get involved are chosen so as to follow an increasing level of difficulty.

I particularly appreciated the math examples that consist of complete sections of sample math articles or books, where a whole subject is dealt with, so that theorems, lemmas, in-line math expressions, displayed labeled and unlabeled equations, simultaneous equations, combinatorial diagrams, ... are shown together with the code for producing them by means of the traditional plain \TeX macros. The fact that the author is a professor of Mathematical Analysis (at the University of Pavia) explains why the math examples are so well chosen.

The third chapter deals with less elementary topics, such as macros without parameters and tabular alignments and tables with both vertical and horizontal rules. Again, a multitude of examples helps in understanding the intricacies of certain \TeX constructs that are necessary with the `\halign` command. For the first time in a book on \TeX I see an example of `\valign` that is not trivial.

The author considers the level of chapter 3 the "survival level"; this is why the title of chapter 4 is "beyond survival". The last chapter in fact deals with the more sophisticated macros containing parameters, conditionals, delayed expansions, and the like. The only important topic that is not dealt with in the whole book is the set of commands and macros for producing the dvi output, that is the output routine. This is a precise choice of the author — he did not want to write a handbook on \TeX , but a book on \TeX that could help beginners to become ever more confident with the language and reach good levels of programming skills so as to be able, if necessary, to deepen their knowledge of certain topics directly from the \TeX book; the latter, as everybody admits, is not a book for beginners (although most of us had to begin with the \TeX book).

Gianni Gilardi achieved another important goal: that of writing the book in an informal way, so

that the reading is pleasant and fluid; as a university professor myself, I know the difficulty we have (in my country) in writing informally, because the tradition of Italian academicians is just the opposite, that is to be formal in every circumstance. The informal attitude is achieved also with the help of a couple of characters, Mr. Tizio and Mr. Caio (who, together with Mr. Sempronio, make up the triad of persons that, since Roman times, have always been used in all examples of legal cases — you see, the academy shows up again!); Tizio is a \TeX guru, or at least a \TeX wizard, while Caio is a stubborn and clumsy beginner who makes a lot of mistakes, and is always asking Tizio for help. These two guys are also depicted in an appendix, \TeX grafica, that displays the graphic capabilities of plain \TeX without the help of special fonts.

The book is completed with a short guide: an appendix where a list of \TeX commands is associated with the most common typesetting tasks identified by simple keywords, so that if you look for, say, “page numbering” you find `\folio`, `\nopagenumbers`, `\pageno`. The instructions for this short guide say that you must use it in conjunction with the analytical index where every command (primitive, plain, or defined in this book) is reported and marked with the page references where the greatest part of the information about that command can be found.

The book does not contain important errors; there are very few typos, and for what concerns \TeX I could notice only the following (minor) ones: the commands `\smallbreak`, `\medbreak`, and `\bigbreak` are described as doing the same as the corresponding `\dotskip` commands with the addition of inviting \TeX to break the page there; on the contrary, the former macros clearly test the last skip amount before doing anything. Further on, `\smash` is described as operating only in math mode, while the definition of `\smash` clearly contains `\ifmmode ... \else ... \fi`.

In conclusion, I find this book a very valuable one for beginners, who may become, with its help, good \TeX users with relatively little effort; I recommend it also for those \LaTeX users who want to start writing for themselves option or style files containing macros of a good level of sophistication; chapter four might be very helpful.

I regret that the book does not spend a word¹ about the language facilities offered by \TeX 3.x;

¹ This is not completely true; languages associated with counters are used for showing how to use `\ifcase` and `\ifnum` in an example macro that sets the date for several languages.

in the United States this problem seems to be not so important but in Europe we use several languages for all purposes — technical, scientific, business, tourism, etc.; we must use at least the national language and English (the variety defined as EFL: English as a Foreign Language) as the *lingua franca* of every international activity. Therefore a section on language shift and customization might have been of great help.

◊ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy
beccari@polito.it

Book review: *Stop Stealing Sheep*

Merry Obrecht Sawdey

Erik Spiekermann & E. M. Ginger, *Stop Stealing Sheep & find out how type works*. Adobe Press, 1993; 174 pp. ISBN 0-672-48543-0. \$19.95.

The significance of the title of the book *Stop Stealing Sheep* is revealed in the sidebar on page 7 of the book. The authors quote Frederick Goudy, an American type designer, as saying, “Anyone who would letterspace black letter would steal sheep.” They point out that they have also seen “lowercase” used in the quote instead of “black letter” but that the idea is the same. Mr. Goudy was given to making broad-based, opinionated statements. He eventually apologized for this one, but this is the kind of passion that the subject of design and typography elicits in a great many people.

Design, typesetting, and printing used to be fields limited to a chosen few who demonstrated the skill, experience, learning, and compulsiveness to work at it. If you ever get a chance to work in a letterpress studio with printers who print using traditional methods, you’ll experience this fervor firsthand. It’s not something that is taken lightly by those who indulge in it.

In recent years, with the advent of highly accessible computers and software almost anyone with an inkling to tinker with page layout software is able to participate in the great publishing frenzy. More people than ever before are producing brochures, signs, their own business cards, self-published books, whatever printed material can be

done with 4 megabytes and access to a printing device. Academia has embraced the new technology not only for use in writing papers (no more dissertations stored in the icebox), but for mailing them electronically to journals and conferences.

One difficulty with typography and design is that we're talking about something that hovers on the edge of being an art and a craft. We need books, advertisements, labels, brochures, mailings, and so on that are legible, readable, and communicate the intended message with elegance and style. With all the magical software available we find people going a little crazy making their text go in circles and wavy paths, shadowing the letters with bright colors and sparkles, filling every piece of available white space with type and illustrations. The ability to do this with all the available powerful, creative software brings out the budding artist in all of us. This isn't always helpful at putting the message across.

Another complication is that with typography, we're basically talking about working with illusion, "optical space," not mathematical precision. Mathematical precision and specifications are possible (and practical), but not always appropriate. While there are many rules, guidelines, and (strong) opinions about good typography, it basically comes down to what looks good. Printing is an old and time-honored craft; rules for what is right and wrong have evolved over many generations of printers and their presses. New technology has jostled those traditions and our understanding about what is readable, legible, and attractive.

Stop Stealing Sheep & find out how type works by Erik Spiekermann and E. M. Ginger was written to help us look at one of the tools we use to transmit information, namely, type. Their goal, as they state it, "is to clarify the language of typography for people who want to communicate more effectively with type." The text of the book is written for people who have little or no experience with typography. The information in the sidebars is "[f]or those who already know something about type and typography and who simply want to check some facts, read some gossip, and shake their heads at our opinionated comments..." The illustrations are "familiar images used ... to show that typography is not an art for a chosen few, but a powerful tool for anyone who has something to say and needs to say it in print."

The book contains samples of different typefaces and some samples of different handling of wordspacing, letterspacing, and font choices. The sidebars contain insights into the purpose the designer might have had when he or she designed a particular typeface as well as a few other histori-

cal tidbits. The authors use common metaphors to aid our understanding of the function of type such as highways, music, running races, and furniture. There are even good discussions of type used for business forms, faxes, and road signs.

The goals of the authors are admirable, and even to some extent accomplished within the pages of the book. The subject is handled with creativity and humor. Anyone who reads the book, previously intimidated by the handling of type and all the myriad decisions that go with it, will find themselves a little more comfortable with typography and its rules and guidelines. The reader will look at advertisements, road signs, magazine layouts, even fax forms with new intelligence (and criticism, probably).

While the intent of the book and much of its information is welcome and helpful, I have a few reservations about the book that keep me from recommending it wholeheartedly. First the layout of the book is intricate and complicated. There are many elements: figure captions, sidebars, the text, type samples, examples of text in different fonts, and the photographs. Many of the photographs are busy with texture and objects. The layout is set up on a specific grid system (as explained in the text), but there is a lot of information being communicated in a relatively small amount of space. Page numbers aren't used on all the pages because the layout doesn't leave room. The text on some of the pages extends almost to the very edge of the page, making it difficult to hold the book and look at the type at the same time. Much of this could have been helped if the book were bigger and allowed for more white space and better organization. As the information is presented, it would probably be overwhelming or confusing for the beginner.

There is a different color used in each chapter of the book. The second color is used consistently from chapter to chapter, making the color change attractive and entertaining, even helpful as a way of distinguishing between sections. However, the sidebar type always appears in this second color in a fairly small type size. There are two chapters in which the second color is yellow or mustard. The second colors used in these two chapters make the sidebar information difficult to read, especially since, due to the vagaries of the printing process the second color fades in and out over the course of the pages. There is also a color screen used in all the chapters behind some of the type samples that is a very light screen in the second color. The color of this screen is often so light it adds to the confusion of the page rather than illuminating.

The book is entertaining and interesting, but there isn't a huge amount of practical information in the book. That may or may not be a problem for you, depending on what you're interested in; not everyone wants or needs to delve deeply into typography and design. If you're just interested in an overview of typography to gain a simple awareness of what is available and how type is used, this might be a good book for you. If you're at all serious about the use of type in your work, I recommend reading more widely. There are vastly diverging points of view and perspectives on design and typography. The authors include a bibliography at the end of their book for further reading, and many bookstores have books on design and typography in their art sections. I've had good experience with the selection in university libraries as well.

There are myriad opinions about what is good, right, and true about the use of type, and it helps to get a sense of the range before you make your own decisions. I find a historical understanding of typesetting and printing helpful (usually given in the introduction or first chapter of many books on design or typography) to understand where we've been, past and present assumptions about what is readable and legible, and what's been done and what's available with design and type. Once you're done some background checking, just pay attention to the print around you: movie titles and credits, advertising, labels, books, brochures, forms, whatever you see that uses type. Develop your own list of fonts you like to use, your own tastes, stay open, and keep experimenting.

◇ Merry Obrecht Sawdey
3532 Bryant Ave So.
Apt. 112
Minneapolis, MN 55408
sawdey@denali.ee.umn.edu

Pre-publication review: *Practical SGML*

Nico Poppelier

Eric van Herwijnen, *Practical SGML*. Kluwer 1994, 284 pages (including indexes). To be published.

In my review of the first edition of *Practical SGML* by Eric van Herwijnen (*TUGboat* 13, no. 2), I praised it as 'one of the best books on SGML currently available.' It still is one of the few books on the practical application of SGML, by someone who has used SGML in practice rather extensively. The new edition has undergone significant changes with respect to the previous one. Unfortunately, they are not all changes for the good: the book still contains a lot of practical information — more than the first edition — but it is not a *better book*.

As a reference work the quality of the book has certainly improved. More material has been added, and the book has been largely re-structured. The previous edition consisted of three parts, *Getting started with SGML*, *Advanced SGML* and *SGML implementations*. The new edition has more chapters, grouped together in four parts, *Getting started*, *Writing a DTD*, *Customizing SGML* and *Special applications*. Especially the second part, about how to write a DTD (document type definition), has improved a lot, with chapters on document analysis, structure diagrams, and the various declarations one can find in a DTD. Part III, about customizing SGML, describes the SGML declaration, and SGML features such as minimization, marked section and short references. It also describes the problems that can arise with ambiguous definitions, and gives advice about how to avoid ambiguities. Under the heading of 'Special applications' (part IV) Mr. van Herwijnen discusses SGML and EDI, SGML and mathematics, and SGML and graphics. He also explains the relation between SGML and other ISO standards, such as, e.g., DSSSL and SPDL. In all the examples in the book the public-domain SGMLs parser is used, which makes it possible for most readers to try the examples for themselves.

On the negative side however: so much material is now contained in the book, especially in the form of figures and tables, that the book, in my opinion, is not a *pleasant-to-read* introduction to SGML any more. Another thing which I find rather distressing, at least in the pre-publication copy the author kindly sent me, is the design: the book uses too many fonts, in sometimes unharmonious combinations, the distribution of vertical space is uneven, and the placement of tables and figures leaves a lot to be desired. A possible explanation could be that this new edition of *Practical SGML* was prepared

using SGML, and was formatted using Adept 5.0 from ArborText Inc. Obviously, designing a book that is comfortable to read is not the same as writing a ‘FOSI’, an output specification for ArborText’s Adept product. I hope that the publisher will work hard on improving the layout of the book, but I have my doubts.

Of course, this says nothing about the applicability of SGML to book production, but only about the quality of available SGML tools, or the expertise of the people using these tools. That computers are capable of producing more readable and more attractive books is shown by a book co-authored by one of Mr. van Herwijnen’s colleagues at CERN, namely *A L^AT_EX Companion*, by Michel Goossens, Alexander Samarin and Frank Mittelbach. But then, of course, that book was made with L^AT_EX!

◇ Nico Poppelier
Elsevier Science Publishers
Amsterdam
The Netherlands
Internet:
n.poppelier@elsevier.nl

Book review: *Literate Programming*

Christine Detig and Joachim Schrod

Donald E. Knuth, *Literate Programming*. Center for the Study of Language and Information, Lecture notes no. 27, Stanford 1991. (Distributed by the University of Chicago Press.) xvi + 386 pp., index and comprehensive bibliography.
ISBN 0-9370-7380-6 (pb), 0-9370-7381-4 (hc).

The essence of literate programming?

Say it twice!

— D. Knuth (1993)

This book is an anthology of works by Donald Knuth; it tells us the story of literate programming. It consists of an introduction, a lecture, eight articles, three book excerpts, a program, and a bibliography. John Hobby is responsible for the selection of the contents; the introduction is the only text previously unpublished.

The presented material spans almost 20 years of Knuth’s work, in which literate programming developed from concerns about the quality of software description, through first ideas to categorize and improve it, to applications and experience reports based on the methods and tools he has created.

Contents

The collection starts with the *Preface* that presents Knuth’s views on the relation between the different texts selected by Hobby. It shows the “red thread” of the book and gives advice on how to read this book. Besides this introduction, the only new material is a paragraph at the start of each text that presents the context of original publication.

The first text, chapter 1, is the lecture given by Knuth in 1974 when he received the *Turing Award*, the most important Computer Science award. Already at that time, Knuth had named the basic principles and motivation of literate programming:

The chief goal of my work as educator and author is to help people learn how to write *beautiful programs*. [...]

[The goals of correctness and adaptability] are achieved when the program is easily readable and understandable to a person who knows the appropriate language. [...]

Please, give us tools that are a pleasure to use, especially for our routine assignments, instead of providing something we have to fight against.

In this lecture, *Computer Programming as an Art*, Knuth argues that programming has much in common with music composition. Here we also find the reasoning behind the statement that programming is not a science, but an art. Knuth still holds the professorship for the “Art of Computer Programming” and this chapter shows us basic principles of his whole professional life.

Chapter 2 presents one of Knuth’s most cited articles: *Structured Programming with go to statements* (1974). This article must be read in the context of its time: People had just started to develop programs in a systematic, structured way; the scientific community was discussing for the first time how to write long-living programs. It’s written in the context of Dijkstra’s famous letter “Go to statement considered harmful” and shows that the problem of unstructured programs is not based on language constructs.

Chapter 3 continues with an early effort of Knuth to present a larger piece of code in a readable and understandable way: *A structured program to generate all topological sorting arrangements* (1974). According to Knuth himself, the presentation of this article left much to be desired. He had realized that writing programs intended for critical reading means to make construction and evolution recapitulable. This requires other forms of writing and presentation than the old, machine oriented, style.

Chapter 4 marks a turning point in the story told by this anthology — Knuth takes the step from structured to *Literate Programming* (1984). He formulates the credo of this new paradigm:

Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do.

This new way of thinking is presented by an example; that example used the tool WEB he had created for the work on the programs T_EX and METAFONT. WEB supports the intertwining of informal and formal parts: the former marked up with T_EX tags, the latter, pieces of Pascal code. These code pieces are organized as refinements. Besides the facilities outlined above, WEB is also burdened with a set of features meant to overcome inherent deficiencies of the underlying programming language Pascal.

Both readers and writers face new requirements and viewpoints with this new programming paradigm. In chapters 7 and 9, two reflections on this aspect are presented: *How to Read a WEB* (1986) and an excerpt from *Mathematical Writing* (1987) that reports from discussions between Knuth and students on the subject of literate programming.

We just skipped chapter 8; there we enter an area well known to our fellow TUGboat readers. Two *Excerpts from the Programs for T_EX and METAFONT* (1986) show the application of the literate programming paradigm in production-quality software. We read something that seems to come from a textbook on algorithm design, not from real-life programs that are in use at hundreds of thousands of installations.

Chapters 10 and 11 present data to support the claim that literate programming leads to programs that are better maintainable. *The Errors of T_EX* (1989) presents the history of T_EX, categorizes the errors Knuth made, and makes a thorough analysis of the development process. The data beyond this analysis is the diary, *The Error Log of T_EX* (1978–1991). This diary gives an insight into his work situation: programming at night, the switch from SAIL to Pascal, finishing the last T_EX version of 1982 at December 31, 23:59.

Back to chapters 5 and 6, where the reaction of the scientific community is representatively outlined. Jon Bentley picks up literate programming in his regular column in the *Communications of the ACM*, *Programming Pearls: Sampling* (1986). (Eventually, this led to the establishment of an actual *Literate Programming* column.) In *Program-*

ming Pearls: Common Words (1986) Knuth presents a WEB solution to a problem posed by Bentley; a review by Malcolm McIlroy follows.

Chapter 12 finally presents Knuth's actual interest in literate programming by giving a programming example in CWEB, the tool he has also chosen for presenting combinatorial algorithms in his newest book *The Stanford Graphbase*. (This book will be a base for the next volume of his major work, the series *The Art of Computer Programming*.)

Review

The collection enlists pieces of work on the topic without “glue”, except for a few remarks that relate to the origin of the articles. The reader himself is in charge of finding the relation between them. But especially the first article gives a clue to Knuth's motivation behind all technical aspects: He, now professor of *The Art of Computer Programming*, has always tried to make programming an art.

[P]rogramming can give us both intellectual and emotional satisfaction, because it is a real achievement to master complexity and to establish a system of consistent rules. [...] My claim is that it is possible to write *grand* programs, *noble* programs, truly *magnificent* ones!

Starting with the metaphor of programming as an analogy to music composition, he later recognized that it's more like writing literature. “The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style.”

The anthology reveals that Knuth always made literate programming speak for itself. In his articles, it is always presented by examples, connected to the tool WEB, based on Pascal (later C with CWEB).

The principles of literate programming,

1. integrating informal and formal expressions of the same thing, combining explanation and program code into one document,
2. presenting the software according to the solution's semantic structure, keeping the design method visible until the implementation is finished; yielding — together with the human explanation focus — traceability of the development process,
3. concurrent, independent abstraction hierarchies for informal and formal parts, i.e., sections for documentation and refinements for code,
4. linking definition and usage of entities; in WEB by the form of cross references, index, table of contents, etc.,

5. using modern presentation techniques; including, but not limited to, typography, graphics, formulas, tables, etc.,

have to be recognized by ourselves; they are partly obscured by the concrete details of the used examples and tools.

Free distribution of all required tools and integration with T_EX made Knuth's way *the* choice to access literate programming. For a long time, literate programming was totally determined by that orientation, both by concept and problems addressed, but also concerning the tools used. This has influenced at first hand also the reception of literate programming, shown prototypically by the review of McIlroy in chapter 6. Literate programming is taken to be synonymous with WEB, the presented programs are attacked for being monolithic and not reusing other modules — caused in fact by the base language Pascal.

In the meantime, more and more people use literate programming not for the creation of academic solutions to small problems, but for their day-to-day work instead. The discussion forum, a USEnet newsgroup and electronic mailing list, shows that literate programming really starts to be a paradigm in the sense of Thomas Kuhn: A new generation starts to use the principles without caring if it's fully accepted in the traditional development process. As Norman Ramsey put it once, it's the time of the "true believers".

The book doesn't go beyond the starting period of literate programming. Neither does it give a reflection on the paradigm itself, isolated from its own development. Nevertheless, Knuth calls for a second generation of work on literate programming. In the comprehensive bibliography, he lists current work of those that follow his direction, but also of those that go different ways: Extension of literate programming to development of large software systems and to the whole software development process is addressed, printed publication is substituted by electronic documents, and different programming language concepts are taken into account as well as separating literate programming from fixed target languages.

Conclusion

As we expect from an anthology, no new material is presented. The book provides a collection of texts that might not have been very accessible to people outside of universities. This gives the chance to gain a clear understanding how Knuth developed the literate programming paradigm from first requirements to its realization, built upon his ideas for structured programming. If you are interested in such a time-spanning view on scientific work, be it for delight only or for interest in the topic itself, this book is a *must*.

But beyond the formation of a paradigm, this book also shows something rare: It provides insights into the thoughts and working of one of the most influential computer scientists of this century — a man who does not only want to gain knowledge, but wants to share it, wants to make it understandable and accessible. This is so important for him that he was willing to spend years of his work on projects for realizing his ideas, and he has created something qualitatively new with a potential not yet fully exploited.

LET'S GO FORTH now and create *masterpieces of the literate programming art!*

- ◇ Christine Detig
Technical University of Darmstadt
WG Systems Programming
Alexanderstraße 10
D-64283 Darmstadt
Germany
detig@iti.informatik.th-darmstadt.de
- ◇ Joachim Schrod
Technical University of Darmstadt
WG Systems Programming
Alexanderstraße 10
D-64283 Darmstadt
Germany
schrod@iti.informatik.th-darmstadt.de

Tutorials

Output routines: Examples and techniques Part IV: Horizontal techniques

David Salomon

Note on notation: The logo OTR stands for ‘output routine’, and MVL, for ‘Main Vertical List’.

Abstract. The Output Routines series started in 1990 with three articles. The first is an introduction; the second discusses communications techniques; the third is on insertions. The current article is the result of research efforts for the last three years. It discusses advanced techniques for communicating with the OTR from horizontal mode, making it possible to solve problems that require a detailed knowledge of the contents of the lines of text on the page. Logically, this article should be the third in the series, so new readers are advised to read the first two parts, then this part, and finally the part on insertions.

Also, part II should now be called “Vertical Techniques”, instead of “Examples and Techniques”.

Introduction

Certain typesetting problems can only be handled by the OTR. Many times, such a problem is solved by *communicating with the OTR*. Ref. 1 discusses the details and shows examples, but here is a short recap. Certain clues (such as a small piece of glue or kern, or a box with small dimensions) are left in the document, normally by means of the primitive `\vadjust` (Ref. 1). The OTR searches `\box255` for clues and, on finding them, modifies the document in the desired way *in the vicinity of the clue*.

Searching `\box255` is done by breaking it up into its components, and checking each to see if it is a clue. A component can be a line of text, interline glue, vertical kern, or anything else that can go into a vertical list. The breakup is done by means of the `\lastxx` commands.

The problem with this technique is that the clues can only be placed *between* lines of text, and not *inside* a line. We thus say that it is possible to communicate with the OTR from vertical mode, but not from horizontal mode. The reason for this is that a line of text is a box made up of characters of text, and a character is not the same as a box. Specifically, the `\lastbox` command does

not recognize a character of text as a box. The following tests are recommended for inexperienced readers:

```
\setbox0=\hbox{ABC}
\unhbox0 \setbox1=\lastbox
\showbox1
\bye

\setbox0=\hbox{AB\hbox{C}}
\unhbox0 \setbox1=\lastbox
\showbox1
\bye
```

The first test above shows `\box1` to be void, and typesets ‘ABC’. In contrast, the second test shows `\box1` to consist of an `\hbox` with the ‘C’, and typesets only ‘AB’. Ref. 1, p. 217 contains a more detailed discussion of this point.

Communicating with the OTR from horizontal mode is, however, very desirable, since many OTR problems can easily be solved this way. Three methods to do this have consequently been developed, and are described here, each followed by an application to a practical problem. Note that each method has its own limitations, and none is completely general.

The main idea in methods 1 and 2 is to enclose each character of text, as it is being read from the input file, in a box. Now there are no longer any characters, just many small boxes. When the OTR is invoked, each line of text is a box containing other boxes (and glue, kern and penalties) but no characters. The `\lastbox` command can now be used to break up the line of text into its components and search for clues. (To simplify the discussion, we assume text without any math, rules, marks or whatsits.)

Before discussing the details of the first two methods, their disadvantages should be mentioned. Since we no longer have any characters, just boxes, we lose hyphenation, kerning and ligatures. As a result, we normally have to use `\raggedright`, so these methods can only be used in cases where a ragged right margin, and lower typesetting quality, are acceptable.

How can we coerce \TeX to place each character, as it is being input, in a box? Here are the principles of the first two methods:

Method 1. Declare every character of text active, and define it to be itself in a box. Thus we say `\catcode‘\a=13’` followed by `\def a{\hbox{a}}` and repeat for all characters. (The simple definition above cannot be used, because it is infinitely recursive. See below for how it is really done.)

The main disadvantage of this method is that no macros can be embedded in the text. Something like `\abc` will be interpreted as the control sequence ‘`a`’ followed by the non-letters ‘`b`’ and ‘`c`’.

Method 2. Use `\everypar` (and also redefine `\par`) to collect an entire paragraph of text in `\toks0`. Scan `\toks0` token by token and place each non-space token in a small `\hbox`. Then typeset the paragraph. This method does not have the disadvantage of the previous one, since there are no active characters.

Method 3 is completely different. It does not place characters in boxes, and does not use `\lastbox` to break up a line of text. Instead it writes `\box255` on a file, item by item, then reads it back, looking for the clues. This method is slow and tedious, but it does not have the disadvantages of the previous two.

Method 1

We need to declare all the letters, digits and punctuations active (actually, I only did this for the lower case letters, for the uppercase ‘`L`’, for the three digits ‘`123`’, and for ‘`,;`’). Each character should now be defined as a box containing its own character code. Turning ‘`a`’, e.g., into an active character is done by ‘`\catcode‘\a=13 \def a{\hbox{\char‘a}}`’. When we get to the ‘`b`’, however, the command ‘`\catcode‘\b=13 \def b{\hbox{\char‘b}}`’ fails because ‘`a`’ is no longer a letter, so instead of `\catcode`, `TeX` sees `\c` followed by a non-letter. The solution is to use ‘`\let`’ to redefine the control sequences `\catcode`, `\def`, `\hbox` and `\char`. Also the number 13 may cause a problem later, after the digit ‘`1`’ is declared active. The result is declarations such as:

```
\let?=\catcode \let!=\active
\let*=\def \let+=\char \let==\hbox
\let<=\leavevmode \let\=\bye
```

following which, the active characters can be defined by commands such as ‘`\?‘\a! *a{\={\+‘\a}}`’. After this is done, any character of text input by `TeX` is expanded into a box containing that character. Note that `TeX` does not see any text anymore, just a lot of small boxes. This means that there will be nothing to start a paragraph (we will have to place a `\leavevmode` explicitly at the beginning of every paragraph). The following example is a simple application of this technique.

About the examples

All three examples use the following text, that was artificially divided into two paragraphs.

in olden times, when wishing still helped one, there Lived a king whose daughters were aLL beautiful; and the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. cLose by the kings castLe Lay a great dark forest, and under an oLd Lime tree

in the forest was a weLL, and when the day was very warm, the kings child went out into the forest and sat down by the side of the cool fountain; and when she was bored she took a golden baLL, and threw it up on high and caught it; and this baLL was her favorite pLaything.

Example 1. Widening certain letters

This example uses method 1. Before delving into the details of the example, here is the code used to activate characters and to conduct the test:

```
\hsize=3in\tolerance=7500
\raggedright\zeroToSp

\let\?=\catcode \let!=\active
\let*=\def \let+=\char \let==\hbox
\let<=\leavevmode \def\{ \vfill\eject\end}

\?‘\a! \*a{\={\+‘\a}}
\?‘\b! \*b{\={\+‘\b}}
\?‘\c! \*c{\={\+‘\c}}
...
\?‘\x! \*x{\={\+‘\x}}
\?‘\y! \*y{\={\+‘\y}}
\?‘\z! \*z{\={\+‘\z}}
\?‘\L! \*L{\={\+‘\L}}
\?‘\1! \*1{\={\+‘\1}}
\?‘\2! \*2{\={\+‘\2}}
\?‘\3! \*3{\={\+‘\3}}
\?‘\.! \*.{\={\+‘\!}}
\?‘\,\! \*{,\}{\={\+‘\,\}}
\?‘\;\! \*{;\}{\={\+‘\;\}}
```

< in oLden times... Lime tree

< in the forest... favorite pLaything.

\

The example itself is an interesting OTR problem that has recently been communicated to me (Ref. 2), and was the main reason for developing these OTR methods. If one decides, for some reason, not to hyphenate a certain document, then a ragged right margin is a good choice, which makes the text

look best. Certain religious texts, however, don't use hyphenation, and also frown on raggedright. They produce a straight right margin by widening certain letters. In the example below (Figs. 1 & 2) I have selected the 'L', since it's easy to design this letter out of two parts that connect with a rule. I did not actually bother to design a special 'L', and I simply extended it with an `\hrulefill`.

in oLden times, when wishing stiLL
heLped one, there Lived a king whose daughters
were aLL beautifuL; and the youngest was so
beautifuL that the sun itseLf, which has seen so
much, was astonished whenever it shone in her
face. cLose by the kings castLe Lay a great dark
forest, and under an oLd Lime tree

in the forest was a weLL, and when the
day was very warm, the kings chiLd went out
into the forest and sat down by the side of the
cool fountain; and when she was bored she
took a goLden baLL, and threw it up on high
and caught it; and this baLL was her favorite
pLayingthing.

Figure 1

in oL_____den times, when wishing stiL____L____
heLped one, there Lived a king whose daughters
were aL__L_ beautifuL_; and the youngest was so
beautifuL_ that the sun itseLf, which has seen so
much, was astonished whenever it shone in her
face. cLose by the kings castLe Lay a great dark
forest, and under an oL_____d L_____ime tree

in the forest was a weL____L____, and when the
day was very warm, the kings chiL____d went out
into the forest and sat down by the side of the
cool_____ fountain; and when she was bored she
took a goL__den baL__L__, and threw it up on high
and caught it; and this baL__L__ was her favorite
pL_____ayingthing.

Figure 2

When I started thinking about this problem, it was clear to me that this was an OTR problem, and I tentatively outlined the following steps to the solution:

1. Typeset the text with `\raggedright`. This makes the interword glue rigid, and the `\rightskip` glue flexible. Each line of text is placed in an '`\hbox to \hspace`', and `\rightskip` is stretched as necessary.

2. In the OTR, break `\box255` up into individual lines of text. For each line, perform steps 3 through 6.

3. Perform an `\unhbox` on the line, to return `\rightskip` to its natural size (zero). Subtract the present width of the line from its original width (`\hspace`). The difference is the amount by which all the L's on the line will have to be stretched.

4. Break the line up into individual components (mostly characters, glue, and penalties), and count the number of L's in the line.

5. Divide the difference from step 3 by the number of L's from step 4. The result is the amount by which each L will have to be widened.

6. Break the line up again, widening each L. Pack the line in a new `\hbox`.

7. Rebuild the page from the line boxes generated in 6, and ship it out.

The only problem was step 4. A line of text cannot normally be broken up into individual characters and examined. However, using method 1, it is possible to break up such a line, since it does not include any characters, and search for clues. A clue, in our case, is a box whose width is the same as that of an 'L' (if other characters happen to have the same width, the width of the 'L' can be changed by 1sp).

The seven steps above can now be implemented, using the breakup technique (Ref. 1, p. 214).

Step 1. Just say `\raggedright`.

Steps 2 and 7. The OTR becomes

```
\newbox\brk
\output={\setbox\finPage=\vbox{}}%
\setbox\brk=\vbox{\unvbox255 \breakup}%
\ifdim\ht\brk>0pt
\message{Incomplete breakup,
\the\ht\brk}\fi
\shipout\box\finPage \advancepageno}

\newif\ifAnyleft \newcount\pen
\newbox\finPage
\def\breakup{%
\loop \Anyleftfalse
\ifdim\lastskip=0pt
\else \Anylefttrue
\skip0=\lastskip \unskip
\global\setbox\finPage
=\vbox{\vskip\skip0 \unvbox\finPage}%
\fi
\ifdim\lastkern=0pt
\else \Anylefttrue
\dimen0=\lastkern \unkern
\global\setbox\finPage
```



```

    =\vbox{\kern\dimen0 \unvbox\finPage}%
\fi
\ifnum\lastpenalty=0
\else\Anylefttrue
\pen=\lastpenalty \unpenalty
\global\setbox\finPage
    =\vbox{\penalty\pen \unvbox\finPage}%
\fi
\setbox0=\lastbox
\ifvoid0 \else
\Anylefttrue\message{.}\breakupline
\global\setbox\finPage
    =\vbox{\box2 \unvbox\finPage}%
\fi
\ifAnyleft
\repeat}

```

Macro `\breakup` is essentially the same as in Ref. 1. It places each line of text in `\box0`, and expands `\breakupline`. Note the lines with `\global\setbox\finPage=...` They rebuild the page, line by line, in `\box\finPage` (step 7). When the entire process is complete, the OTR ships out `\box\finPage`.

Steps 3 and 5. Macro `\breakupline` resets the line of text to its natural width, calculates the width difference in `\diff`, expands `\countLonline` to count the number of L's in the line, divides `\diff` by that number, and expands `\longLline` to actually widen the L's in the line.

```

\newdimen\diff \newcount\Lnum
\def\breakupline{\diff=\hsize
\setbox0=\hbox{\unhbox0}
\advance\diff-\wd0
\Lnum=0
\setbox1=\hbox{\unhcopy0 \countLonline}
\ifdim\wd1>Opt \message{%
  Incomplete line breakup}\fi
\ifnum\Lnum=0 \diff=Opt
\else \divide\diff by\Lnum
\fi
\setbox2=\null
\setbox1=\hbox{\unhbox0 \longLline}}

```

Step 4. Macro `\countLonline` breaks the line up into individual components and counts the number of L's in the line. It uses a breakup loop similar to the one in `\breakup` above. Note how boxes with 'L' are identified by their width.

```

\newif\ifCharleft
\def\countLonline{%
\Charleftfalse
\ifdim\lastskip=Opt\else \Charlefttrue
\skip0=\lastskip \unskip\fi
\ifdim\lastkern=Opt\else \Charlefttrue

```

```

\dimen0=\lastkern \unkern\fi
\ifnum\lastpenalty=0 \else\Charlefttrue
\pen=\lastpenalty \unpenalty\fi
\setbox1=\lastbox
\ifvoid1\else
\ifdim\wd1=6.25002pt
\global\advance\Lnum1
\fi
\Charlefttrue
\fi
\ifCharleft \countLonline\fi}

```

Step 6. Macro `\longLline` uses the same technique to break the line up again, extend all the 'L's, and rebuild it in `\box2`. Macro `\extendL` packs an 'L' with an `\hrulefill` in a new `\hbox`.

```

\newif\ifSomeleft
\def\longLline{%
\Someleftfalse
\ifdim\lastskip=Opt\else \Somelefttrue
\skip0=\lastskip \unskip \global\setbox2
    =\hbox{\hskip\skip0 \unhbox2}\fi
\ifdim\lastkern=Opt\else \Somelefttrue
\dimen0=\lastkern \unkern \global\setbox2
    =\hbox{\kern\dimen0 \unhbox2}\fi
\ifnum\lastpenalty=0 \else\Somelefttrue
\pen=\lastpenalty \unpenalty \global
\setbox2=\hbox{\penalty\pen \unhbox2}\fi
\setbox1=\lastbox
\ifvoid1\else
\ifdim\wd1=6.25002pt \extendL\fi
\setbox2=\hbox{\box1 \unhbox2}%
\global\Somelefttrue
\fi
\ifSomeleft \longLline\fi}

```

```

\newdimen\Lwidth
\def\extendL{%
\Lwidth=\wd1 \advance\Lwidth by\diff
\setbox1=
\hbox to\Lwidth{\unhbox1\hrulefill}}

```

The code is somewhat long, but is well structured, and most macros use the same breakup technique.

Problems. 1. A line of text without L's is not extended, so it normally comes out shorter.

2. Since there are no letters in our texts, just boxes, there is nothing to signify the start of a paragraph. Each paragraph must therefore start with a `\leavevmode` command (`\<` in our case).

3. `\box255` may only contain boxes, glue, kern and penalties. Anything else (such as text, rules, whatsits or marks) would stop the breakup macros.

Note that overfull lines contain rules, so they should be avoided (by increasing the tolerance, increasing the stretch of `\rightskip`, or by rewriting the text).

4. Because of reasons discussed in Ref. 1, glues with a natural size of 0pt stop the breakup macros. Macro `\zeroToSp` below changes the natural size of several such glues to 1sp. It also changes the plain values of some common penalties from 0 to 1. This macro should be expanded once, at the start of the document.

```
\def\zeroToSp{\parskip=1sp plus1pt
\parfillskip=1sp plus1fil
\advance\leftskip by1sp
\advance\rightskip by1sp
\def\vfil{\vskip1sp plus1fil} %
\def\vfill{\vskip1sp plus1fill}%
\abovedisplayshortskip=1sp plus3pt
\postdisplaypenalty=1
\interlinepenalty=1}
```

5. To identify boxes with an 'L', we use the width of an 'L' in font `cmr10`. To guarantee reliable identification, no other character in the font should have the same width.

Possible improvements and applications. 1. If `\diff` is less than `\hfuzz` (or some other small parameter) it can be set to zero, since there is no point in widening a letter by a very small amount.

2. The L's on the last line of a paragraph are normally widened a lot. If this is not desirable, the macros can be changed to treat the last line differently.

Method 2

As mentioned earlier, the principle is to collect the text of an entire paragraph in a `\toks` register, then to scan the register token by token, placing each character token in a small `\hbox`. We again lose hyphenation, kerning and ligatures, so we normally have to resort to a ragged right margin. However, we can have control sequences embedded in the text. Care should be taken to identify each control sequence (and its argument) and to expand it, instead of placing it in a box. Here are the macros and the test text.*

```
\hsize=4in \tolerance=7500
\raggedright \zeroToSp
\begingroup
\newif\ifargmn
\everypar{\catcode' =12 \toks0=\bgroup}
```

* Editor's note: This text, used to produce Figures 3 and 4, has been realigned to fit the narrow *TUGboat* measure.

```
\def\par{\catcode' =10 \argmnfalse
\expandafter\Tmp\the\toks0 \end \endgraf}
\def\Tmp#1{\ifx\end#1\def\next{\relax}%
\else
\ifargmn\cs{#1}\argmnfalse
\else
\ifcat\relax\noexpand#1%
\let\cs=#1\argmntrue
\else
\ifnum11=\catcode'#1\hbox{#1}%
\else
\ifnum12=\catcode'#1\hbox{#1}%
\else\ifnum'40='#1\ \fi
\fi
\fi
\fi
\let\next=\Tmp\fi\next}%
%
in\Mnote{xyz *} oLden times, when
\Mnote{abc 2}wishing stiLL
heLped\Mnote{note 3} one, there Lived a
king whose daugh\Mnote{note 4}ters
were aLL beautifuL; and the
youngest\Mnote{note 5} was so beautifuL that
the sun itseLf, which has seen so much, was
\Mnote{note 6}astonished whenever it
shone in her face. cLose by the kings
castLe L\Mnote{note 7}ay a great dark
forest, and under an oLd Lime tree}

in the fore\Mnote{note 8}st was a weLL,
and when the day was\Mnote{note 9} very
warm, the kings child went out into the
fores\Mnote{note 10}t and sat down by the
side of the coo\Mnote{note 20}L fountain;
\Mnote{note 21}and when she was bored
\Mnote{note 22}she took a golden
baLL,\Mnote{note 12} and threw it up on high
and caught it; and this baLL was her favorite
\Mnote{note 13}pLaything.}

\endgroup
\bye
```

The `\everypar` parameter is modified to place `'\toks0=\bgroup'` at the start of each paragraph. At the end of a paragraph we need a closing `\egroup`, which is easy to insert by redefining `\par`. Unfortunately, the command `'\toks0=\bgroup...\egroup'` does not work. Using `\bgroup` is okay, but a right brace (a token of catcode 2) is required, instead of the control sequence `\egroup`. When using this method we unfortunately have to insert a '}' explicitly at the

end of every paragraph. This is one of two unsolved problems with this method.

The `\par` primitive is modified to expand `'\the\toks0'`, to append an `\end` to it, to expand `\Tmp`, and to close the paragraph.

Macro `\Tmp` uses recursion to extract the next token from `\toks0` and to test it. Tokens with catcodes 11 and 12 are placed in boxes and appended to the current list (normally the MVL) (except spaces, which are appended as spaces to the MVL). Control sequence tokens are also identified. Each such token is kept in `\cs` until its argument is identified in the following recursive iteration, where it is expanded. In the current version, any control sequence embedded in the text must have exactly one argument. The changes of `\everypar` and `\par` are confined to a group.

Spaces present a special problem. The scanning of tokens skips all spaces. Therefore, the catcode of space had to be changed. It has been changed to 12 (other) and `\Tmp` identifies spaces by their character code. When a catcode 12 space is identified by `\Tmp`, a normal (catcode 10) space is appended to `\box0`.

The second unsolved problem in this method is the end of lines. They are converted into spaces, but only after the catcode of a space has been changed. As a result, they appear in `\toks0` as normal spaces (catcode 10) and are skipped.

Example 2. Marginal notes

Typesetting notes in the margins of a scholarly book is very common. Ref. 3 is an interesting example, familiar to many \TeX users. Another well known example is the marginal notes of the mathematician Pierre Fermat. When trying to prove the so-called Fermat's last theorem (there is no integer $n > 2$ such that $x^n + y^n = a^n$ for rational x , y and a), he wrote in the margin of the book he was reading (Bachet's *Diophantus*) "I have discovered a truly marvellous demonstration of this general theorem, which this margin is too narrow to contain" (Ref. 4). Unfortunately for us, to this day no one has been able to prove (or find a counterexample to) this theorem.* I like to call this famous note *Fermat's warning*. It warns us not to abuse this useful tool of the author.

* Editor's note: While this article was in production, it was announced that Andrew Wiles of Princeton University had found a proof, then that a gap may exist in the proof; Wiles is continuing work on the paper.

When teaching \TeX I have always noticed how, when discussing marginal notes, the class suddenly comes to life and starts following the discussion with renewed interest. In the lab that follows, people start writing macros for marginal notes, invariably ignoring Fermat's warning, and overdoing this useful feature.

A single note can easily be placed in the margin of a given line with the help of `\vadjust`. When writing a text with many marginal notes, however, the writer may end up with two or more notes appearing on the margin of the same line. Because of the limited space on the margin, the notes for the same line of text may have to be rearranged before the page is shipped out, and this is an OTR problem. Rearranging notes may involve placing some on the left, and some on the right margin; it may mean to typeset them in very small type, to move some up or down (if there is room on adjacent lines), or to warn the author that there is no room.

In this example, rearranging is done in a simple way. The first note found on a line is typeset on the left, the second one, on the right margin. If more notes are found on the same line, none is typeset, and a warning, with the input line number, is placed in the log file.

The implementation is straightforward. Macro `\Mnote` places the text of the note in an `'\hbox to1sp'` inside the paragraph `'\def\Mnote#1{\hbox to1sp{#1\hss}}'`. Method 2 is used to place every character of text in a box. The OTR breaks up `\box255` into its top level components and identifies the lines of text. Each line is further broken up, and all the clues (boxes of width 1sp) in it located. Depending on how many clues were found, the macros place the notes as described above. The OTR is straightforward:

```
\newbox\brk
\output={\setbox\finPage=\vbox{
  \setbox\brk=\vbox{\unvcopy255 \breakup}%
  \ifdim\ht\brk>0pt \message{Incomplete
    breakup, \the\ht\brk}\fi
  \shipout\box255 \shipout\box\finPage}}
```

Note that it also ships out `\box255`, for comparison purposes. Macro `\breakup` rebuilds all the elements of `\box255` in `\box\finPage`, except that each line of text is further broken up by `\breakupline` (and the notes properly placed in the margins) before being rebuilt and appended to `\box\finPage`.

```
\newif\ifAnyleft \newcount\pen
\newbox\finPage
\def\breakup{%
  \loop \Anyleftfalse
```

```

\ifdim\lastskip=0pt
\else
  \Anylefttrue \skip0=\lastskip \unskip
  \global\setbox\finPage
    =\vbox{\vskip\skip0 \unvbox\finPage}%
\fi
\ifdim\lastkern=0pt
\else
  \Anylefttrue \dimen0=\lastkern \unkern
  \global\setbox\finPage
    =\vbox{\kern\dimen0 \unvbox\finPage}%
\fi
\ifnum\lastpenalty=0
\else \Anylefttrue
  \pen=\lastpenalty \unpenalty
  \global\setbox\finPage
    =\vbox{\penalty\pen \unvbox\finPage}%
\fi
\setbox0=\lastbox
\ifvoid0
\else \Anylefttrue\message{.}%
  \breakupline
  \global\setbox\finPage
    =\vbox{\box2 \unvbox\finPage}%
\fi
\ifAnyleft
\repeat}

```

Macro `\breakupline` expands `\countNotesonline` to break up one line of text, and count the number of notes. It then rebuilds the line in `\box2` with the notes placed in the margins, and with the special boxes emptied.

```

\newcount\numnotes
\def\breakupline{\numnotes=0
\setbox1=\hbox{\unhbox0 \countNotesonline}%
\ifdim\wd1>0pt \message{%
  Incomplete line breakup}\fi
\ifcase\numnotes
\relax % \numnotes=0 -> 0 notes on this line
\or % 1 note
  \setbox2=\hbox to\hsize{%
    \llap{\box3\kern3pt}\unhbox2\hfil}%
\else % 2 or more notes
  \setbox2=\hbox to\hsize{%
    \llap{\box4\kern3pt}\unhbox2\hfil
    \rlap{\kern3pt\box3}}%
\fi}

```

Macro `\countNotesonline` is a simple application of the breakup technique for one line of text. The first note found in the line is placed in `\box3`, and the second one, in `\box4`. All subsequent notes are flushed. A small dash is inserted in each special box to show where the note came from.

```

\newif\ifCharleft
\def\countNotesonline{%
  \Charleftfalse
  \ifdim\lastskip=0pt
  \else \Charlefttrue
    \skip0=\lastskip \unskip
    \global\setbox2
      =\hbox{\hskip\skip0 \unhbox2}%
  \fi
  \ifdim\lastkern=0pt
  \else \Charlefttrue
    \dimen0=\lastkern \unkern
    \global\setbox2
      =\hbox{\kern\dimen0 \unhbox2}%
  \fi
  \ifnum\lastpenalty=0
  \else \Charlefttrue
    \pen=\lastpenalty \unpenalty
    \global\setbox2=
      \hbox{\penalty\pen \unhbox2}%
  \fi
  \setbox1=\lastbox
  \ifvoid1\else
  \ifdim\wd1=1sp % a special box
  \ifnum\numnotes=0
    \global\setbox3=\hbox{\unhbox1}%
  \fi
  \ifnum\numnotes=1
    \global\setbox4=\hbox{\unhbox1}%
  \fi
  \ifnum\numnotes>1
    \global\setbox3=\hbox{!!!}%
    \global\setbox4=\hbox{!!!}%
    \message{Too many notes on line
      \the\inputlineno}%
  \fi
  \global\advance\numnotes 1
  \global\setbox2=
    \hbox{\pop\unhbox2}%
  \else % not a special box
  \global\setbox2=\hbox{\box1 \unhbox2}%
  \fi
  \Charlefttrue
  \fi
\ifCharleft \countNotesonline\fi}

```

Finally, macro `\pop` places a small dash in the special box after it has been emptied. The dash is character "37 of font `cmsy` (the math symbols). This character is constructed in a box of width 0 and it sticks out on the right. Normally it is followed by a minus or a right arrow, to create a "maps to" symbol (Ref. 5, p. 515).

```

\def\pop{\leavevmode\raise4pt\hbox to1sp
{\hss\smash{\tensy\char"37}\kern1.2pt\hss}}

```

Tests

The two paragraphs used for the test were shown earlier. The first diagram (Fig. 3) shows `\box255` before any changes. Note how the text of the notes overlap the text of the paragraphs, since they are saved in boxes *inside the paragraph*. The diagram in Fig. 4 shows the final result shipped out.

In practical use, sophisticated macros can be developed that will set the notes in small type, will number them consecutively, and will move them vertically, if necessary. However, as long as they are based on the principles shown here, raggedright will normally have to be used, which is not always acceptable.

Method 3

This method is based on a two-pass job. In the first pass the text is typeset in the normal way, with characters, not boxes. Clues are inserted in the text, to be found later, by the OTR, in pass 2. Pages can either be shipped out or trashed, but the OTR writes `\box255` on a file, to be read by pass 2. Advanced users know that a box cannot be written on a file in the usual way, using `\write`. The novelty of this method is that a box can be written on the *log file*, using `\showbox`.

The user has to make sure that the log file is saved after pass 1. Pass 2 reads the contents of `\box255` from the file, searches for the beginning of each line of text, and for clues inside the line.

in¹ oLden times, when ²wishing stiLLheLped³ one, there Lived
 a king whose daught⁴ers were aLL beautifuL; and the youngest⁵
 was so beautifuL that the sun itseLf, which has seen so much,
 was ⁶astor⁶ished whenever it shone in her face. cLose by the kings
 castLe ⁷Lay⁷ a great darkforest, and under an oLd Lime tree
 in the forest ⁸st⁸ was a weLL, and when the day was ⁹not⁹ very
 warm, the kings chiLd went out into the forest ¹⁰and sat down by the
 side of the cool ¹¹fountain; ¹²and when she was bored ¹³she took a goLden
 baLL, ¹⁴and threw it up on high and caught it; and this baLL was
 her favorite ¹⁵pl¹⁵aying thing.

Figure 3

!!! in¹ oLden times, when ²wishing stiLLheLped³ one, there Lived !!!
 note-4 a king whose daught⁴ers were aLL beautifuL; and the youngest⁵ note-5
 was so beautifuL that the sun itseLf, which has seen so much,
 note-6 was ⁶astor⁶ished whenever it shone in her face. cLose by the kings
 note-7 castLe ⁷Lay⁷ a great darkforest, and under an oLd Lime tree
 note-8 in the forest ⁸st⁸ was a weLL, and when the day was ⁹not⁹ verywarm, note-9
 note-10 the kings chiLd went out into the forest¹⁰ and sat down by the
 side of the cool ¹¹fountain; ¹²and when she was bored¹² she took a goLden !!!
 note-12 baLL, ¹³and threw it up on high and caught it; and this baLL was
 note-13 her favorite ¹⁴pl¹⁴aying thing.

Figure 4

If successful, pass 2 knows what clues are stored in each text line. Pass 2 then reads the source file, typesets it in the usual way and has the OTR modify `\box255`, before shipping it out, according to the clues read earlier.

Note that `\lastbox` is not used. The details of each line of text in `\box255` are read from the file. The main advantage of this approach is that none of the high quality typesetting features, such as hyphenation, kerning and ligatures, is lost.

The main problem with this approach is how to read and analyse the contents of `\box255` from the log file in pass 2 (an example of such a file is shown below for the benefit of inexperienced readers). This turns out to be easy, and it involves the following tasks:

1. Certain records contain backslashes that should be ignored. Examples are: `'..\tenrm i'`, `'.\glue(\topskip) 3.05556'` and `'..\glue 3.33333 plus 1.66666 minus 1.11111'`. To ignore these, pass 2 uses the following declarations (inside a group):

```
\let\vbox=\relax \let\glue=\relax
\let\topskip=\relax \let\kern=\relax
\let\rightskip=\relax
\let\baselineskip=\relax
\let\parfillskip=\relax
\let\parskip=\relax
\def\shipout\box{\bgroup}%
\let\showbox=\egroup
\let\discretionary=\relax
```

2. Other records are important and should be identified. Examples are:

- a. `'> \box255='` (this signals the start of the box)
- b. `'.\hbox(6.94444+1.94444)x216.81, glue set 0.45114'` (this signals a new line of text).
- b. `'..\hbox(0.0+0.0)x0.00002, glue set ...'` (this is a box of width 1sp, denoted a clue of type 1).
- c. `'! OK (see the transcript file).'` (this signals the end of the box).

Records of type a are identified by defining `\def\box255={\global\clues={}}{}`. The definition of `\box255` is changed (locally) to insert a '(' in the toks register `\clues`.

Records of type b are identified by redefining `\hbox`.

```
\def\hbox(#1)x#2 {\toks0={}}\one#2\end
\tmp=\the\toks0 pt
\ifnum\tmp=1sp\appendclue1
\else
\ifnum\tmp=2sp\appendclue2
```

```
\else
\ifnum\tmp=\Hsize\appendclue+
\fi\fi\fi}
\def\one#1{\def\arg{#1}%
\ifx\end#1\let\rep=\relax
\else\ifx\comma\arg\let\rep=\one
\else\toks0=\expandafter{\the\toks0 #1}%
\let\rep=\one
\fi\fi\rep}
\def\appendclue#1{\global\clues=%
\expandafter{\expandafter#1\the\clues}}
```

Parameter '#2' is the width of the `\hbox`. In the records that interest us, it is either `\hsize` or `1sp` or `2sp`. The examples in b above show that the width is followed by a comma and a space, but there are records on the log file (such as the paragraph indentation `'..\hbox(0.0+0.0)x20.0'`) where the width is followed by a space. This is why '#2' in the definition of `\hbox` is delimited by a space. If the width is followed by a comma it (the comma) is removed by macro `\one`. The width is stored in the `\dimen` register `\tmp`.

Macro `\hsize` thus identifies the important records, and appends the tokens '+', '1' or '2' to the toks register `\clues` every time a line of text, or a clue of type 1 or type 2, respectively, is found.

The end of the box in the log file is identified when a type c record is found. We simply compare each record read to the string `'! OK (see the transcript file).'`. When finding it, a '(' is appended to `\clues`, and the loop reading the file is stopped. Note that our macros are supposed to stop reading when the end of box is found. They are never supposed to read the end of file. If an end of file is sensed while reading the log file, an error must have occurred.

All the clues found in the log file for one page (a single `\box255`) are stored in the toks register `\clues`, so that later macros can easily find out what clues were found in what text lines. A simple example is the tokens `')21++2+++121+('` where the ')' and '(' stand, respectively, for the end and start of `\box255` in the log file, each '+' stands for a line of text, and each '1' or '2', for a clue of type 1 or 2 found in that line. Thus in the above example, a type 2 followed by a type 1 clue were found in the bottom line, another type 2 clue, in line 3 from the bottom, and three more clues, in line 6 (the top line).

Pass 1 normally writes several boxes on the log file, each corresponding to a page. The following appears in the log file between pages, and has to be 'neutralized'.

```
<output> {\showbox 255
          \shipout \box 255}
```

This is done by the weird definitions ‘\def\shipout\box{\bgroup}’ and ‘\let\showbox=\egroup’. The method is illustrated below by applying it to a practical example.

Example 3. Revision bars

Certain documents, such as the bylaws of an organization, go through periodic revisions. It is good practice to typeset each new revision with vertical bars on the left of parts that have been revised. This is an OTR problem (note that a revision may be broken across pages), and the solution shown here requires the two passes mentioned above. Pass 1 involves:

- Two macros are defined, to indicate the start and end of each revision.

```
\def\({\leavevmode\raise4pt\hbox to1sp{%
  \hss\smash{\tensy\char"37}\kern1.2pt\hss}}
\def\){\leavevmode\raise4pt\hbox to2sp{%
  \hss\smash{\tensy\char"37}\kern1.2pt\hss}}
```

The macros also place small dashes in the text, to indicate the boundaries of the revision.

- The OTR writes \box255 on the log file, and can also ship it out, for later comparison. If a shipout is not required, the OTR can say \box255=\null \deadcycles=0 instead of \shipout\box255.

```
\hsize=3in \vsize=2.2in \tolerance=7500
\showboxbreadth=1000 \showboxdepth=10
\output={\showbox255 \shipout\box255
  \advancepageno}
```

```
\input source
\vfill\eject
```

The log file is saved between the passes. Note that the two passes can be parts of the same T_EX job, and the log file can be saved when T_EX stops, as usual, for a user’s response, after the \showbox. Pass 2 starts by opening the log file, if it exists:

```
\newread\logfile
\newtoks\clues \newdimen\tmp
\newif\ifmore \moretrue
\newdimen\Hsize \Hsize=\hsize
\newbox\brk \newbox\bars
\newif\ifendRev \newif\ifbegRev
\newif\ifRev \newif\ifSplitrev

\immediate\openin\logfile=Log
\ifeof\logfile\errmessage{No log file}\fi
```

Note that the file name ‘Log’ is used here. In the general case, it is possible to read the name from the keyboard. Now comes the OTR. It is divided into two phases. Phase 1 reads a chunk off the log file, corresponding to one page, and prepares tokens in \clues. Phase 2 starts the breakup of \box255, and ships out \box\bars (stretched to \vsize) and \box255, side by side.

```
\output={%
% Phase 1. Read a chunk off the log file
% and prepare codes in \clues
\begingroup
\def\appendclue#1{\global\clues=%
  \expandafter{\expandafter#1\the\clues}}
\def\OK{! OK (see the transcript file). }
\def\comma{,}
\def\box255={\global\clues={}}
\def\hbox(#1)x#2 {\toks0={}\one#2\end
  \tmp=\the\toks0 pt
  \ifnum\tmp=1sp\appendclue1
  \else \ifnum\tmp=2sp\appendclue2
  \else \ifnum\tmp=\Hsize\appendclue+
  \fi\fi\fi}
\def\one#1{\def\arg{#1}%
  \ifx\end#1\let\rep=\relax
  \else\ifx\comma\arg\let\rep=\one
  \else\toks0=\expandafter{\the\toks0 #1}%
  \let\rep=\one
  \fi\fi\rep}
%
\let\vbox=\relax \let\glue=\relax
\let\topskip=\relax \let\kern=\relax
\let\rightskip=\relax
\let\baselineskip=\relax
\let\parfillskip=\relax
\let\parskip=\relax
\def\shipout\box{\bgroup}
\let\showbox=\egroup
\let\discretionary=\relax
\setbox0=\vtop{\hsize=\maxdimen
\loop
  \read\logfile to\rec
  \ifeof\logfile\morefalse
  \message{end of log file!}
  \else
  \ifx\rec\OK\appendclue)\morefalse\fi
  \rec
  \fi
  \ifmore
  \repeat}
\endgroup
\nextclue
\if)\clue \else\message{Bad clue}\fi
```

```
% Phase 2.
% Breakup \box255 and use the clues
\global\setbox\bars=\vbox{}%
\global\endRevfalse \global\beginRevfalse
\global\Revfalse
\setbox\brk=\vbox{\unvcopy255 \breakup}%
\ifdim\ht\brk>0pt \message{%
  Incomplete breakup, \the\ht\brk}\fi
\shipout\hbox{\vbox to\vsizelength{\unvbox\bars}%
  \kern4pt\box255} \advancepageno}
```

3. Macro `\breakup` breaks up `\copy255` into its top level components. For each component with a dimension, the macro places either a skip or a `\vrule` in `\box\bars`. It is important to realize that when we say, e.g., `\skip0=\lastskip` we lose the specific glue set ratio of `\box255`. This is why the rules are placed in `\box\bars` using `\leaders` and not `\vrule`. This way `\box\bars` can later be stretched to `\vsizelength`, and all the leaders in it will be stretched.

Exercise: Why is it that a rule placed by means of `\vrule height\skip0` cannot be stretched later?

Answer: Because the command `\vrule` is supposed to be followed by a `height<dimen>`. If we use glue, such as `\skip0`, only the natural size is used, and the stretch and shrink components are ignored.

```
\newif\ifAnyleft \newcount\pen
\def\breakup{%
  \loop \Anyleftfalse
  \ifdim\lastskip=0pt
  \else \Anylefttrue
  \skip0=\lastskip \unskip
  \global\setbox\bars=\vbox{\ifRev\leaders
    \vrule\fi\vskip\skip0\unvbox\bars}%
  \fi
  \ifdim\lastkern=0pt
  \else \Anylefttrue
  \dimen0=\lastkern \unkern
  \global\setbox\bars=\vbox{\ifRev\leaders
    \vrule\fi\kern\dimen0\unvbox\bars}%
  \fi
  \ifnum\lastpenalty=0
  \else \Anylefttrue
  \pen=\lastpenalty \unpenalty
  \fi
  \setbox0=\lastbox
  \ifvoid0 \else \Anylefttrue
  \dimen0=\ht0 \advance\dimen0 by\dp0
  \setbox2=\vbox{\unhbox0 \searchclues}%
  \ifbeginRev
  \ifendRev
%TT
  \global\Revfalse \global\endRevfalse
```

```
\global\setbox\bars=\vbox{\leaders
  \vrule\vskip\dimen0\unvbox\bars}%
\else
%TF
\global\Revfalse
\ifSplitrev \global\Splitrevfalse
\global\setbox\bars=\vbox{\leaders
  \vrule\vskip\ht\bars}%
\global\setbox\bars=\vbox{\leaders
  \vrule\vskip\dimen0\unvbox\bars}%
\else
\global\setbox\bars
  =\vbox{\vskip\dimen0\unvbox\bars}%
\fi\fi
\else
\ifendRev
%FT
\global\Revtrue
\global\setbox\bars=\vbox{\leaders
  \vrule\vskip\dimen0\unvbox\bars}%
\else
%FF
\global\Revfalse
\global\setbox\bars
  =\vbox{\vskip\dimen0\unvbox\bars}
\fi\fi\fi
\ifAnyleft
\repeat}
```

When a line of text is found, `\searchclues` is expanded (see below), to update variables `\beginRev` and `\endRev`. Four cases are possible:

a. Both variables are false (case FF above). This means no revisions have been found yet. A skip, equal in height to the current line of text, is appended to `\box\bars`. Variable `\Rev` is set to false, indicating that any future components found in `\box255` should become skips in `\box\bars`.

b. `\beginRev` is false and `\endRev` is true (case FT above), meaning the current line contains the end of a revision. A rule, the height of the current line, is appended to `\box\bars`. Also, `\Rev` is set to true, indicating that any future components found in `\box255` should become rules in `\box\bars`.

c. Case TT. A revision starts on this line. A rule is appended to `\box\bars` but `\Rev` is set to false. (Also `\endRev` is set to false, so case TF will be in effect from now on.)

d. Case TF. Normally this indicates a line with no revisions but, if `\Splitrev` is true, we have just found the start of a revision that will end on the next page. In this case, `\box\bars` (which has only skips in it so far) is filled up with a rule.

Macro `\searchclues` removes the next token from `\clues` and, if it is 1 or 2, sets `\begRev` or `\endRev` to true, respectively. Note that a revision may start and end on the same line. If the start of a revision is found while `\endRev` is false, it means that the revision will end on the next page. In such a case, variable `\Splitrev` is set to true, indicating that the entire `\box\bars` should be filled with a rule.

```
\def\searchclues{\nextclue
  \if+\clue\let\Next=\relax
  \else
    \if(\clue\let\Next=\relax
      \message{bad Clue}
    \else
      \if2\clue \global\endRevtrue
        \global\begRevfalse
        \let\Next=\searchclues
      \else
        \if1\clue \global\begRevtrue
          \let\Next=\searchclues
          \ifendRev
            \else\global\Splitrevtrue\fi
          \else
            \message{bad clue}
          \fi\fi\fi\fi\Next}

\def\nextclue{\expandafter\extr\the\clues X}
\def\extr#1#2X{\gdef\clue{#1}%
  \global\clues=\expandafter{#2}}

The rest of pass 2 is straightforward.

\zeroToSp
\input source
\bye
```

For a multi-page document, the OTR performs the same tasks for each page. It first receives `\box255` of page 1. It reads the corresponding lines from the log file, looking for clues and storing them in `\clues`. The OTR then breaks `\box255` up, isolating the lines of text from the bottom. It uses the tokens in `\clues` to modify only the right lines. At the end, `\box255` (and `\box\bars`) are shipped out. The process repeats for each successive page sent to the OTR.

For each page, the OTR reads another chunk off the log file. This is why the two passes must typeset the same text. The best way to handle this is to `\input` the text in the two passes from the same source file. It is possible to make the macros more robust by checking to see, in pass 2, that the chunk read from the log file actually has the same number of text lines as the current `\box255`.

The source file for our test is, as usual:

in oLden times, when wishing stiLL heLped one, there Lived a king whose daughters were aLL beautifuL; and th\ (e youngest was so beautifuL that the sun itseLf, which has seen so much, was astonished whenever it shone in her face. cLose by the kings castLe Lay a great dark forest, and under an oLd Lime tree

in the forest\) was a weLL, and when the day was very warm, the kings chiLd went out into the forest and sat down\ (by the side of the cooL fountain; and when she was bored she took a \)goLden baLL, and threw it up on high and caught it; and this baLL was her favorite pLaything.

Following are the final result and parts of the log file produced by pass 1.

in oLden times, when wishing stiLL heLped one, there Lived a king whose daughters were aLL beautifuL; and the youngest was so beautifuL that the sun itseLf, which has seen so much, was astonished whenever it shone in her face. cLose by the kings castLe Lay a great dark forest, and under an oLd Lime tree

in the forest' was a weLL, and when the day was very warm, the kings chiLd went out into the forest and sat down' by the side of the cooL fountain; and when she was bored she took a 'goLden baLL, and threw it up on high and caught it; and this baLL was her favorite pLaything.

Figure 5

```
Textures 1.5 (preloaded format=plain 92.6.1)
  21 OCT 1992 17:54
(test (source)
> \box255=
\ vbox(158.99377+0.0)x216.81, glue set 3.04933fill
.\ glue(\topskip) 3.05556
.\ hbox(6.94444+1.94444)x216.81, glue set 0.45114
..\ hbox(0.0+0.0)x20.0
..\ tenrm i
..\ tenrm n
..\ glue 3.33333 plus 1.66666 minus 1.11111
..\ tenrm o
..\ tenrm L
..\ tenrm d
.....
.....
..\ tenrm a
..\ tenrm n
..\ tenrm d
..\ glue 3.33333 plus 1.66666 minus 1.11111
```

```

..\tenrm t
..\tenrm h
..\hbox(0.0+0.0)x0.00002, glue set - 0.59999fil,
  shifted -6.0
...\glue 0.0 plus 1.0fil minus 1.0fil
...\hbox(0.0+0.0)x0.0
....\tensy 7
...\kern 1.2
...\glue 0.0 plus 1.0fil minus 1.0fil
..\tenrm e
.....
.....
..\tenrm g
..\tenrm .
..\penalty 10000
..\glue(\parfillskip) 0.0 plus 1.0fil
..\glue(\rightskip) 0.0
.\glue 0.0 plus 1.0fill

! OK (see the transcript file).
<output> {\showbox 255
          \shipout \box 255 \advancepageno }
\break ->\penalty -\@M

1.12 \vfill\ejct

?
```

A summary and a wish

The methods described here have limitations and disadvantages, so they cannot be used in every situation. Method 2 still has a few unsolved problems. As a result, the macros described here cannot be canned and used 'as is'. They should be carefully studied and understood, so that they could be applied to practical problems. This means that they are beyond the grasp of beginners but, because of their power, they may provide the necessary incentive to many beginners to become full fledged wizards.

It would be so much easier to solve the three problems discussed here if the `\lastbox` command could recognize characters of text, or if a new command, `\lastchar`, were available for this purpose. This is a private wish that I hope will be shared by readers.

Finally, I would like to thank the many people who have responded to the original OTR articles of 1990. I would like to think that I was able to help some of them, and I know that their comments, questions, and criticism have helped me become more proficient in this fascinating field of OTR techniques.

References

1. Salomon, D., *Output Routines: Examples and Techniques. Part II*, TUGboat 11(2), pp. 212-236, June 1990.
2. Haralambous, Y., Private communication.
3. Graham, R. L., et al., *Concrete Mathematics*, Addison-Wesley, 1989.
4. Bell, E. T., *Men of Mathematics*, Simon and Schuster, 1937.
5. Knuth, D. E., *Computers and Typesetting*, vol. E, Addison-Wesley, 1986.

◇ David Salomon
 California State University,
 Northridge
 Computer Science Department
 Northridge, CA 91330-8281
 dxs@secs.csun.edu

Verbatim Copying and Listing

David Salomon

A general note: Square brackets are used throughout this article to refer to *The TeXbook*. Thus [39] refers to page 39. Also, the logo OTR stands for 'output routine', and MVL, for 'Main Vertical List'.

Introduction

Methods are developed, and macros listed, to solve the following two problems. Verbatim copying is the problem of writing a token string verbatim on a file, then executing it. Verbatim listing involves typesetting a token string verbatim, in either horizontal or vertical mode.

We start with a short review of `\edef`. In '`\edef\abc{\xyz \kern1em}`', the control sequence `\xyz` is expanded immediately (at the time `\abc` is defined), but the `\kern` command is only executed later (when `\abc` is expanded).

The same thing happens when `\abc` is defined by means of `\def`, and is then written on a file. Thus '`\write\aux{\abc}`' writes the replacement text that would have been created by `\edef\abc{...}`.

Sometimes it is desirable to write the name of a control sequence on a file, rather than its expansion. This can be done either by '`\write\aux{\noexpand\abc}`' or, similarly, by '`\write\aux{\string\abc}`'. The former form

writes a space following `\abc`, while the latter one does not.

Verbatim copying

With this in mind we now consider the following problem: given an arbitrary string, containing text, control sequences, active characters, and special characters (such as `#{}`), first write it on a file without expansion (verbatim), then expand it.

Before delving into the details, here are some examples that show that this problem is practical:

1. When writing a textbook with exercises and answers, the author would like to be able to say:

```
\exercise...
\answer...
...
\endanswer
```

and have the answer written verbatim on a file. The file can later be input, to typeset all the answers in an appendix. However, while the book is being written, the author may also want, for proofreading purposes, to typeset the answer right following the exercise. Note that an answer may contain many control sequences, and may be long.

2. When writing a book on \TeX , the author would like to have an active character (say `~`), such that `~{\baselineskip}=24pt` would write `\baselineskip` on a file (perhaps with the page number, for later preparation of an index) and also execute `'\baselineskip=24pt'`.

We develop two approaches to this problem. The first one uses catcode changes to suppress the special meanings of certain characters before the string is read by \TeX . It is then easy to read the string and write it verbatim on a file. However, in order to also expand the string, all characters should have their normal catcodes. This is done by writing the string on another file and reading it back immediately. This way, the string is parsed into tokens that get their normal catcodes, and can later be expanded.

In the second approach no catcodes are modified; the string is input as usual and tokens created. The string is then scanned, token by token, to identify the control sequence tokens. Expansion is avoided either by placing a `\noexpand` in front of each control sequence, or by temporarily redefining each control sequence as `\relax` (which is non-expandable). The string can then be written on a file with nothing expanded. Following which, all control sequences get back their original meanings, and the string can be expanded in the usual way.

My usual disclaimer applies heavily to this material and is therefore repeated: the macros presented here are simple. Each has its limitations, and can be used for certain applications only. The macros should therefore not be copied and used verbatim. They should be carefully studied and fully understood by the reader, so that they could be modified for specific applications.

Approach 1. We present a number of macros, all based on catcode changes. The first two change the catcodes of all the special characters. The other three change the catcodes of just a few characters. In between the two groups, we illustrate how the macros can be modified to handle a specific problem, namely, writing index items, with page numbers, on a file.

Basic verbatim copying: `\VwriteA`. To avoid expansion we change the catcodes of the special characters, such as `'`, `#` and `\`, to 12 (other). This way, the `'` is no longer the escape character, so \TeX does not recognize any control sequences, and there is nothing to expand. The catcode changes should, of course, be done locally, in a group. Macro `\VwriteA` below starts a group, does the catcode changes, and expands `\aux`. Macro `\aux` absorbs the argument, does the `\write`, and closes the group.

```
\def\makeother#1{\catcode'#1=12\relax}
\def\sanitize{\makeother\ \makeother\\
\makeother\$\makeother\&%
\makeother#\makeother\_%
\makeother%\makeother\~\makeother\|}
```

```
\newwrite\out
\immediate\openout\out=filename
```

```
\def\VwriteA{\begingroup\sanitize\aux}
\def\aux#1{\write\out{#1{\folio}}\endgroup}
```

The following is a representative expansion `'\VwriteA{te xt#$$&_~\a\x fin}'`. The argument, which seems to belong to `\VwriteA`, is actually absorbed by `\aux`. Also, since the actual writing is done in the OTR, it is possible to write the page number on the file, even in braces, as above.

Any active characters that may appear in our strings should, of course, also be sanitized. In the example above the vertical bar `|` was sanitized, since we declare it active and use it for verbatim listings. The braces, on the other hand, were not sanitized, which makes it possible to enclose the argument in braces (but then the argument cannot contain arbitrary braces, only balanced ones).

Verbatim copying with braces. Macro `\VwriteB` below is a slightly different version that does sanitize the braces. The argument can now contain arbitrary braces, but it must be delimited by something else (the string `'endP'` in our case).

```
\def\sanitize{\makeother\ \makeother\\%
\makeother\$\makeother\&%
\makeother\#\makeother\_ \makeother\%
\makeother\^ \makeother\|%
\makeother\} \makeother\{ }

\newwrite\out
\immediate\openout\out=filename

\def\VwriteB{\begingroup\sanitize\aux}
\def\aux#1endP{\write\out{#1{\folio}}%
\endgroup}
```

An example. As a practical example, we use `\VwriteA` to illustrate the creation of a raw index file. The following macros declare the `^` an active character to write index items (with page numbers) on a file. They also use `\futurelet` to allow silent index items (items that should be written on the file, but should not be typeset).

Typical uses are `^{\kerning}`, `^[\kern]{}` and `^[B.L.]{User}`. Up to two arguments can be specified, and are written on the index file as one string (with the page number). However, only the main argument (in braces) is typeset. The optional argument, in brackets, is silent.

```
\def\Caret{\ifmmode\def\next{^}%
\else\let\next=\indexT\fi\next}
\catcode'\^=\active \let^=\Caret

\def\indexT{\futurelet\new\macX}
\def\macX{\ifx\new[\let\Next=\inxB%
\else\let\Next=\inxA\fi
\begingroup\sanitize\Next}
\def\inxA#1{#1\writeinx{#1}}
\def\inxB{#1}#2{#2\writeinx{#1#2}}
\def\writeinx#1{\write\inx{%
\string\indexentry{#1}{\folio}}\endgroup}
```

This example is simple and easy to understand, but it is not completely general. The problem is that an item such as `^{\TeX}` is expanded when the `\futurelet` sees it (before sanitizing). Therefore, its expansion, rather than its name, is written on the file. When the item is enclosed in braces `^{\TeX}`, the `\futurelet` only sees the `{`, so the item is not immediately expanded. After sanitizing, its name is written on the file. However, because of the sanitizing, the name of the item, rather than its

expansion, is also typeset in the document. In the case of index items, the user can write the control sequence twice, once outside the `^` to expand it, and once inside the `^` to write its name on the file. Thus `^{\TeX} [\TeX]{}`.

In general, a way is needed to write the contents of any string, with no expansion, on a file, then expand it. Unfortunately, sanitizing is done by changing catcodes and, once a catcode is assigned to a token, this assignment is permanent [39] and cannot be changed. A solution exists, however, and is developed below.

Verbatim copying with an auxiliary file. Developing the solution is done in three steps. In step 1, a simple macro, `\VwriteC`, is developed that can write strings on a file without expanding control sequences. Its limitations are: (a) The macro sanitizes certain characters so, after writing the string on a file, it (the string) cannot always be expanded; (b) A multi-line string is written on the file as one line.

In step 2, limitation (a) above is removed. Macro `\VwriteD` is a generalization of `\VwriteC`, which does the following: The string is written on the file as before, and is then written on another file which is immediately `\input`. When the string is read back, all tokens get their normal catcodes, and the string can be expanded as usual.

In step 3, macro `\VwriteD` is modified, à la `\elp` below, to scoop up one input line at a time. The result is called `\VwriteE`. This way, a multi-line string is written on a file line by line. The string can also be long, since only one line need be saved at a time.

Note that `\VwriteE` is a generalization of macro `\VwriteD` which, in turn, has been developed from `\VwriteC`. The reader is advised to carefully follow the development of all three macros, however, since each has different features and involves different ideas that can be used for other problems, not just verbatim copying.

Step 1: Verbatim copying with a toks register.

To write a string on a file without expansion, it is placed in a `\toks` register, and then written from the register.

```
\toks0={...string...}
\immediate\write\out{\the\toks0}
```

This works since the control sequence `'\the'` creates a string of tokens, all of catcode 12, except spaces. Fortunately, `'\the'` can be applied to a `\toks` register. Note that this illustrates an advantage of `\toks` registers over macros, since `'\the'` cannot be applied to a macro. Trying to say:

```
\def\aux{...string...}
\immediate\write\out{\aux}
```

would expand `\aux` and all the commands in it. Things such as `\noexpand\aux` or `\string\aux` would simply write the name of the macro, not its contents, on the file.

In practice, the string to be written on a file is the argument of a macro, so the actual code is:

```
\def\aux#1\endP{\toks0=\expandafter{#1}
\immediate\write\out{\the\toks0}
...}
```

This works since `\expandafter` is a one-step expansion. It expands `#1` into individual tokens, but does not expand the tokens further. Our first version is thus:

```
\newtoks\str \newwrite\out
\immediate\openout\out=filename
\def\VwriteC{\begingroup\sanitize \aux}
\def\aux#1\endP{\str=\expandafter{#1}%
\immediate\write\out{\the\str}%
\endgroup}
\def\sanitize{\catcode\ =12
\catcode\%=12 \catcode\#=12
\catcode\{=12 \catcode\}=12 }
```

Following which, the macro can be expanded by, e.g.: `\VwriteC a$x\le0$c C\vrule 1\endP` or `\VwriteC X\hskip10pt\sum\endP`

Step 2: Verbatim copying with an auxiliary file. The string written on a file cannot, in general, be expanded by our macros, since the catcodes of the characters `#{}` (and of the space) were changed. Trying to say, e.g.:

```
\def\aux#1\endP{\str=\expandafter{#1}%
\immediate\write\out{\the\str}%
\endgroup #1}
```

might produce wrong results, or an error message, if the string contains a `#`, an `&`, or any braces.

Macro `\VwriteD` below solves this problem by writing the string on a second file, and reading it back immediately. Upon reading, the string is parsed into tokens that get their normal catcodes. The string can now be expanded. This is a general (albeit slow) solution to the problem. `TEX` is coerced into scanning the same string twice, the first time with special catcodes, and the second time, under normal conditions.

```
\newtoks\str \newwrite\out \newwrite\Tmp
\def\sanitize{\catcode\ =12
\catcode\%=12 \catcode\#=12
\catcode\{=12 \catcode\}=12 }
```

```
\immediate\openout\out=\jobname.aux
\def\VwriteD{\begingroup\sanitize \aux}
\def\aux#1\endP{\str=\expandafter{#1}%
\immediate\write\out{\the\str}%
\immediate\openout\Tmp=\jobname.tmp
\immediate\write\Tmp{\the\str}%
\endgroup \immediate\closeout\Tmp
\input\jobname.tmp }
```

Step 3: Verbatim copying of a multi-line string. We start by developing a macro `\elp` (short for End Line Parameter), whose single parameter is delimited by the end of the line. It is used to pick up an entire input line at a time. We cannot simply write `\def\elp#1^^M{...}` since the `^^M` would terminate the current line and send `TEX` looking for the definition `{...}` on the next line.

So we try to change the catcode of the end-of-line (carriage return) character. Initially we try to change it to 13 (active). The first attempt is `\def\elp#1^^M{\catcode\^^M=13...}`, but this does not work since, when `TEX` finds the catcode change, it has already scanned and determined what the argument is. We have to change the catcode before `\elp` is expanded. The definitions `\catcode\^^M=13\def\elp#1^^M{...}` work, but this means that `\elp` can only be used when the catcode change is in effect (i.e., inside a group).

A similar solution defines `\elp` in `\obeylines` mode [352] (in which `^^M` is active). Thus `{\obeylines\gdef\elp#1^^M{...}...}`. It has the same disadvantage as above.

A better solution is to define `\elp` *without a parameter*, change the catcode inside `\elp` (by means of `\obeylines`), and then expand another macro, `\getpar`, that actually picks up the argument. The result is:

```
\def\elp{\begingroup\obeylines\getpar}
{\obeylines
\gdef\getpar#1
{ '#1' \endgroup}}
```

Macro `\elp` performs the catcode change, and expands `\getpar`. Macro `\getpar` is thus always expanded when `TEX` is in `\obeylines`, but `\getpar` is also defined inside an `\obeylines`. The fact that its definition is on a separate line means that its parameter, `#1`, is delimited by an end-of-line. The `\endgroup` in `\getpar` terminates the effect of the catcode change.

The next step is to realize that the catcode of `^^M` can be changed simply to 12 (other), and there is no need to bother with active characters. Perhaps the best solution is:

```
\def\elp{\begingroup%
\catcode'\^^M=12 \elpAux}
{\catcode'\^^M=12 \gdef\elpAux#1^^M{%
\message{'#1'}\endgroup}}
```

The catcode change is localized by means of `\begingroup` and `\endgroup`. An auxiliary macro, `\elpAux` is used to actually pick up the argument. The macro can be used anywhere.

After this introduction, macro `\VwriteE` is presented. It can read a long, multi-line, argument and write it on a file line by line. The idea is to have macro `\aux` scoop up one line of the source string as its argument, write it on the file, then expand `\VwriteC` recursively until a certain string (`\endP` in our case) is found, that signals the end of the argument.

The main difference between `\VwriteE` and `\VwriteC` is the definition of `\aux`. The version of `\aux` that's expanded by `\VwriteE` below is defined in a group where the catcode of `(return)` is set to 12. It uses the principles of `\elp` to scoop up one line of the argument, write it on the file, and expand `\VwriteE` recursively.

```
\newtoks\str \newwrite\out
\def\sanitize{\catcode'\ =12
\catcode'\%=12 \catcode'\#=12
\catcode'\{=12 \catcode'\}=12 }
\immediate\openout\out=\jobname.aux
```

```
\def\VwriteE{\begingroup\sanitize%
\catcode'\^^M=12 \aux}
```

```
{\catcode'\^^M=12%
\gdef\aux#1^^M{\def\temp{#1}%
\ifx\temp\enP%
\gdef\next{\relax}%
\else \str=\expandafter{#1}%
\immediate\write\out{\the\str}%
\gdef\next{\VwriteE}%
\fi\endgroup\next}}
```

```
\def\enP{\endP}
```

A typical expansion now looks like:

```
Any text... \VwriteE9A{\bf abc} \B
11\halign{#\cr1\cr}\TeX
x$\yy@#%~&_?}\it {\c
\endP
...more text
```

Notes:

1. The `\endP` must be on a line by itself, and must start on column 1. The user may, of course, change from `'\endP'` to any other string.

2. Macro `\aux` is defined when the catcode of `(return)` is 12. Therefore, every line in the definition of `\aux` must be delimited by a `'`. Otherwise the end of line would be typeset as `\char'015` in the current font. (The table on [367] shows that `'015` is the character code of `(return)`.)
3. The temporary macro `\next` is defined by `'\gdef'` instead of by `'\let'`, since it is defined inside `\aux` and `\aux` is defined inside a group.
4. It seems that steps 2 and 3 can be combined. It is suggested that the reader develop a macro `\VwriteF` with the combined features of `\VwriteD` and `\VwriteE`.
5. The three macros above write the value of the toks register `\str` on the file. They therefore cannot use a delayed `\write`, and must use `\immediate\write`. Trying to say `'\write\out{\the\str}'` would delay all the write operations to the OTR, where register `\str` may contain the string from the most recent write, or may even be undefined.

Approach 2. In this approach there are no catcode changes (except that `\obeyspaces` is used locally, during scanning). The string is input and is parsed into tokens in the normal way. This way, our macros can expand it by simply saying `'#1'`. The first version, `\VwriteM`, scans the string of tokens and inserts a `\noexpand` in front of every control sequence token. The second version, `\VwriteN`, does the same scanning, and changes the meaning of every control sequence to `\relax`. The scanning is done with macros `\scan` and `\onestep`, which are based on the last example on [219].

Version 1: verbatim copying with `\noexpand`. Macro `\onestep` receives the next token in the string, checks to see if it is a control sequence (by comparing its catcode to that of `\relax`) and, if it is, inserts a `\noexpand` in front of it. The new string is created, token by token, in the toks register `\str`. The only step that needs detailed explaining is macro `\temp`. It is important to understand why this macro is necessary (why not simply say `'\immediate\write\out{\the\str}'`), and why the use of `\edef`?

Consider the expansion `'\VwriteM{a\TeX}'`. When scanning is complete, the toks register `\str` contains `'a\noexpand \TeX '` (including the spaces). Now `'\immediate\write\out{\the\str}'` would write that string (including the `\noexpand`) on the file, as in approach 1 above. Defining `\temp` by means of `\def` would make `'\the\str'` the replacement text of `\temp`, so the command

'\immediate\write\out{\temp}' expands \temp and would be identical to writing '\the\str'. The \edef, however, creates 'a\noexpand \TeX' as the replacement text of \temp. During the write operation \temp is expanded, which is when the \noexpand does its job and prevents the expansion of \TeX.

```
\newwrite\out \newtoks\str
\immediate\openout\out=filename
```

```
\def\VwriteM#1{'#1'}{\str={}}%
\obeyspaces\scan#1\end
\edef\temp{\the\str}
\immediate\write\out{\temp}}}
```

```
\def\scan#1#2\end{\def\aux{#1}%
\ifx\aux\empty
\else
\def\aux{#2}%
\onestep{#1}%
\ifx\aux\empty
\else
\scan#2\end
\fi\fi}
```

```
\def\onestep#1{\ifcat\relax\noexpand#1%
\str=\expandafter{\the\str\noexpand#1}%
\else\str=\expandafter{\the\str #1}\fi}
```

Note the following:

1. There is a (local) use of \obeyspaces. Without it, spaces are skipped when T_EX determines the arguments of \scan.
2. Because no catcodes are changed, the four characters '#%{}' cannot appear in the argument of \VwriteM. A '#' in the argument will become '##' when the argument is absorbed. A '%' will send T_EX looking for the rest of the argument on the next line. Unbalanced braces will cause an error message when the argument is absorbed. Balanced braces would be absorbed, would be used to nest groups in the argument, and will not appear on the file. For this reason, the use of \VwriteM is limited to cases where these characters do not appear in the strings to be written on file.
3. The \noexpand command adds an extra space. Thus '\VwriteM{\?M}' writes '\? M' on the file. To suppress the space, use \string instead of \noexpand in

```
\str=\expandafter{\the\str\noexpand#1}%
```

This may look better, but may give wrong results in some cases. A typical example is

the expansion '\VwriteM{\bf M}', that would write '\bfM' on the file.

4. Our macros do not attempt to identify active characters. If the string includes any active characters, their expansions would be written on the file. It is, however, relatively easy to test for tokens of catcode 13 and insert a \noexpand in front of them.

Version 2: verbatim copying with \relax.

A different way of avoiding expansion during file output is to temporarily turn an expandable control sequence into a non-expandable one. The simplest way of achieving this is to \let the control sequence be equal to \relax. Thus

```
\def\abc{...}
...
{\let\abc=\relax
\immediate\write\aux{\abc}}
```

will write \abc on the file. Using this method we illustrate a different solution to the same problem. In this version, macro \onestep identifies all tokens in the string that are control sequences, and sets each equal to \relax.

After every control sequence in the string has been changed in this way, the string is written on a file. This version is similar to the previous one, the most important difference being that the final quantity being written on the file is '#1' and not the replacement text of a macro or the contents of a toks register. As a result, any braces in the argument will be written on the file (but see below for a subtle problem with braces).

```
\newwrite\out
\immediate\openout\out=filename
\def\VwriteN#1{{%
\scan #1\end\immediate\write\out{#1}}}
```

```
\def\scan#1#2\end{\def\aux{#1}%
\ifx\aux\empty
\else
\def\aux{#2}%
\onestep{#1}%
\ifx\aux\empty
\else
\scan#2\end
\fi\fi}
```

```
\let\Let=\let
\def\onestep#1{\ifcat\relax\noexpand#1%
\Let#1=\relax\fi}
```

The following points should be mentioned:

1. Macro `\VwriteN` has an extra pair of braces, so everything done in it is local. This way, the setting of control sequences to `\relax` is only temporary.
2. Imagine the string `'abc\let hjk\x'`. The control sequence `\let` is first identified, and is set to `\relax`. Later the control sequence `\x` is identified, but saying `\let\x=\relax` fails because `\let` is now equal to `\relax`. This is why the command `'\let\Let=\let'` has been added. Macro `\onestep` uses `\Let` instead of `\let`. Of course, a string such as `'...\Let...\x'` would cause the same problem, so this method cannot handle such strings.

Exercise: What about the string `'...\Let'`?

Answer: If `\Let` is the last control sequence (or the only one) in a string, our macros can handle it.

3. The above considerations apply to other commands used by `\onestep`, such as `\ifcat` and `\noexpand`. In principle, they should be redefined.
4. The `#` and active characters still cannot appear in the argument to `\VwriteN`, for the same reason as above.
5. Braces in the argument still must be balanced, but will be written on the file as mentioned earlier. There is another, subtle, problem associated with braces. Consider the expansion `'\VwriteN{\bf M}'`. At a certain step during the scanning, the argument of `\onestep` becomes the group `'\bf M'`. The `\noexpand#1` thus becomes `'\noexpand\bf M'` which typesets the `'M'`. The `'\Let#1=\relax'` becomes `'\Let\bf M=\relax'` which lets `\bf` to `'M'` and typesets the `'=`. As a result, this version too, should only be used in limited cases.
6. Macro `\scan` does not use tail recursion because it has to expand either `\onestep` or itself with different parameters. As a result, each recursive expansion of `\scan` saves two `\fi`'s in the parameter stack, whose size is limited. A long argument will thus exceed `TEX`'s capacity. This limitation is removed in the indexing example below.
7. This method works for an `\immediate\write` only. A non-immediate `\write` is executed in the OTR, where the various control sequences are no longer equal to `\relax`. This limitation, too, is removed in the indexing example below.

Indexing example. We again use indexing as an example to illustrate a general solution to the problem of verbatim copying. The macros for indexing discussed below are general and sophisticated

but — in the opinion of the author — still readable. Among other things, they show how to handle general strings, and how to write the page number with the string. The basic task of the macros is to pick up certain items (flagged by a `^`) and write them on the `.idx` file, which is later processed by `MakeIndex` (and, perhaps, other utilities) to create the final index.

The circumflex `^` is defined, as usual, to be the indexing character. It is declared active and is defined to be macro `\Caret`. A valid index item for the macros below must be one of the following:

- `^|abc|` where `abc` may contain any special characters (including unbalanced braces). The string `abc` is typeset verbatim, and also written verbatim on the `.idx` file.
- `^|abc|` where `abc` is as before. This is a 'silent' index item that's only written on the `.idx` file but is not typeset.
- `^|xyz|` where `xyz` may contain special characters (including a `^`) but not a `\` (since it is sanitized during indexing), and not unbalanced braces. The string `xyz` will be typeset and written on the `.idx` file.

It is, however, invalid to say `^\abc` because the `\` is sanitized during indexing (one should say `^[\abc]\abc` instead). Also the argument of a macro cannot have index items, since all tokens in the argument get their catcodes when the argument is absorbed, and those catcodes cannot be changed later.

Here are examples of valid index items (see Refs. 1, 2 for the special meaning of the `!`, the `@` and the parentheses).

```
^[character!special] ^|\pop|
^[verbatim!listing|(| ^[cmsy10]
^| | ^[|!|!as an index] ^[null@<null>]
^[|^M|] ^[|\TeX||)] ^{#%~&}
^{page break} ^{\dvi\ file}
```

Exercise: How can one index a left (or right) brace?

Answer: It is invalid to write `^{{}`. An item of the form `^[]` is fine, but it creates a record of the form `'\indexentry{}-1'` on the `.idx` file, which cannot be properly read later. The item `^| |` is fine but is not silent. A good choice is `^[| |]`. It is silent and it writes `'\indexentry{| |}-1'` on the file. An even better choice is `^[| |@ (left brace)]`. The part on the left of the `@` is the sort key, and the part on the right will be typeset (the print key).

If the `^` is used outside math mode, it becomes macro `\Caret`, which expands `\indexT`, which, in

turn, uses `\futurelet` to peek at the following token. If that token is a `'l'`, macro `\inxC` is expanded. If it is a `'[`, macro `\inxB` is expanded; otherwise, `\inxA` is expanded. Each of these macros, in turn, expands `\finidx` which is responsible for the rest of the job.

Macro `\finidx` uses the primitive `\meaning`, so a short review of `\meaning` is necessary. The control sequence `\meaning [213]` creates a short explanation of its argument (such as `'the letter'`, `'macro'` or `'math shift character'`), followed by the tokens that make up the argument, with catcode 12 attached (except spaces, which get catcode 10). If the argument is a macro (e.g., `'\def\abc#1;{A\B$#1~&\C_}'`), then the commands `'{\tt\meaning\abc}'` result in `'macro:#1;->A\B $#1~&\C _'`. The unnecessary tokens at the beginning can easily be stripped off by using `>` as a delimiter. This is done by macro `'\def\strip#1>{'`.

Macro `\finidx` places the index item in a macro (`\idxitem`) and uses `\meaning` to obtain a string consisting of the individual tokens of the index item, each with catcodes as shown above. This string (together with other things) becomes the replacement text of macro `\INDEX` when `\finidx` says:

```
\edef\INDEX{\write\inx{%
  \string\indexentry%
  {\expandafter\strip\meaning\idxitem}%
  {\noexpand\folio}}}
```

A typical definition of `\INDEX` is thus:

```
\write\inx{\string\indexentry{abc}%
  {\noexpand\folio}}
```

`\INDEX` is then immediately expanded. Its expansion, however, is the `\write` command, which is not immediate, and is therefore saved, as a whatsit, in the MVL, to be executed in the OTR. Note that macro `\INDEX` is no longer needed, and can therefore be redefined when the next index entry is encountered.

Here is the complete set of indexing macros:

```
\def\Caret{\ifmode\def\next{^}%
  \else\let\next=\indexT\fi\next}
\catcode'\^=\active \let^=\Caret
\def\indexT{\begingroup\sanitize%
  \makeother\\\obeyspaces%
  \makebgroup\{\makeegroup\}%
  \futurelet\new\inxcase}
\def\inxcase{\begingroup%
  \ifx\new|\aftergroup\tmpC
  \else\ifx\new[\aftergroup\tmpB
  \else\aftergroup\inxA \fi\fi \endgroup}
```

```
\def\tmpC{\makeother\}\makeother{\\inxC}
% deactivate braces during ^[...] & ^|...|
\def\tmpB{\makeother\}\makeother{\\inxB}
\def\inxA#1{\finidx{#1}#1}
\def\inxB[#1]{\finidx{#1}}
\def\inxC|#1|{\finidx{|#1||#1|}}
\def\strip#1>{}
\long\def\finidx#1{\def\idxitem{#1}%
  \edef\INDEX{\write\inx{\string\indexentry%
    {\expandafter\strip\meaning\idxitem}%
    {\noexpand\folio}}}%
  \INDEX \endgroup}
```

A Warning. If a word is immediately followed by an index item, and the word happens to be the last one on the page, there is a chance that the item would be written on the index file with the number of the next page. The reason for this is that the indexing macros generate a (delayed) write, which becomes a whatsit in the MVL. Such a whatsit is executed in the OTR, when the page is shipped out. If the whatsit follows the last line of text on the page, there is a chance that the page builder would leave the whatsit in the MVL when preparing the current page. In such a case, the whatsit would become the first item on the next page.

A similar thing may happen if an index item immediately precedes the first word of a page. The item may end up being written on the index file with the number of the previous page.

A typical example is `'...^[abc]xyz...'`. If `'xyz'` happens to be the first word on page 2, index item `'abc'` may be written on the file with page number 1. If the document is short, the user may notice such a thing, and correct it by saying `'...\hbox{^[abc]xyz}...'`. This firmly attaches the index item to `'its'` word (but then the word can no longer be hyphenated). If the word `'xyz'` starts a paragraph, the user should change the mode to horizontal by `'...\leavevmode\hbox{^[abc]xyz}...'`

Verbatim listing

We now turn to the other aspect of verbatim namely, verbatim listing. The problem is to typeset verbatim any string of tokens, including spaces, braces, backslashes, or any other special characters. The problem is only important to people who write about \TeX . Most other texts can get away with `'\$'` or `'${$}'` to typeset any occasional special characters.

We start with a short review of interword spaces. A space (between words) is glue whose value is determined by the font designer. It is usually flexible but, in a fixed-space font, it should

be rigid (its value for font `cmmt10`, e.g., is 5.25pt). The size of a space is affected by the space factor, so that spaces following certain punctuation marks get more stretch (and sometimes even greater natural size). Naturally, this discussion applies to any character with catcode 10 (space being the only character assigned this catcode by INITEX [343]).

Consecutive spaces are treated as one space. To defeat this, the `plain` format offers macro `\obeyspaces`. The format starts by defining [351] `\def\space{ }'`. Thus `\space` is a macro whose replacement text is a normal space (affected by the space factor). Next, `\obeyspaces` is defined as a macro that declares the space active `\def\obeyspaces{\catcode'_ =13}'`, and `plain` says (on [352])

```
{\obeyspaces\global\let_=\space}
```

This means that when `\obeyspaces` is in effect (when the space is an active character) the space is defined as `\space`.

To get spaces that are not affected by the space factor, one of the following methods can be used:

- Change the `sf` codes of the punctuation marks to 1000 by means of `\frenchspacing`.
- Use a control space `_`. Control space [290] is a primitive that inserts glue equal to the interword space of the current font, regardless of the space factor. Defining the space as a control space is done by saying `{\obeyspaces\global\let_=_}'`.
- Assign nonzero values to `\spaceskip` and `\xspaceskip`.

Now we are ready for the verbatim macros. Four macros are discussed here, all extensions of macro `\elp` above. The aim is to develop macros that would typeset any given text, verbatim, in font `cmmt10`. The main problem is that the text may include special characters, such as `\` and `#`, so these have to be turned off temporarily. Another problem is that the text has to be picked up line by line, and each line typeset individually. We shouldn't try to absorb the entire text as a macro argument since there may be too much of it. Other problems have to do with blank lines and consecutive spaces.

We start with macro `\sanitize` that's used, as usual, to change the catcodes of certain characters to 12 (other). It is similar to `\dospecials` [344].

```
\def\makeoother#1{\catcode'#1=12\relax}
\def\sanitize{\makeoother%\makeoother\#%
\makeoother\~\makeoother\\ \makeoother\}%
\makeoother\{\makeoother\&\makeoother\$\%
\makeoother\_ \makeoother\^ \makeoother\~M%
\makeoother\ }
```

Now comes the main macro `\ttverbatim`. We tentatively start with the simple definition

```
\def\ttverbatim{\begingroup
\sanitize\tt\gobble}
```

but the final definition below also contains

```
'\def\par{\leavevmode\endgraf}'
```

(in addition to a few other things). This is necessary because of blank lines. A blank line becomes a `\par` in the mouth, and `\par` has no effect in vertical mode. We thus have to switch to horizontal mode and do an `\endgraf`, which is the same as `\par`.

Macro `\gobble` gobbles up the end-of-line following the `\ttverbatim`, and expands `\getline` to get the first line of verbatim text. Without gobbling, `\getline` would read the end-of-line and translate it into an empty line in the verbatim listing.

Macro `\getline` gets one line of text (à la `\elp`), typesets it, executes a `\par`, and expands itself recursively. When it senses the end of the verbatim text, it should simply say `\endgroup` to revert to the original catcodes. The end of the text is a line containing just `\endverbatim` (without any preceding blanks), and the main problem facing `\getline` is to identify this line. The identification is done by means of `\ifx`, which compares two strings, stored in macros, character by character. The point is that an `\ifx` comparison is done by character code *and* category code. When the `\endverbatim` is read, sanitizing is in force, and the `\` has catcode 12 (the eleven letters have their normal catcode, 11).

We thus cannot simply define a macro `\def\endverb{\endverbatim}'` and then compare `\ifx\endverb\aux`, because the string in macro `\endverb` starts with `_0` instead of `_12`. The solution is to define `\endverb` in a group where the `\` has catcode 12. Thus

```
{\catcode'\_*=0 \makeoother\%
*gdef*endverb{\endverbatim}}
```

Now `\getline` can say `\ifx\endverb\aux`.

One of the verbatim methods below uses the vertical bar `|` to delimit small amounts of verbatim text. This is done by declaring the `|` active. Since we want to be able to include the `|` in verbatim listings, we sanitize it in `\ttverbatim` by saying `\makeoother|`.

After using the macros for several years, I was surprised one day to see a `?` listed as `¿`. A closer look revealed that it was the pair `?'` that was listed as `¿`. It took a while to figure out that, in the `cmmt` fonts, the combinations `?'` and `!'` are considered

ligatures and are replaced by \grave{u} and \grave{i} , respectively (Ref. 3, p. 36).

The solution is to declare the left quote active and to define it as macro `\lq`. This is why `\ttverbatim` and its relatives include the command `\catcode'\='=13`, and why the code `{\catcode'\='=13\relax\gdef{\relax\lq}}` is also necessary (see also [381]).

The definitions of the two macros should now be easy to understand.

```
\def\ttverbatim{\medskip\begingroup\tt%
\def\par{\leavevmode\endgraf}%
\catcode'\='=13\makeother\| \sanitizelgobble}
{\makeother\~\M\gdef\gobble\~\M{\getline}%
\gdef\getline#1\~\M{\def\aux{#1}%
\ifx\endverb\aux\let\next=\endgroup\medskip%
\else#1\par\let\next=\getline\fi\next}%
\obeyspaces\global\let\lq=\_ \catcode'\='=13%
\relax\gdef{\relax\lq}}
{\catcode'\* =0 \makeother\| %
*gdef*endverb{\endverbatim}}%
```

Exercise: The verbatim text above contains `\endverbatim`, but this string terminates verbatim listings. How was the text produced?

Answer: The string `\endverbatim` is only assigned its special meaning when it appears on a line by itself, with no preceding spaces, so in our case there was no problem. It is, however, possible to list an `\endverbatim` anywhere using the `|` (see below).

Readers trying these macros will very quickly discover that they typeset `'_'` instead of spaces. This is because the space (whose character code is '40) has been sanitized (it is now a regular character, of catcode 12) and font `cmntt10` has `'_'` in position '40. This feature is sometimes desirable, but it is easy to modify `\ttverbatim` to get blank spaces in the verbatim listing.

The new macro is called `\verbatim`, and the main change is to say `\obeyspaces` instead of sanitizing the space. In verbatim listings, of course, we don't want the space to be affected by the space factor, so `{\obeyspaces\global\let\lq=_}`.

```
\def\makeother#1{\catcode'#1=12\relax}
\def\sanitizelgobble{\makeother\% \makeother\#%
\makeother\~ \makeother\| \makeother\}%
\makeother\{ \makeother\& \makeother\${%
\makeother\_ \makeother\~ \makeother\~\M}%

```

Macros `\verbatim` and `\getline` are defined by:

```
\def\verbatim{\medskip\begingroup\tt%
\def\par{\leavevmode\endgraf}%
\catcode'\='=13\sanitizelgobble\~\M%
\makeother\| \obeyspaces\gobble}
```

```
{\makeother\~\M\gdef\gobble\~\M{\getline}%
\gdef\getline#1\~\M{\def\aux{#1}%
\ifx\endverb\aux\let\next=\endgroup\medskip%
\else#1\par\let\next=\getline\fi\next}%
\obeyspaces\global\let\lq=\_ \catcode'\='=13%
\catcode'\='=13\relax%
\gdef{\relax\lq}}
{\catcode'\* =0 \makeother\| %
*gdef*endverb{\endverbatim}}%
```

Exercise: why do we have to place `{\obeyspaces\global\let\lq=_}` outside the macros? It seems more elegant to have it included in the definition of `\verbatim`.

Answer: If we place it inside a macro, then the space following `\let` would get catcode 10 when the macro is defined. When the macro is expanded later, the `\let` command would fail, because it is followed by a catcode 10 token instead of by an active character.

Note the two `\medskip` commands. They create vertical spacing around the entire listing, and the first one also makes sure that the listing is done in vertical mode. They can be replaced, of course, by any vertical skip (flexible or rigid), depending on specific needs and personal taste.

Preventing line breaks. Each line of a verbatim listing is typeset by saying (in `\getline`) `'#1\par'`. The line becomes a paragraph and, if it is too wide, it may be broken. If this is not desirable, then the code above may be changed to `\hbox{#1}`. Macro `\verbatim` changes the mode to vertical, which means: (1) the boxes will be stacked vertically; (2) a wide box will not cause an 'overfull box' error.

Line numbers. The definition of `\verbatim` is now generalized to also typeset line numbers with the verbatim text. Macro `\numverbatim` below uses the same `\sanitizelgobble` as `\verbatim`, and a new count register is declared, to hold the current line number. The line numbers are typeset on the left margin, by means of an `\llap`, but this is easy to modify.

```
\newcount\verblines
\def\numverbatim{\medskip\begingroup\tt%
\def\par{\leavevmode\endgraf}%
\catcode'\='=13\sanitizelgobble\~\M%
\obeyspaces\verblines=0
\everypar{\advance\verblines1
\llap{\sevenrm\the\verblines\_ \_}}\gobble}
{\makeother\~\M\gdef\gobble\~\M{\getline}%
\gdef\getline#1\~\M{\def\aux{#1}%
\ifx\endverb\aux\let\next=\endgroup\medskip%
```

```

\else#1\par\let\next=\getline\fi\next}%
\obeyspaces%
\global\let\lq=\lq\catcode'\|=13\relax%
\gdef{\relax\lq}
{\catcode'\|=0 \makeother\|}%
*gdef*endverb{\endverbatim}}%

```

Verbatim in horizontal mode. The macros developed above are suitable for ‘large’ verbatim listings, involving several, or even many, lines of text. Such listings are normally done in vertical mode, between paragraphs. The next approach declares the vertical bar ‘|’ active, and uses it to delimit small amounts of text (normally up to a line) that should be listed verbatim within a paragraph. This is convenient notation, commonly used, whose only disadvantage is that the ‘|’ itself cannot appear in the text to be listed.

The first step is not to sanitize the space and the end-of-line:

```

\def\makeother#1{\catcode'#1=12\relax}
\def\sanitize{\makeother%\makeother\#%
\makeother~\makeother\\makeother\}%
\makeother\{\makeother\&\makeother\$\%
\makeother\_makeother\^}%

```

Next, the ‘|’ is declared active, and is defined similar to `\verbatim` above. The main differences are:

- Macro `\moreverb` can pick up the entire text as its argument, since there is not much text.
- Instead of defining the space as a control space, we preempt the space by assigning non-zero values to `\spaceskip` and `\xspaceskip`.

```

\catcode'\|=13
\def|{\begingroup\obeyspaces%
\catcode'\|=13\sanitize\moreverb}
\def\moreverb#1{\tt\spaceskip=5.25pt%
\xspaceskip=5.25pt\relax#1\endgroup}

```

(The value 5.25pt is the interword space of font `cmmt10`. If a different font is used, this value should be replaced by its interword space. An alternative is to use `.51em`, which gives good results in most sizes of `cmmt`.)

Exercise: Why is the `\relax` necessary after the 5.25pt?

Answer: To terminate the glue specification. Without the `\relax`, if `#1` happens to be one of the words `plus` or `minus`, `TeX` would consider it to be part of the glue assigned to `\xspaceskip`, and would expect it to be followed by a number.

The reader should note that ‘|’ cannot be used in the argument of a macro. If `\abc` is a macro,

we cannot say, e.g., `\abc{...|xyz|...}`. The reason is that all tokens in the argument get their catcodes assigned when the argument is absorbed, so ‘|’ cannot change them later. Using ‘|’ in a box, however, is okay.

Exercise: Change the definition of ‘|’ to typeset ‘|’ instead of blank spaces.

Answer: Instead of using `\obeyspaces`, just sanitize the space (also the settings of `\spaceskip` and `\xspaceskip` are no longer necessary).

A different approach. Incidentally, there is a completely different approach to the problem of verbatim listing, using the primitive `\meaning`. The way this control sequence works has been reviewed earlier. To use it for verbatim listing, we simply say (compare with [382]):

```

\long\def\verbatim#1{#{%
\tt\expandafter\strip\meaning#1}}
\def\strip#1>{}

```

Since a macro is a token list, we can get verbatim listing of tokens this way, but with the following limitations: (1) extra spaces are automatically inserted by `\meaning` at certain points; (2) end of lines become spaces in the verbatim listing; (3) a single ‘#’ cannot be included in the verbatim text (unless it is sanitized).

Fancy verbatim

Sometimes it is necessary to typeset parts of a verbatim listing in a different font, or to mix verbatim and non-verbatim text. Following are extensions of the verbatim macros, that can read and execute commands before starting on their main job. The commands are typically catcode changes but, in principle, can be anything. The commands are specified in two ways. Commands that should apply to all verbatim listings of a document are placed in the `\toks` register `\everyverbatim`. Commands that should apply to just certain listings are placed between square brackets right following `\verbatim`, thus `\verbatim[...]`.

Macro `\verbatim` uses `\futurelet` to sneak a look at the token following the ‘m’. If this is a left bracket, the commands up to the right bracket are executed. Sanitization is done before the commands are executed, so the user can further modify the catcodes of sanitized characters. However, since the commands start with a ‘\’, sanitization of this token should be deferred. The code below shows how `\verbatim` places the next token into `\nextc`, how `\options` expands `\readoptions` if this token is a ‘[’, and how `\readoptions` scoops

up all the commands and executes them. Macro `\preverbatim` sanitizes the `\`, and performs the other last minute tasks, before expanding `\gobble`.

```
\def\verbatim{\medskip\beginGroup\sanitize%
\the\everyverbatim%
\makeother\~M\futurelet\nextc\options}
\def\options{%
\ifx[\nextc\let\next=\readoptions%
\else\let\next=\preverbatim\fi\next}
\def\readoptions[#1]{#1 \preverbatim}
\def\preverbatim{\def\par{%
\leavevmode\endgraf}\makeother\%
\makeother\~M\tt\obeyspaces\gobble}
```

Note that the left quote is made active very late (together with the sanitization of the `\`). This means that the optional commands can use it in its original meaning, but they cannot change its catcode. It is possible to say, e.g., `\verbatim[\catcode'*=11]`, but something like `\verbatim[\makebgroup\']` won't work because the left quote will be made active at a later point.

Advanced readers may easily change the macros such that the left quote would be made active early (perhaps by `\sanitize`). In such a case, the effect of

```
\verbatim[\catcode'\*=11] can be achieved by
defining
\def\makeletter#1{\catcode'#1=11 }, then say-
ing
\verbatim[\makeletter*]
```

Similar remarks apply to the curly braces. Saying `\verbatim[\everypar={...}]` is wrong because the braces are sanitized early. The solution is to define `\def\temp{\everypar={...}}`, then say `\verbatim[\temp]`.

The simplest example is `\verbatim[\parindent=0pt]` which prevents indentation in a specific listing. A more sophisticated example introduces the concept of meta code. The idea is that certain pieces of text in a verbatim listing may have to be typeset in a different font (we use `cmr10`). Such text is identified by enclosing it, e.g., in a pair of angle brackets `<>`. The following simple code implements this idea:

```
\def\enablemetacode{\makeactive\<}
\enablemetacode \gdef<#1>{{\tenrm#1}}
```

And the test:

```
\verbatim[\enablemetacode]
\halign{<...preamble...>\cr \beginCont
<...1st line...>\cr
<...>
```

```
<...last line...>\endCont\cr}
\endverbatim
produces
```

```
\halign{<...preamble...>\cr \beginCont
...1st line...>\cr
...
...last line...>\endCont\cr}
```

(It's easy to modify `\enablemetacode` to also typeset the brackets.) Next we introduce fancy comments. Suppose we want to typeset comments in a verbatim listing in italics. A comment is anything between a `%` and the end of line. Again, the following simple code is all that's needed to achieve this. It declares the `%` active (preempting the action of `\sanitize`), and defines it as a macro that typesets its argument in `\it` (the `%` itself can also be in `\it`, if desired).

```
\def\itcomments{\makeactive\%}
{\itcomments%
\gdef%#1\par{\char"25{\it#1}\par}}
```

The test

```
\verbatim[\itcomments]
line 1 %comment 1
line 2 % Comment #2
line 3 % Note \this
\endverbatim
```

results in

```
line 1 %comment 1
line 2 % Comment #2
line 3 % Note this
```

The next example typesets selected parts of a verbatim listing in `\bf`. The `!` is declared a temporary escape character, and the two parentheses, as temporary braces. The result of:

```
\verbatim[\makeescape\!
\makebgroup\(\makeegroup\))
(!bf while) \lineno>\totalines
\lineshiped:=\totalines
(!bf extract) \temp (!bf from) \Bsav
\totalines:=\totalines+\temp
(!bf end while);
\endverbatim
```

is

```
while \lineno>\totalines
\lineshiped:=\totalines
extract \temp from \Bsav
\totalines:=\totalines+\temp
end while;
```

Note that the fancy commands between the square brackets should all fit on one line. They were broken

over two lines in the example above because of the narrow margin of *TUGboat*.

Sometimes, underlining is called for, to indicate keywords in a computer program. This can be achieved with:

```
\def\@#1@{\underbar{#1}}
\verbatim[makeescape\!\catcode'\$=3]
!@var@ x, y, x1, x2: real;
x:=x1;
!@repeat@
y:=a*x+b;
point(round(x),round(y));
x:=x+0.01;
!@until@ x>x2;
\endverbatim
```

resulting in

```
var x, y, x1, x2: real;
x:=x1;
repeat
y:=a*x+b;
point(round(x),round(y));
x:=x+0.01;
until x>x2;
```

Sometimes a mixture of visible and blank spaces is required in the same verbatim listing. Here are two simple ways of doing this. The first one is:

```
{\tt\makeactive\!\gdef!{\char32 }}
\verbatim[makeactive\!]
a 1!!2 3
x!4 5!!!6
\endverbatim
```

resulting in

```
a 1  2 3
x  4 5  6
```

and the second one is:

```
{\makeactive\!\gdef!{\ }}
\verbatim[\vispacetrue\makeactive\!]
a 1!!2 3
x!4 5!!!6
\endverbatim
```

resulting in

```
a  1 2  3
x 4  5 6
```

One more example, to convince the skeptics, that shows how math expressions can be placed inside a verbatim listing. We simply say:

```
\verbatim%
[makeescape\!\catcode'\$=3 \catcode'\^=7]
prolog $\!sum x^2$ epilog
```

\endverbatim

And the result is:

```
prolog  $\sum x^2$  epilog
```

The concept of optional commands is powerful and can be extended to create verbatim listings that are numbered or that show visible spaces. This way, macros `\ttverbatim` and `\numverbatim` are no longer necessary and are replaced by `\verbatim[\vispacetrue]` and `\verbatim[\numbered]`, respectively.

The difference between macros `\verbatim` and `\ttverbatim` is that the former says `\obeyspaces`, whereas the latter says `\makeother\` . We add a boolean variable `\ifvispace` that selects one of the choices above.

Macro `\numverbatim` says

```
\everypar{\advance\verblineli
\llap{\sevenrm\the\verblineli \ }}
\endeverypar
```

We therefore define the two macros:

```
\def\numbered{\everypar{\advance\verblineli
\llap{\sevenrm\the\verblineli \ }}
\def\notnumbered{\everypar{}}
```

that can turn the numbering on and off. The final version of `\verbatim` is shown below.

```
\def\verbatim{%
\medskip\begingroup\tt\sanitize%
\makeother\^M\makeother\|%
\futurelet\nextc\options}
\def\options{%
\ifx[\nextc\let\next=\readoptions
\else\let\next=\preverbatim\fi\next}
\def\readoptions[#1]{#1 \preverbatim}
\def\preverbatim{\def\par{%
\leavevmode\endgraf}%
\lasttasks\ifvispace\makeother\ \else%
\obeyspaces\fi\gobble}
```

The following tests are especially interesting:

```
\everyverbatim{\numbered\vispacetrue}
\verbatim
abc 123 \x %^?'!
@#$$ ^& *( )_
\endverbatim
```

```
\verbatim[\vispacefalse]
abc 123 \x %^?'!
@#$$ ^& *( )_
\endverbatim
```

```
\verbatim[\notnumbered]
abc 123 \x %^?'!
@#$$ ^& *( )_
\endverbatim
```

`\endverbatim`

They result in:

```
1 abc_123_x_%^?‘!‘
2 @$%^&_*( )_
```

```
1 abc 123 \x %^?‘!‘
2 @$%^& * ( ) _
```

```
abc_123_x_%^?‘!‘
@$%^&_*( )_
```

The vertical bar can also take optional arguments. Below we show how to generalize the definition of ‘|’, so things like

```
|[\enablemetacode]...<text>..|
|[\vispacetrue]..A B..|
```

will work.

The method is similar to the one used with `\verbatim`, with one difference: the backslash must be sanitized before `\futurelet` peeks at the next token. Consider the simple example ‘|`\abc`|’. The `\futurelet` will scoop up `\abc` as one (control sequence) token. Later, when `\moreverb` typesets its argument (when it says ‘#1’), there will be an error, since `\abc` is undefined.

If the `\futurelet` reads a ‘|’, the backslash has to be restored (by `\makeescape\|`), so that macro `\readOptions` can read and execute the optional commands. Following that, macro `\preVerb` expands `\lasttasks` to resanitize the backslash before the rest of the verbatim argument is read.

```
\makeactive\|
\def|{\begingroup\tt\obeyspaces%
\sanitize\makeother\|}%
\futurelet\nextc\Voptions}
\def\Voptions{\ifx[\nextc\makeescape\|}%
\let\next=\readOptions
\else\let\next=\preVerb\fi\next}
\def\readOptions[#1]{#1\relax\preVerb}
\def\preVerb{\lasttasks\ifvispace%
\makeother\| \else\obeyspaces\fi%
\moreverb}
\def\moreverb#1|{\spaceskip=5.25pt%
\xspaceskip=5.25pt\relax#1\endgroup}
```

Exercise: Why is the `\relax` necessary in macro `\readOptions` and why isn’t it necessary in the similar macro `\readoptions` (expanded by `\verbatim`)?

Answer: `\readOptions` is expanded in horizontal mode, where spaces are sometimes significant, and `\readoptions`, in vertical mode, where spaces are ignored. The rule is that a space that’s necessary as a separator is not typeset (does not become

spurious). There is no strict need for a space after the ‘#1’ since the # can be followed by one digit only (there can be at most nine parameters).

Exercise: Now that the ‘|’ is special, how can we typeset it verbatim?

Answer: Easy, just turn it temporarily into a letter (catcode 11). The following `{\makeletter\|[\dots]}` works nicely because the `\ifx[\nextc]` compares `\nextc` to a left bracket with catcode 12.

Exercise: What is the effect, if any, of `|[\numbered]...|`?

Answer: No effect, since `\numbered` only changes `\everypar`, which is not used during vertical bar verbatim listing anyway. Since the change is done in a group, it is local.

Complete verbatim macros

Following is the complete code for all the verbatim macros necessary to implement the concepts discussed here. Note the new macro `\verbfile`. It can be used to list the contents of a given file verbatim. The argument ‘#1’ is the name of the file. This is a simple modification of `\verbatim`, without optional commands (but see exercise below).

```
\newtoks\everyverbatim
\newcount\verblines
\newif\ifvispace \vispacefalse
\def\makeescape#1{\catcode#1=0 }
\def\makebgroup#1{\catcode#1=1 }
\def\makeegroup#1{\catcode#1=2 }
% can have similar \make.. macros
% for catcodes 3--10
\def\makeletter#1{\catcode#1=11 }
\def\makeother#1{\catcode#1=12 }
\def\makeactive#1{\catcode#1=13 }
\def\makecomment#1{\catcode#1=14 }

\def\sanitize{\makeother\#\makeother\&%
\makeother\% \makeother\~%
\makeother\} \makeother\{ \makeother\$%
\makeother\_ \makeother\^%
\the\everyverbatim}%
\def\lasttasks{\makeactive\| \makeother\|}

\def\verbatim{\medskip\begingroup\tt%
\sanitize\makeother\~M\makeother\|}%
\futurelet\nextc\options}
\def\options{%
\ifx[\nextc\let\next=\readoptions
\else\let\next=\preverbatim\fi\next}
\def\readoptions[#1]{#1 \preverbatim}
\def\preverbatim{%
\def\par{\leavevmode\endgraf}%
```

```

\lasttasks\ifvispace\makeother\ \else%
\obeyspaces\fi\gobble}

{\makeother\^^M%
\gdef\gobble^^M{\getline}%
\gdef\getline#1^^M{\def\aux{#1}%
\ifx\endverb\aux
\let\next=\endgroup\medskip%
\else#1\par\let\next=\getline\fi\next}%
\obeyspaces\global\let =\ \makeactive\`%
\relax\gdef{\relax\lq}}
{\makeescape\*\makeother\}%
*gdef*endverb{\endverbatim}}%

\makeactive\|
\def|{\begingroup\tt\obeyspaces\sanitize%
\makeother\\\futurelet\nextc\oOptions}
\def\oOptions{\ifx[\nextc\makeescape\}%
\let\next=\readOptions
\else\let\next=\preVerb\fi\next}
\def\readOptions[#1]{#1\relax\preVerb}
\def\preVerb{\lasttasks\ifvispace%
\makeother\ \else\obeyspaces\fi%
\moreverb}
\def\moreverb#1|{\spaceskip=5.25pt
\xspaceskip=5.25pt\relax#1\endgroup}

\def\verbfile#1 {\medskip\begingroup%
\tt\def\par{\leavevmode\endgraf}%
\sanitize\lasttasks\makeother\|%
\obeylines\obeyspaces\input#1\endgroup%
\medskip}

\def\enablemetacode{\makeactive\<}
{\enablemetacode \gdef<#1>{\tenrm#1}}
\def\itcomments{\makeactive\%}
{\itcomments%
\gdef%#1\par{\char"25{\it#1}\par}}
\def\numbered{\everypar{\advance\verblineli
\llap{\sevenrm\the\verblineli \}}}
\def\notnumbered{\everypar{}}

```

Exercise: Why the `\relax` in `\makeactive\` \relax\gdef...` (normally a space is enough to terminate a number)?

Answer: Normally, a space following a number is considered a terminator, and is not printed. However, at this point, because of the `\obeyspaces`, the space is active (has catcode 13 instead of the normal 10), and is defined as a control space. It would therefore be typeset as a spurious space. This is especially annoying if the verbatim macros are part of a format file that is eventually dumped. We don't want such a file to create any typeset material.

Exercise: Extend the definition of `\verbfile#1 {...}` to detect and execute optional commands.

Answer:

```

\def\verbfile{%
\medskip\begingroup\tt\sanitize%
\makeother\| \futurelet\nextc\oOptions}
\def\oOptions{%
\ifx[\nextc\let\next=\getoptions
\else\let\next=\preverbfile\fi\next}
\def\getoptions[#1]{#1 \preverbfile}
\def\preverbfile#1 {%
\def\par{\leavevmode\endgraf}%
\lasttasks\obeylines\ifvispace%
\makeother\ \else\obeyspaces\fi%
\input#1\endgroup\medskip}

```

A typical expansion is `\verbfile[\numbered\vispacetrue]test` where `test` is the name of the file (no space between the `]` and the file name).

References

1. Lamport, L., *MakeIndex: An Index Processor for L^AT_EX*, (available from archives carrying L^AT_EX stuff).
2. Chen, P., et al, *Index Preparation and Processing*, Software—Practice & Experience **18**(#9), 1988, pp. 897–915.
3. Knuth, D. E., *Computers and Typesetting*, Vol E, *Computer Modern Typefaces*, Addison-Wesley, 1986.

◇ David Salomon
California State University,
Northridge
Computer Science Department
Northridge, CA 91330-8281
dxs@secs.csun.edu

Macros

The bag of tricks

Victor Eijkhout

Hello all. The other day I was asked to assist in solving the following problem: format a file of address labels that is given as plain text, like

```
My Name
123 My Street
My Town
```

with the items of a label on separate lines and the labels separated by empty lines. Without inserting any `TeX` commands, of course.

This problem turned out to be a tricky one, and I'll make this into a sort of a tutorial on end-of-line handling in `TeX`¹.

Normally, `TeX` converts the end of the line into a space, but here we want to keep the lines the way they are. For this, we start mucking about with the 'secret character' that `TeX` puts at the end of each line. Every time `TeX` reads a line from the input file, it appends the character with number `\endlinechar`². Usually, this is character 13, and you can write `'\^^M` if you don't want to remember that number³.

The crucial point is that this character has category code 5, for end-of-line. `TeX` converts such characters into spaces, or into `\par` if it finds them on an otherwise empty line⁴.

And now we are in trouble: on the one hand you would want to write a macro along the lines of `\def\OneLabel#1\par{...}`

but that `\par` token needs an `\endlinechar` of category code 5, and that would obliterate the line ends, which we wanted to keep.

Looking at it from the other side, if we change the category code of the line end to anything but 5

¹ And I'll delegate all the 'unlesses' to the footnotes. If you want to read about this topic in more detail, read chapter 2 of my book 'TeX by Topic'.

² Unless this number falls outside the range of 0-255.

³ So why is `'\^^M` easier to remember? Well, in Ascii, `<Control>-M` is the Carriage Return. Does that help?

⁴ It doesn't hurt if your input file has some spaces on the 'empty' line, because spaces at the end of a line are discarded. There is also something about spaces at the beginning of a line, but that's a different story.

in order to keep it recognizable, we don't get a `\par` token after the label text anymore.

There doesn't seem to be another possibility than changing the category code of the line end, and processing the labels line by line.

Let us start programming bottom-up with some preliminaries: we will need macros to process the labels. I assume that you define macros `\AddToLabel` and `\LabelFinished`, for instance like this:

```
\newbox\LabelBox
\def\LabelFinished
  {\box\LabelBox}
\def\AddToLabel#1%
  {\setbox\LabelBox
   \vbox{\unvbox\LabelBox
         \hbox to 5cm{#1\hfil\strut}}}
}
```

Now we continue top-down by specifying how the formatting is going to look to the user. Here is how it could be done in plain `TeX`.

```
\def\endplainlabels{\Bye}
\plainlabels{Here come the labels:\par}
```

```
My Name
My Street 1
My Town
```

```
Your Name
Your Street 2
Your Town
```

```
\Bye
```

The command `\endplainlabels` specifies how `TeX` recognises that the labels are finished. Whatever is on that line is also executed. The `\plainlabels` command⁵ starts the formatting of the labels, and its argument (which can be empty) specifies what ever should be done prior to typesetting the labels. The blank lines before the first and after the last label are optional.

One remark: the `\bye` and `\end` macros are outer macros, so you cannot write

```
\def\endplainlabels{\bye}
```

Instead you have to resort to the following trick:

```
\edef\endplainlabels{\noexpand\bye}
```

Here is the full implementation of the line processing macros. I am assuming that you will put them into a separate file

⁵ I've tried to make this into a `LATEX` environment, but ran into all sorts of problems. Sorry. Simply use the same syntax in `LATEX` as in plain `TeX`.

```

\endlinechar=-1
\def\empty{}
\newif\iflabelpending
\catcode'\^^M\active
\long\def^^M#1^^M{\def\test{#1}
  \ifx\test\endplainlabels
    \catcode'\^^M=5\relax
    \iflabelpending\LabelFinished\fi
    \expandafter\endplainlabels
  \else \ifx\test\empty \LabelFinished
    \labelpendingfalse
    \else \AddToLabel{#1}
    \labelpendingtrue
    \fi
    \expandafter ^^M
  \fi
}
\long\def\plainlabels#1
{\toks0{#1}\labelpendingfalse
\edef\next{\everypar
{\the\everypar
\everypar{\the\everypar}
\the\toks0\relax
\catcode'\noexpand\^^M\active
\noexpand^^M}}
\next\par}
\endlinechar='\^^M \catcode'\^^M=5 \relax
\endinput

```

There are lots of tricky points to these macros. Here are a few

- All that redefining of `\everypar` is for the benefit of packages such as \LaTeX which themselves redefine `\everypar`. The macros given here make sure that the custom `\everypar` first executes whatever was in the old one, and after executing its own commands, restores the old value.
- The conditional `\iflabelpending` handles the case where there is no blank line after the last label. Without it, that label would not be printed.
- The `\par` at the end of `\plainlabels` puts \TeX into vertical mode, so that the first label will trigger `\everypar`.
- The `\toks0` register makes sure that the initial commands get executed after \TeX has come out of vertical mode: this is mostly for the case of \LaTeX lists; see the examples below. They do not like it if an item occurs in vertical mode⁶.

For a slightly more complicated example, let us turn the labels into items in a \LaTeX list. Define

⁶ This is also the reason that I could not use grouping: \LaTeX 's tests have to be set globally.

```

\def\LabelFinished
  {\item[]\box\LabelBox}
and process the labels with
\def\endplainlabels{\end{trivlist}}
\plainlabels{\begin{trivlist}}
...
\end{trivlist}

```

Would you like to have several labels on one line? Use the following macros:

```

\newbox\AllLabels
\def\LabelFinished
  {\setbox\AllLabels\hbox
    {\unhbox\AllLabels\hfil
    \box\LabelBox}}
\def\ejectlabels{\linepenalty100
  \noindent
  \unhbox\AllLabels}

```

This appends all labels to a long `\hbox`, which you'll have to eject at the end — and it is then treated as a paragraph — with

```

\def\endplainlabels
  {\ejectlabels\end{something}}
\plainlabels{Something something}
\ejectlabels\end{something}

```

If you catch my drift. And you may want to make sure that all labels have the same width:

```

\def\AddToLabel#1%
  {\setbox\LabelBox
    \vbox{\unvbox\LabelBox
    \hbox to 3cm{#1\hfil\strut}}
}

```

Finally, a comment for the true hackers among you: suppose you don't want to assume that the `\endlinechar` is 13. First of all you'll have to add a few lines:

```

\count0=\endlinechar \endlinechar=-1
...
\endlinechar=\count0

```

to save and restore the proper `\endlinechar` while you define the macros. More importantly, you don't know what active character to define for processing the lines! Here's a way out:

```

\begingroup
\uccode'\^^N=\count0 \catcode'\^^N\active
\uppercase{\gdef^^N#1^^N}{...}
\endgroup

```

This uppercases an active character 14, and you've set it up so that this is the (saved) end-of-line character.

Happy hacking to you hackers, and to the rest, don't be afraid to ask, we hackers are only too happy

to show off. And reader contributions for this column are still welcome!

◊ Victor Eijkhout
 Department of Computer Science
 University of Tennessee at
 Knoxville
 Knoxville TN 37996-1301
 Internet: eijkhout@cs.utk.edu

Random Bit Generator in T_EX

Hans van der Meer

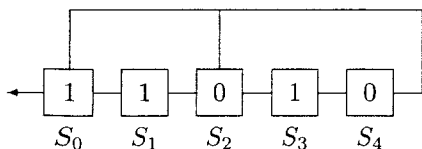
1 Introduction

When I started using T_EX for my collection of exam questions, the need of a random bit generator arose. With such a generator it is easy to randomly permute items of multiple choice questions, choose between different variants, etc.

Since part of my interests are in the field of cryptography it was most natural to look for a convenient source of a random bitstream in that field. Such a source is provided by shiftregisters, the simplest form of which is the linear variety. Although not strong enough for direct use in cryptographic applications, their random properties are nevertheless excellent. Furthermore they are easily implemented, a real asset because of T_EX's limited abilities in arithmetic. The prime reference for shiftregisters is the famous book by Golomb[1].

2 Linear Shiftregisters

Before describing how such a shiftregister can be implemented in T_EX, it is necessary to have a modest look at their construction. The figure shows a small linear shiftregister. It consists of five so-called *stages* $S_0 \dots S_4$ and is therefore called a five-stage register. Each stage is a memory unit capable of holding one bit. The values of all the stages together make up the *state* of the register; in the figure the current state $S = (11010)$.



The register is operated in the following way. At each step the bits in the stages are shifted to the

stage at their left. The bit in stage S_0 is thereby produced as the output bit. Of course the vacancy left in the rightmost stage must be filled up. Therefore all stages which in the figure have an exit at the top of the stage box, also spawn their bit through this exit just before the bit migrates to the left. These exits are called *taps*. The bits spawned are combined by the exclusive-or operator and the resultant bit fills the rightmost stage. E.g., with taps at S_i, S_j, S_k, \dots the mod 2 sum $S_i \oplus S_j \oplus S_k \oplus \dots$ is formed. Thus the register produces an output bit and a new state at each operation step. In the example the output bit will be a 1 and the next state $S = (10101)$.

It is easily understood that eventually the bitstream must repeat itself. Because an n -stage register holds an n -bit quantity it can exist in 2^n different states only. Since new states are produced by a strictly deterministic process, a periodic pattern of successive states must result. Thus the output stream will be periodic. Of course it is desirable that the length of the cycle be as long as possible.

These registers can also be described with a polynomial in a bit variable $x \in \{0, 1\}$, called the *characteristic polynomial*. The example register has characteristic polynomial

$$f(x) = 1 + x^2 + x^5$$

It turns out that the length of the cycle produced by a register characterized by a given polynomial is connected to certain properties of this polynomial. Particularly useful are the so-called primitive polynomials.¹ One is able to show that primitive polynomials lead to the longest possible period for a linear shiftregister of a given size. In fact two cycles are produced: (1) a cycle of period 1 consisting of a stream of zeroes, (2) a fine random stream of zeroes and ones of length $2^n - 1$. The first cycle, the zero cycle, is not entirely useless as it offers a natural way for shutting off the random stream.²

After having explained how a shiftregister works, it is easy to see why I chose the register based on

$$f(x) = 1 + x^{21} + x^{22}$$

for the implementation of a random bit generator in T_EX. It is a primitive polynomial and therefore has a longest period of 4,194,303 bits—more than enough for all but the most exotic applications. And another important fact is that it has only two taps,

¹ Roughly the equivalent of a prime number among polynomials plus an additional condition.

² I am using this stream when typesetting the full collection of exam questions. The absence of random shuffling makes it easier to connect the printed output with the T_EX input.

located at the extremities of the register. This simplifies the implementation significantly.

3 Implementation

We are arriving at the implementation of all this stuff. At last! The simplicity of the implementation is in part due to the choice of an exponent below 32, making it possible to represent the complete state of the register with a single count register. Since the character @ is used in internal macros, don't forget the catcode change with `\makeatletter` or `\catcode'\@=11` and changing it back afterwards.

```
\newcount\@SR
```

Furthermore we need a constant, necessary for handling the case where a 1-bit fills the vacancy in the rightmost stage. Our choice $n = 22$ dictates the value $2^{21} = 2,097,152$.

```
\def\@SRconst{2097152}
```

Initialization of the stream is done by simply setting the count register (globally) to the intended start value. Keeping this value between 1 and 4,194,303 can be left as the responsibility of the user.³

```
\def\@SRset#1{\global\@SR#1\relax}
```

Each step in the register cycle needs the calculation of the exclusive-or of the stages having a tap. The form of the characteristic polynomial chosen confines this to the bits corresponding to x^{21} and x^0 . Intricate calculations are therefore not needed. The value of the tap at the highest coefficient can be tested by comparing the register contents with the constant `\@SRconst` and jotting down the result in a scratch register. With `\ifodd` we take a look at the parity of the state which provides the value for x^0 . A division by 2 then conveniently shifts the contents of all stages one place to the left. We place a 1 in the highest stage by adding `\@SRconst`, if there is an odd number of 1's in the two taps examined. Finally note that the new status is assigned `\global` and that the whole process is enclosed in a group which localizes the changes to the scratch register.

```
\def\@SRadvance{\begingroup
  % examine value of highest tap
  \ifnum\@SR<\@SRconst\relax \count@=0
  \else \count@=1
  \fi
  % examine value of lowest tap
  \ifodd\@SR \advance\count@ by 1 \fi
  % all stages advance
  \global\divide\@SR by 2
  % place 1 in highest stage
```

³ It is not difficult to write a macro that takes for its argument the value modulo 4,194,304 but one has to be careful not to end up with the null cycle.

```
\ifodd\count@
  \global\advance\@SR\@SRconst\relax
\fi
\endgroup}
```

Production of an output bit and advancing the register one step is done by:

```
\def\@SRbit{\@SRadvance
  \ifodd\@SR 1\else 0\fi
}
```

The bit thus produced can be used in decision making. An example is the macro below which chooses between its first and second argument on the value of the next output bit of the register. With it we can write

```
\SRtest{choice 1}{choice 2}
```

and effect a random choice between the arguments. The implementation of `\SRtest` is

```
\def\@SRtest#1#2{%
  \@SRadvance
  \ifodd\@SR #1\else #2\fi
}
```

Another application is the permutation of items. Two items will be randomly interchanged by

```
\def\@permtwo#1#2{\@SRtest{#1#2}{#2#1}}
```

Those who are interested in the current value of the register state can obtain this by looking at the count register:

```
\def\@SRvalue{\number\@SR}
```

References

- [1] Golomb, S.W. 1967 *Shift Register Sequences*, San Francisco: Holden-Day.

◊ Hans van der Meer
University of Amsterdam
Faculty of Mathematics and
Computer Science
Plantage Muidersgracht 24
1018 TV Amsterdam
Netherlands
hansm@fwi.uva.nl

Slanted lines with controlled thickness

David Salomon and Matthew N. Hendryx

So far as the authors know, the original idea for a slanted line is due to A. Hendrickson (Ref. 1). The idea was to typeset a period, to move a small step in the desired direction, and to repeat the process. Some improvements to the basic idea are described in Ref. 2, together with the observation that better results can be obtained by typesetting a small rule, since the size of a rule can be precisely controlled, and can be adapted to the specific printer used.

`PiCTEX`, by M. Wichura (Ref. 3), is an excellent macro package for drawing diagrams in `plain TEX`. It uses the basic idea of typesetting a period and moving it. The article by Wichura does not explain the plotting algorithms used by `PiCTEX`, except to say that linear and quadratic interpolation are used.

One problem with the traditional method is the lack of control over the thickness of the resulting line. In a high-quality diagram containing horizontal, vertical and slanted lines, the thickness of all lines should be the same.

The principle

The method described here makes it possible to typeset slanted lines of any thickness by typesetting a rule, shifting it in the desired direction, and repeating the process a number of times (Fig. 1).

The user specifies the following four quantities:

- 1-2. X and Y , the horizontal and vertical displacements of the line, respectively (Fig. 1).
3. The thickness, t , of the line (Fig. 2).
4. The height, h , of a rule. This is determined by the printer used. For a given printer, the same value of h is used for all slanted lines.

The program has to calculate three values:

1. The width, b , of a single rule (Fig. 1).
2. The amount i by which each rule is shifted relative to its predecessor (Fig. 1).
3. The number r of rules necessary to get a complete slanted line (register `\rep` in the macros below).

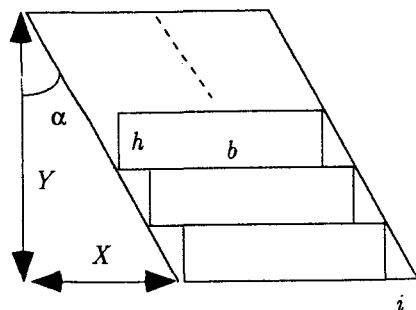


Fig. 1

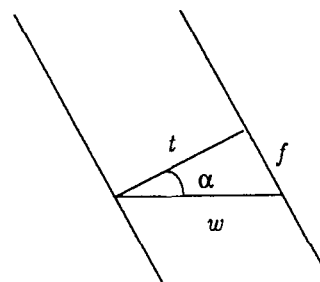


Fig. 2

The derivation

The three quantities are derived from elementary trigonometry. From Fig. 1 we get $\tan \alpha = X/Y$ and also $\tan \alpha = i/h$ or $i = h \tan \alpha$. From Fig. 2, $f^2 + t^2 = w^2$ and also $\sin \alpha = f/w$ or $f = w \sin \alpha$ which, in turn, implies $w^2 \sin^2 \alpha + t^2 = w^2$ or $t^2 = w^2(1 - \sin^2 \alpha) = w^2 \cos^2 \alpha = w^2/(1 + \tan^2 \alpha)$. (The last step uses the identity $\sin^2 \alpha + \cos^2 \alpha = 1$.) Since w is defined as $b + i$, we get $b = t\sqrt{1 + \tan^2 \alpha} - i$. The number r of necessary rules is easily seen to be $(Y/h) - 1$.

To summarize, the quantities X , Y , t , and h are given. From them, the three quantities i , b and r should be calculated by: $i = h \tan \alpha = hX/Y$, $b = t\sqrt{1 + \tan^2 \alpha} - i = t\sqrt{1 + (X/Y)^2} - i$ and $r = (Y/h) - 1$.

The only problem is the square root calculation. this calculation is easy to perform using Newton's method but, because of the limited precision of `TEX`, the results are often imprecise (which does not seem to affect the quality of the final lines by much). Here are the details of Newton's method.

Square root calculation

Newton's method for finding a root of a given function $f(x)$, is iterative. One starts with a first approximation x_0 (usually a guess), and performs the iteration $x_{i+1} \leftarrow x_i - f(x_i)/f'(x_i)$, $i = 0, 1, \dots$

To adapt the method for square root calculation, we select the function $f(x) = x^2 - n$. Clearly, any root of this function equals \sqrt{n} . Since $f'(x) = 2x$, the iterations above become

$$x_{i+1} \leftarrow x_i - \frac{x_i^2 - n}{2x_i} = \frac{1}{2} \left(x_i + \frac{n}{x_i} \right), \quad i = 0, 1, \dots$$

A good guess for x_0 is $n/2$, and 3 or 4 iterations are usually sufficient to get within 1% of the right value. We use eight iterations, to get better precision for small values of n .

Since the calculations involve non-integers, a `TEX` implementation should use `\dimen` registers. Since the calculations involve division, `\count` registers are also necessary. In the macros below, the `\dimen` register `\nn` stands for n , `\xx` stands for x_i

and $\backslash yy$, for x_{i+1} . The final result is returned in $\backslash yy$. The calculation is straightforward except for two points:

1. The ‘ $\backslash multiply\backslash yy$ by 100’ is necessary since otherwise the division that follows ($\backslash divide\backslash yy$ by $\backslash xx$) would result in a truncated quotient.

2. The ‘ $\backslash multiply\backslash yy$ by 655’ is necessary to scale the value of $\backslash yy$ from scaled points to points. The full factor is 65536 but we use 655 because of the previous multiplication by 100. The macros are:

```
\newdimen\mn \newcount\xx \newdimen\yy
\def\sqrt#1{\nn=#1 \xx=\nn \divide\xx by2
\iter\iter\iter\iter\iter\iter\iter\iter}
\def\iter{\yy=\nn \multiply\yy 100
\divide\yy by\xx \multiply\yy by 655
\advance\yy by\xx sp \divide\yy by2
\xx=\yy}
```

A typical expansion is $\backslash sqrt\{.8in\}$. Following which, the command ‘ $\backslash the\backslash yy$ ’ produces ‘7.59865pt’, less than 1% away from the true value. Note that, because of the limited arithmetic capabilities of T_EX, values over 163pt cause an arithmetic overflow. For small values, more iterations may be necessary. For example $\backslash sqrt\{50sp\}$ produces 0.125pt after 4 iterations, 0.036pt after 6 iterations, and 0.024pt after 8 iterations. The correct value is close to 0.0276pt. A more robust square-root macro is presented in a later section.

The result


The final macro, $\backslash slant$, takes three parameters, the quantities X , Y and t above. It creates the rules in $\backslash box\backslash slnt$ whose width is (Fig. 1) $X + b + i$ and whose height is Y . The user can then typeset the box in any desired way.


Note that the height of an individual rule is not a parameter of $\backslash slant$ but must be assigned explicitly to the $\backslash dimen$ register $\backslash hh$ before $\backslash slant$ is expanded. This is because our experience shows that, in practice, the height of the rules that make up the slanted lines depends on the printer used, and is thus the same for all slanted lines in the document. It is easy, of course, to specify the height as a parameter (#4) of $\backslash slant$, if desired. The macro should simply say ‘ $\backslash ttt=#3 \backslash ii=#2 \backslash auxi=#4$ ’.

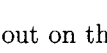
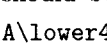
The macro has four parts. The first three calculate r (in register $\backslash rep$), i and b . Part four creates the rules.

```
\newdimen\ttt \newdimen\hh \newdimen\ii
\newdimen\bb \newdimen\tga
\newcount\auxi \newcount\rep \newbox\slnt

% Part 1. Number of repetitions
\def\slant#1#2#3{\ttt=#3 \ii=#2 \auxi=\hh
\multiply\ii by655 \divide\ii by\auxi
\multiply\ii by100
\rep=\ii \divide\rep by65536 % \rep:=Y/h-1
% Part 2. i=h tan a
\tga=#1 \ii=#2 \auxi=\ii
\multiply\tga by655 \divide\tga by\auxi
\multiply\tga by100 \auxi=\tga \ii=\hh
\divide\ii by655 \multiply\ii by\auxi
\divide\ii by100 % i=h tan a
% Part 3. b:=t\sqrt{tan^2a +1}-i
\divide\auxi by10 \multiply\auxi by\auxi
\divide\auxi by655 % tan^2a
\tga=\auxi sp \advance\tga by1pt
\sqrt{\the\tga}% \sqrt{tan^2a +1}
\auxi=\yy \divide\auxi by655 \bb=\ttt
\multiply\bb by\auxi \divide\bb by100
\ifdim\ii>0pt\advance\bb by-\ii
\else\advance\bb by\ii\fi % b:=t\sqrt{...}-i
\tga=0pt % Part 4. Build the rules
\setbox\slnt=\vbox{\offinterlineskip
\loop \ifnum\rep>0\advance\rep-1
\hbox{\kern\tga\vrule width\bb height\hh}%
\advance\tga by\ii
\repeat}}
```

A typical expansion is: $\backslash hh=.3pt$
 $\backslash slant\{15pt\}\{15pt\}\{1pt\}A\backslash box\backslash slnt B$,
 which results in  Opt plus 1em

 Opt plus 1em (the boundaries of $\backslash box\backslash slnt$ are shown for illustration purposes). The macro works for a negative X , but the width of $\backslash box\backslash slnt$ in this case is b , and the line sticks

out on the left. A typical example is  CD , created by $\backslash slant\{-15pt\}\{15pt\}\{1pt\}C\backslash box\backslash slnt D$. The macro does not work for negative values of Y . To create slanted lines that go below the text, a $\backslash lower$ should be used. Thus $\backslash slant\{15pt\}\{15pt\}\{1pt\}A\backslash lower4pt\backslash box\backslash slnt B$, creates  B .

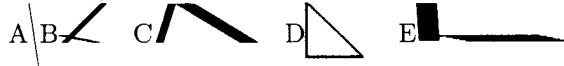
Other examples are:

```
\hh=.3pt
A\slant{4pt}\{25pt\}\{.4pt\}\lower10pt\backslash slnt
B\slant{15pt}\{3pt\}\{1pt\}\backslash box\backslash slnt
\slant{-15pt}\{15pt\}\{2pt\}\backslash box\backslash slnt
\quad
C\kern6pt\slant{-5pt}\{15pt\}\{3pt\}\backslash box\backslash slnt
\slant{25pt}\{15pt\}\{4pt\}\backslash box\backslash slnt
\quad
```

```
\hh=.5pt
D\lower6pt\hbox{\vrule height21pt width1pt
 \slant{21pt}{21pt}{1pt}\wd\slnt=0pt\box\slnt
 \vrule width21pt height1pt depth0pt}
\quad
```

```
E\slant{1pt}{15pt}{8pt}\box\slnt
 \slant{15pt}{3pt}{8pt}\box\slnt
```

resulting in



Note that the last example is wrong. We ask for a slanted line with $Y = 3\text{pt}$ and a thickness of 8pt , which is impossible. The line came out with a thickness of 3pt . The reader should compare this line with, e.g., `\slant{15pt}{13pt}{8pt}\box\slnt`, which has the right thickness of 8pt .

Large values of `\hh` can create nice patterns (try `\hh=4pt\slant{18pt}{18pt}{6pt}\box\slnt`), but experience shows that values around 0.3pt – 0.5pt are best for a typical 300dpi laser printer.

Possible improvements

1. The square root algorithm used here is iterative. It turns out that, for lines with slants close to 45° , 3 or 4 iterations are enough to get to within 1% of the correct square root. For slants closer to 0° or to 90° , more iterations are necessary. In a document with many slanted lines, it is possible to speed up the macros by changing the number of iterations depending on the slant, because both $\tan \alpha$ and $\cot \alpha$ are available.

2. The square-root macro presented earlier is simple and fast, but is not robust. It causes an arithmetic overflow for values over 163pt . The macro shown below is slower and more complex, but produces results accurate to 28 significant bits. This macro is the one used by METAFONT to calculate square-roots (Ref. 4, sections 121–123). It has been translated from WEB to T_EX, and has been provided to us by the referee.

```
\def\incr#1{\advance#1 by 1 }
\def\decr#1{\advance#1 by -1 }
\def\half#1{\divide#1 by 2 }
\def\double#1{\multiply#1 by 2 }
\def\fractiontwo{536870912 } % 2^29,
% represents 2.000000000
\def\fractionfour{1073741824 } % 2^30,
% represents 4.000000000

% Find the square root of #1 placing the
% results in \yy.
% The square root is in scaled numbers
% which are integer representations
```

```
% of numbers multiplied by 2^{-16}.
% That is, 1 = 65536, 2=131072,
% and so on. This uses Newton
% approximation, with shifts to
% preserve accuracy.
```

```
\newcount\k \newcount\y
\newcount\q \newcount\x
\newdimen\yy
```

```
\def\sqrt#1{\yy=#1 \x=\yy
 \ifnum\x>0
 \k=23 \q=2
 \loop\relax
 \ifnum\x<\fractiontwo
 \decr\k \multiply\x by 4
 \repeat
 %
 \ifnum\x<\fractionfour \y=0
 \else
 \advance\x by -\fractionfour \y=1
 \fi
 %
 % Decrease k by 1, maintaining the
 % invariant relations between x, y & q
 \loop
 \double\x \double\y
 \ifnum\x<\fractionfour \else
 \advance\x by -\fractionfour
 \incr\y
 \fi
 \double\x
 \advance\y by \y \advance\y by -\q
 \double\q
 \ifnum \x<\fractionfour \else
 \advance\x by -\fractionfour
 \incr\y
 \fi
 \ifnum\y>\q
 \advance\y by -\q \advance\q by 2
 \else\ifnum \y > 0 \else
 \advance\q by -2 \advance\y by \q
 \fi\fi
 \decr\k
 \ifnum\k>0\relax
 \repeat
 %
 \half\q \global\yy=\q sp %final result
 %
 \else
 % case of non-positive argument
 \ifnum\x<0
 \message
 {sqrt of #1 has been replaced by 0}
```

```
\fi
\global\yy=0 sp
\fi}}
```

References

1. Hendrickson, A., *Some Diagonal Line Hacks*, TUGboat 6(2), 83-86 (July 1985).
2. Salomon, D., *DDA Methods in T_EX*, TUGboat 10(2), 207-217 (July 1989).
3. Wichura, M., *PiCT_EX: Macros for Drawing Pictures*, TUGboat 9(2), 193-197 (July 1988). (The actual macros are available from the various T_EX archives.)
4. Knuth, D., *The METAFONTbook*, Reading, MA, Addison-Wesley, 1986.

◇ David Salomon
California State University,
Northridge
Computer Science Department
Northridge, CA 91330-8281
dxs@secs.csun.edu

◇ Matthew N. Hendryx
P. O. Box 218
North Manchester, IN 46962

Letters

On indexing and errors

In his letter (*TUGboat* 14, no. 2, page 141) Lincoln Durst finds fault with my article (*TUGboat* 13, no. 4, page 495) which in turn found fault in his earlier article (*TUGboat* 12, no. 2, pages 248-52).

However, I still hold to my original criticism, which was that the code of Durst would on specified occasions produce 0010 in an index file when in fact 010 is required. No criticism of Knuth's code was intended, nor I believe made, in my article.

It is easy to make an error. Harder is to find error in the work of another, and harder yet in one's own work. But most difficult, I have found, is to express and accept criticism in a friendly and respectful manner.

Yours sincerely,
Jonathan Fine
203 Coldhams Lane
Cambridge CB1 3HY, England
J.Fine@pmms.cam.ac.uk

L^AT_EX

L^AT_EX_{2 ϵ} update, dateline: 31 January 1994

Chris Rowley

As I am sure you are aware by now, L^AT_EX_{2 ϵ} will soon be the standard version of L^AT_EX — prepared and supported by the L^AT_EX₃ Project Team. An explanation of the need for this new system appeared in a recent issue of *T_EX and TUG NEWS*.

The test release, which hit the archives on December 21 — just in time to give the dedicated something to do over the holidays, turned out to be far more robust than we (the L^AT_EX₃ project team) had dared to hope. This is not to say that there were no problems but the testing so far has produced only about half-a-dozen clear bugs. Nevertheless, there is still, as I write, a lot to do before we have a complete system which can rapidly become the standard L^AT_EX for everyone, everywhere.

Perhaps the biggest sigh of relief came when Leslie Lamport reported that he had successfully used the new system — to format ‘The Manual’ (the old text, that is, using the compatibility mode provided by L^AT_EX_{2 ϵ}). Since then he has been working on the new edition and this work, together with that of many others, has been useful in exposing some of the rough edges which still need smoothing. This new edition of *L^AT_EX: A document preparation system* will, we hope, be published at about the same time as the full release of the new L^AT_EX becomes available; meanwhile, an expanded description of the system (covering both old and new features), together with a gold-mine of information about exciting things to do with it, can be found in *The L^AT_EX Companion*, by Michel Goossens, Frank Mittelbach and Alexander Samarin, Addison-Wesley, 1993, ISBN 0-201-54199-8.

Many of the reports we have been getting back from the testers have illuminated features of the old L^AT_EX which need attention but are not directly related to the changes made for L^AT_EX_{2 ϵ} . Some of the problems that have been raised will have to wait until L^AT_EX₃ for a solution but others may be attended to sooner than that. They have also shown that there are tensions between the different uses, and users, of L^AT_EX which make the important task of providing a standard system difficult — both technically and diplomatically. Some examples are: portability of documents versus local requirements (e.g. language, or graphics inclusion); authors versus publishers. If you are interested in an electronic

discussion of such issues (and many others), particularly in the context of looking forward to L^AT_EX₃, then you should join the L^AT_EX-L discussion list by sending:

message: SUBSCRIBE LATEX-L

to:

LISTSERV@vm.hd-net.uni-heidelberg.de

Thus the continuing work of the project team is not just a matter of fixing bugs, improving the efficiency of the system and simplifying the installation process; we are also, where possible, adding flexibility and ensuring a smooth transition for everyone to the new standard L^AT_EX when the full system is released in the Spring. It will then be available for ftp access in the following subdirectory of the CTAN archives:

tex-archive/macros/latex/core

We also expect that it will be available from the TUG office, from other T_EX User Groups throughout the world and, of course, with all good commercial T_EX systems. Part of the distribution is a file named `features.tex`; this can be typeset by any version of L^AT_EX and contains a brief outline of all the new features.

When we say that a ‘full system’ is being prepared, this does not just refer to our work on the core system; it also includes the many L^AT_EX-based packages on the CTAN archives which will be updated, by whoever supports them, so as to ensure that their material will continue to be widely available to L^AT_EX users. If you are one of these supporters and are wondering what you therefore need to do then don’t panic: we are sure that, in the majority of cases, it will be very little! If you need further information about what needs to be done then please send a message to the following address:

bugs@minnie.zdv.uni-mainz.de

Please ensure that it starts with the following line **exactly**:

>Category: latex-class-writing

If you want a prompt and useful reply, please make your message as short and precise as possible.

To summarise: thanks to a lot of support and the very enthusiastic response we have had from everyone involved, we are confident that the decision taken last Spring to produce L^AT_EX_{2 ϵ} was a good one, and that this Spring will see its fulfilment. Then it is ‘onwards and upwards to L^AT_EX₃!’.

◊ Chris Rowley
 Open University
 Walton Hall
 Milton Keynes MK7 6AA
 United Kingdom
 c.a.rowley@vax.acs.open.ac.uk

Addenda: A suggested “operational requirement” for L^AT_EX3’s treatment of bibliographic references

David Rhead

[Editor’s note: The original article with this title appeared in TUGboat 14, no. 4, December 1994, pp. 425–433. Owing to a production error, the version that appeared was the unrefereed original. The most significant changes made in response to the referees’ comments appear below. The full text of the revised version can be found on a CTAN node as `tex-archive/digests/tugboat/articles/14-4/rhead.ltx`; the associated references are in `rhead.bib`. Owing to the timeliness of the material, these files have been assigned a deletion date corresponding to the release of L^AT_EX3.

The Editor regrets the error.]

2 Doing it yourself

[New subsection; insert at the end of section 2.]

Multi-author documents

I think it desirable that L^AT_EX3’s successors to the “standard styles” should support multi-author documents (e.g., a journal-issue made up of a number of articles, or a conference-proceedings made up of several contributions).

Hence:

- it should be possible to have several reference-lists within a single document
- there should be allowance for the possibility that a single document may use two or more citation schemes. E.g., since the “instructions for authors” in *Mathematische Zeitschrift* gives a choice of three citation schemes, an issue of the journal may involve three distinct schemes.

3 Using bibliography-formatting software

[New subsection; adjust numbering.]

3.1.5 Hybrid approaches 2

Other hybrid approaches might use a proprietary system and B_IB_TE_X “in series”:

- One might regard the proprietary system’s database as a “staging post”, where information stays briefly before being converted to a B_IB_TE_X database. For example, if a proprietary system can import from library catalogues and export to a B_IB_TE_X .bib file, the approach gives a mechanism for getting information from library catalogues to B_IB_TE_X.
- Alternatively, one might regard the B_IB_TE_X database as the “staging post”. If a proprietary

system exports a B_IB_TE_X .bib file, information held in the proprietary database can be converted to a B_IB_TE_X database just before being used in conjunction with L^AT_EX.

The following problems are likely to arise with such approaches:

- The standard “L^AT_EX, B_IB_TE_X, L^AT_EX, L^AT_EX” sequence is already fairly laborious. An additional (“proprietary database to B_IB_TE_X database”) stage will make things worse.
- Questions could arise about “which database is the definitive, up-to-date one — the proprietary system’s or B_IB_TE_X’s?”
- Mapping problems could arise. The usual B_IB_TE_X analysis of structure (in terms of entry-types and fields) differs from those used by other systems. In literature-areas where the B_IB_TE_X analysis is relatively coarse, subtleties will be lost if a finer analysis is mapped to the B_IB_TE_X analysis (e.g., if Library Master’s public document, manuscript collection, computer program, audio recording, video recording, interview, and artwork record-types are all converted to @MISC).
- Documentation may be cumbersome, since the end-user will have to consult that for the proprietary system, that for the conversion procedure, and that for B_IB_TE_X. The user will also need to understand the two lots of terminology, and be able to “translate” from one to the other.

Because of these potential problems, I’m not inclined to pursue this type of hybrid approach either.

3.1.6 The user’s choice

Given some *modus vivendi*, end-users would be able to make their own assessments of which bibliographic software suits their needs.

- Someone who wants ready-made methods of downloading information from commercial bibliographic databases, CD-ROMs, library catalogues, etc., will probably favour one of the proprietary programs. The proprietary systems also offer database administration and searching facilities.
- Different systems implement different analyses of the structure of “the literature” (i.e., using B_IB_TE_X’s terminology, there are different ways of defining entry-types and fields), and different people also have different viewpoints. E.g., an analysis that suits a scientist may be too coarse for keeping track of “primary sources” in the humanities.
- Cost is obviously a factor.

[Add the following at the end of the list.]

- Some software supports “imprecise citations” (e.g., “the item in my database whose author is ... which has ... in the title”). Others, such as `BIBTEX`, require a precise citation via a unique key. People who are continually adding items to their bibliographic databases may prefer the latter, so as to avoid situations in which a match becomes ambiguous even though a document’s text has not changed.

4 Miscellaneous

[Revised subsection.]

4.1 “Local names” for keys

If you are “doing it yourself”, choice of keys (i.e., in `LATEX 2.09` terms, the arguments for `\bibitem`) is unlikely to be a problem. For example, you could equally well use `lamport-86` or `latexbook` as a key for the `LATEX` manual. There is no particular need for consistency from one document to another: you can use `lamport-86` as the key in one document, and use `latexbook` as the key in another.

However, if you have a large bibliographic database (perhaps shared with a group of colleagues), it may be impracticable to keep track of keys assigned on an *ad hoc* basis, and difficult to guarantee that keys will stay unique whenever a new item is added to the database.

Moreover, a `.tex` file to be `\input` may contain bibliographic details and `LATEX` commands that are generated automatically by bibliographic software (even though `LATEX` will have no way of distinguishing the file from one that you might produce when “doing it yourself”). Such bibliographic software might be programmed to assign keys automatically, e.g.,

- based on the ISBN, in the case of books
- of the form `journal-volume-number-page`, in the case of journal-articles
- based on “record number”, if the bibliographic software assigns a unique number to each record in the database
- of the form `lamport-86`, constructed automatically from the “author” and “year” fields in the database.

There may be a dilemma about whether to have automatically assigned keys that are relatively easy-to-remember, or to have keys that are guaranteed to stay unique no matter what else gets added to the database. As an example, consider what key might be used for the `LATEX 2.09` manual: `lamport-86` is easy to remember, but is potentially ambiguous (because `Lamport` published other work in 1986); if the

ISBN 0-201-15790-X was used as a key, it should stay unique but would be difficult to remember.

To help cater for such situations, it might be useful if `LATEX3` allowed “local names” for keys, i.e., some mechanism whereby an author could declare (e.g., in a document’s root file) that, for the duration of a document, a particular “informal key” (to be used in in-text citation commands) should be treated as a synonym for a “formal key” (which appears in an entry in an automatically generated reference-list). For example, it might be useful to be able to declare that `lamport-86` can be used as a “local name” for 0-201-15790-X.

5 Acknowledgement

I am grateful to the referees for their comments on a draft of this paper. The paper incorporates a number of ideas taken from the referees’ comments.

◊ David Rhead
 Cripps Computing Centre
 University of Nottingham
 Nottingham NG7 2RD
 United Kingdom
 d.rhead@vme.nott.ac.uk

<h2>Calendar</h2>

1994				
Apr 11–15	Four conferences, Darmstadt, Germany:			<ul style="list-style-type: none"> • EP94, Electronic Publishing, Document Manipulation and Typography (for information, contact ep94@gmd.de); • RIDT94, Raster Imaging and Digital Typography (for information, contact ridt94@irisa.fr); • TEP94, Teaching Electronic Publishing (for information, contact ltsdyson@reading.ac.uk); • PODP94, Principles of Document Processing (for information, contact podp94@cs.umd.edu).
Apr 18	T _E X–Stammtisch in Bonn, Germany. For information, contact Herbert Framke (Herbert_Framke@BN.MAUS.DE ; telephone 02241 400018). Third Monday, Anno, Kölnstraße 47.	May 5		T _E X–Stammtisch at the Universität Bremen, Germany. For information, contact Martin Schröder (115d@zfn.uni-bremen.de ; telephone 0421/628813). First Thursday, 18:30, Universität Bremen MZH, 4 th floor, across from the elevator.
Apr 19	T _E X–Stammtisch in Duisburg, Germany. For information, contact Friedhelm Sowa (tex@ze8.rz.uni-duesseldorf.de ; telephone 0211/311 3913). Third Tuesday, 17:30, at Gatz an der Kö, Königstraße.	May 16		T _E X–Stammtisch in Bonn, Germany. (For contact information, see Apr 18.)
Apr 27	T _E X–Stammtisch, Hamburg, Germany. For information, contact Reinhard Zierke (zierke@informatik.uni-hamburg.de ; telephone (040) 54715-295). Last Wednesday, 18:00, at TEX's Bar-B-Q, Grindelallee 31.	May 17		T _E X–Stammtisch in Duisburg, Germany. (For contact information, see Apr 19.)
Apr 30– May 2	BachoT _E X '94, Bachotek, Poland. 2 nd General Meeting of GUST, the Polish T _E X Users Group. For information, contact Hanna Kołodziejska (hkolo@plearn.edu.pl).	May 25		T _E X–Stammtisch, Hamburg, Germany. (For contact information, see Apr 27.)
		Jun 2		L ^A T _E X _{2ϵ} conference and GUTenberg AGM, Paris. Presenters: Michel Goossens and Frank Mittelbach. For information, contact tresorerie.gutenberg@ens.fr .
		Jun 2		T _E X–Stammtisch at the Universität Bremen, Germany. (For contact information, see May 5.)
		Jun 6		TUGboat Volume 15, 2nd regular issue: Mailing date (tentative).
		Jun 8–10		Society for Scholarly Publishing (SSP), 16 th Annual Meeting, San Francisco, California. For information, contact the SSP office (303-422-3914; fax: 303-422-8894).
		Jun 9–10		NTG 13 th Meeting, “(L ^A)T _E X, METAFONT, and tools education”, Groningen, at RUG. June 10: “4T _E X” course, taught by Wietse Doi and Erik Frambach. For information, contact Gerard van Nes (vannes@ecn.nl).
		Jun 20		T _E X–Stammtisch in Bonn, Germany. (For contact information, see Apr 18.)

- Jun 21 T_EX–Stammtisch in Duisburg, Germany. (For contact information, see Apr 19.)
- Jun 29 T_EX–Stammtisch, Hamburg, Germany. (For contact information, see Apr 27.)
- Jul 7 T_EX–Stammtisch at the Universität Bremen, Germany. (For contact information, see May 5.)
- Jul 6–8 C.N.E.D. 94: 3ième Colloque National sur l'Écrit et le Document, Rouen, France. For information, contact Jacques Labiche (labiche@la3i.univ-rouen.fr).
- Jul 11 UK T_EX Users' Group, Cambridge University. L^AT_EX fonts and graphics: a hands-on tutorial. For information, e-mail uktug-enquiries@ftp.tex.ac.uk
- Jul 18 T_EX–Stammtisch in Bonn, Germany. (For contact information, see Apr 18.)
- Jul 19 T_EX–Stammtisch in Duisburg, Germany. (For contact information, see Apr 19.)
- Jul 24–29 SIGGRAPH'94: 21st International ACM Conference on Computer Graphics and Interactive Techniques. Orlando, Florida. (For information, contact siggraph-94@siggraph.org, telephone 312-321-6830.)
- Jul 27 T_EX–Stammtisch, Hamburg, Germany. (For contact information, see Apr 27.)
- Jul 31–
Aug 4 **TUG 15th Annual Meeting:** Innovation, Santa Barbara, California. For information, contact Debbie Ceder (tug94@tug.org). (For the preliminary program, see p. 68.)
- Aug 4 T_EX–Stammtisch at the Universität Bremen, Germany. (For contact information, see May 5.)
- Aug 15 T_EX–Stammtisch in Bonn, Germany. (For contact information, see Apr 18.)
- Aug 16 T_EX–Stammtisch in Duisburg, Germany. (For contact information, see Apr 19.)
- Aug 17 **TUGboat Volume 15, 3rd regular issue:** Deadline for receipt of *technical* manuscripts (tentative).
- Aug 31 T_EX–Stammtisch, Hamburg, Germany. (For contact information, see Apr 27.)
- Sep 14 **TUGboat Volume 15, 3rd regular issue:** Deadline for receipt of news items, reports (tentative).
- Sep 26–30 EuroT_EX '94, Sobieszewo, Poland. For information, contact Wlodek Bzyl (EuroTeX@Halina.Univ.Gda.Pl). (See announcement, p. 69.)
- Oct 19 UK T_EX Users' Group, Aston University. Annual General Meeting. For information, e-mail uktug-enquiries@ftp.tex.ac.uk
- Nov–Dec UK T_EX Users' Group, location to be announced. Topic: T_EX, SGML and electronic publishing. For information, e-mail uktug-enquiries@ftp.tex.ac.uk
- Nov 23 **TUGboat Volume 15, 3rd regular issue:** Mailing date (tentative).
- 1995**
- Jan 5–8 Linguistic Society of America, 69th Annual Meeting, Fairmont Hotel, New Orleans. For information, contact the LSA office, Washington, DC (202-834-1714, zzlsa@gallua.gallaudet.edu).
- Apr UK T_EX Users' Group, location to be announced. Topic: Maths is what T_EX does best of all. For information, e-mail uktug-enquiries@ftp.tex.ac.uk
- For additional information on the events listed above, contact the TUG office (805-963-1338, fax: 805-963-8358, email: tug@tug.org) unless otherwise noted.

TUG '94**Announcement and Preliminary Program
Santa Barbara, California
31 July–4 August 1994**

The T_EX Users Group is proud to announce its **fifteenth** annual meeting. This year's theme will be "Innovation". The meeting will be held in Santa Barbara, California, the home of the T_EX Users Group itself. We would like to extend a gracious invitation to T_EX users world-wide—come join us in what is sure to be quite an experience.

As usual, courses will be offered during the week preceding and the week following the conference.

Social functions are to include an opening reception, a conference banquet, and a number of culturally stimulating events that are sure to give everyone great pleasure. These events will invariably include the ever-enlightening night of bowling, so be prepared!

For the second year, there will be a Bursary Fund set up to assist T_EX users who demonstrate need, and/or have never attended a TUG meeting before, to participate in the conference. Owing to differences in exchange rates throughout the world, participants from Central and Eastern Europe in particular are expected to experience difficulty in raising funds to enable them to attend; however, the focus is not on any specific geographic region. All members are encouraged to consider contributing to the fund. The Bursary Fund Committee members are Bernard Gaulle and Norman Naugle, with Christina Thiele as liaison to the board of directors.

If you would like to attend the conference and/or any of the courses, or to obtain more information about contributing to the Bursary Fund, please contact the TUG office by e-mail to tug@tug.org, or by post to the address given on the inside front cover.

If you wish to apply for a bursary, kindly send a letter to Bernard Gaulle, chair of the Bursary Committee, providing details of your requirements, including precise costs where possible. Bernard Gaulle: 4, avenue Cadoux-Girault, F-92270 Bois-Colombes, France. E-mail: gaulle@idris.fr.

The program

This preliminary program has been arranged by Sebastian Rahtz and Malcolm Clark, who comprise the Program Committee. Speakers, titles and specific dates are subject to change.

Sunday, July 30th*Publishing, Languages, Literature and Fonts*

- Glenn Reid: Reflections
- Frank Mittelbach: Real life book production — lessons learned from *The L^AT_EX Companion*
- Yannis Haralambous: Typesetting the holy Bible in Hebrew, with T_EX
- Basil Malyshev: Automatic conversion of METAFONT fonts to Type1 PostScript
- Alan Jeffrey: PostScript font support in L^AT_EX2_ε
- Yannis Haralambous: An Indic T_EX preprocessor — Sinhalese T_EX
- Michael Cohen: Zebrackets: a metaMETAFONT

Monday, August 1st*Color and L^AT_EX*

- Leslie Lamport: Looking back at, and forward from, L^AT_EX
- Tom Rokicki: Advanced 'special' support in a dvi driver
- Timothy van Zandt and Denis Girou: An introduction to PSTricks
- Sebastian Rahtz and Michel Goossens: Simple colour design in T_EX
- Michael Sofka: Color book production using T_EX
- James Hafner: The (pre)history of color in Rokicki's dvips
- Friedhelm Sowa: Printing colour pictures
- Angus Duggan: Colour separation and PostScript
- Jon Stenerson: A L^AT_EX style file generator
- Johannes Braams: Document classes and packages in L^AT_EX2_ε

Tuesday, August 2nd*T_EX Tools*

- Oren Patashnik: BIB_TE_X 1.0
- Minato Kawaguti and Norio Kitajima: Concurrent use of interactive T_EX previewer with an Emacs-type editor
- Yannis Haralambous: Humanist
- Pierre Mackay: A typesetter's toolkit

- Jean-luc Doumont: Pascal pretty-printing: an example of “preprocessing within T_EX”
- Michael P. Barnett and Kevin R. Perry: Symbolic computation for electronic publishing
- Norm Walsh: A World Wide Web interface to CTAN

Wednesday, August 3rd

Futures

- Chris Rowley and Frank Mittelbach: The Floating World
- Joachim Schrod: Towards interactivity for T_EX
- Arthur Ogawa: Object-oriented programming, descriptive markup, and T_EX
- William Erik Baxter: An object-oriented programming system in T_EX
- John Plaice: Progress in the Omega project
- Phil Taylor: e-T_EX & NTS: A progress report
- Jonathan Fine: Documents, compuscripts, programs and macros
- George Greenwade: T_EX as a commodity

Thursday, August 4th

Publishing and Design

- Maurice Laugier and Yannis Haralambous: T_EX innovations by the Louis-Jean printing house
- Michael Downes: Design by template in a production macro package
- Gabriel Valiente Feruglio: Macro packages for typesetting commutative diagrams
- Alan Hoenig: Less is More: Restricting T_EX's scope enables complex page layouts
- Don Hosek: Sophisticated page layout with T_EX
- Henry Baragar and Gail E. Harris: An example of a special purpose input language to L^AT_EX
- Marko Grobelnik, Dunja Mladenić, Darko Zupanič and Borut Žnidar: Integrated system for encyclopaedia typesetting based on T_EX
- Yannis Haralambous and John Plaice: First applications of Ω: Adobe Poetica, Arabic, Greek, Khmer

Posters, workshops and discussion sessions

The TUG WWW server (Peter Flynn)

Accessing CTAN (Norm Walsh)
 Practical indexing (Nelson Beebe)
 T_EX and linguistics (Christine Thiele)
 T_EX and humanities journals (Christine Thiele)
 Database publishing (Marko Grobelnik)
 Floats (David Salomon)
 Adobe Acrobat and related technology (Sebastian Rahtz)
 Standards for colour drivers (Tom Rokicki)

See you in Santa Barbara!

EuroT_EX '94

Is North America too far / too expensive / too alien a culture? Then come to EuroT_EX '94, the T_EX conference of the year. EuroT_EX will take place at Sobieszewo on an idyllic island off the coast of Gdansk in Poland. The conference will run from Monday September 26th to Friday September 30th, and the *maximum* cost (based on two persons sharing) will not exceed \$260-00 / sterling 175-00 / DM 450-00. Those arriving early on Monday will be able to take part in a guided tour of the old town of Gdansk, whilst Tuesday to Thursday will be jam-packed with talks and tutorials on T_EX and related topics.

All delegates will be accommodated in a single building, and for the whole week will be cut off from civilisation: no distractions, no need to leave the island; everything will be provided. For those unable to sustain the pace, quiet meditative walks along the shore searching for amber will provide the ideal opportunity for therapeutic meditation.

Papers are solicited *now*, and early registration for the conference is advised: with its Central European location and idyllic setting, the conference is expected to attract many delegates. If you are even *thinking* of coming, then you are urged to notify the Conference Organisers at the address below: we will then add you to the mailing list and keep you posted about any changes in the schedule.

Deadlines. Please note the following deadlines:

- 1 May 1994: Abstracts for papers
- 15 August 1994: Final papers
- 1 June 1994: Provisional registration (no charge for cancellation)

- 1 September 1994: Confirmed registration
(cancellation charged at
50%)
- 15 September 1994: Late registration (no
cancellation possible)

Bursaries. As with EuroTeX 92 and the TUG meeting at Aston last year, it is hoped to be able to offer financial assistance to delegates who would otherwise be unable to attend; of course, we cannot be sure at this stage that sponsors will be as generous as they have been in the past, but intending delegates who will need assistance in order to be able to attend should state the *minimum* bursary which would allow them to be able to attend, and should give clear reasons why they are applying. All applications will be treated in the strictest confidence. Delegates who are in no need of a bursary and who are able to assist others less fortunate are urged to pledge a donation.

Tutorials and Courses. It is intended to offer both tutorials (which will take place during the week of the conference proper), and courses (which will take place during the week following the conference). Proposed topics include book design and L^AT_EX 2_ε, but no firm decisions have yet been taken on topics, durations or costs. ('Tutorials' are usually of one day or less; 'courses' are usually of one day or more. Although no firm decisions have been taken at this stage, it is possible that tutorials will be free of charge whilst courses will be charged for. Every effort will be made to ensure that even charged-for courses are affordable and in line with local currency values.) Further details concerning this area will be circulated as soon as they are known. If you are interested in particular tutorials or courses, or wish to suggest topics, please communicate this to the Conference Organisers.

Address for further information.

Conference Organisers, EuroTeX '94
 % Włodek Bzyl
 Department of Mathematics
 University of Gdansk
 Wita Stwosza 57
 80-952 Gdańsk
 Poland
 E-mail: Włodek Bzyl (Mathematics)
 <eurotex@halina.univ.gda.pl>

When writing to the Conference Organisers, please state your name, full postal and e-mail addresses, phone and fax numbers. Use "EuroTeX '94" as the subject and include this text in the message: "Please add my name and address to the EuroTeX 94 mailing list and keep me posted of developments."

TUG Business

Meet the Board, Part I

The 1993 election for the Board of Directors did not attract as many candidates as there were open positions — the entire Board was up for election, and 15 is a rather large number. According to the TUG Election Procedures, when the number of candidates is fewer than the number of open positions, all candidates who have met the qualifications are to be declared elected by acclamation. As you have already read (*TEX and TUG NEWS* Vol. 2, No. 4, p. 21 and Christina Thiele's "Opening Words" in this issue), three additional positions have been filled by appointment.

But, . . . , this means that TUG members never had the opportunity to read the biographies and personal statements of the candidates. Without this information, it is difficult to know the particular interests of each, and what their vision is for the future of TUG.

The first installment of the statements is presented here, and the remainder will appear in the next issue. Take time to read them. If you have questions for any of these people, feel free to contact them. They have agreed, after all, to serve the interests of you, the members.

The statements appear in alphabetical order by name of the Board member, with the ending year of that person's term of office in parentheses after the name. Effective with this Board, terms of office end with the annual meeting of the year shown.

Barbara Beeton
For the Elections Committee

Barbara Beeton (1995)

American Mathematical Society
 P. O. Box 6248
 Providence, RI 02940 USA
 Phone: (401) 455-4014
 Internet: bnb@math.ams.org

Biography:

TUG: charter member of TUG; charter member of TUG Board of Directors; *TUGboat* production staff since 1980, Editor since 1983; committees: publications, bylaws, elections; chair, Technical Working Group on Extended Math Font Encoding; liaison from Board to Knuth Scholarship Committee 1991-1992

Employed by American Mathematical Society: Staff Specialist for Composition Systems; involved with typesetting of mathematical texts since 1973; assisted in initial installation of \TeX at AMS in 1979; implemented the first AMS document styles; created the map and ligature structure for AMS Cyrillic fonts

Standards organizations: active since 1986 in ANSI X3V1 (Text processing: Office & publishing systems), ISO/IEC JTC1/SC18/WG8 (Document description and processing languages); developing the standard ISO/IEC 9541:1991 Information technology — Font information interchange

AFII (Association for Font Information Interchange): Board of Directors, Secretary since 1988

Personal statement:

TUG has changed in the past few years, with its transition from an appointed to an elected Board. Those charged with shaping its future direction have tried to do so in a way that encourages participation by all members, not just a few. Similarly, the typographic landscape has changed as well, and though the object that is our focus — \TeX — is still a tool of undeniable utility, it is just part of a growing pool of text processing software, some of it borrowing from the features that first attracted us to \TeX . I maintain my commitment to Don Knuth's original goals for this tool: high typographic quality and portability. Within this framework, my goal is to continue working for unconstrained communication among \TeX users, to encourage exploration of techniques consistent with the typographic excellence we have come to expect, and to act as a historian of the \TeX community when that is appropriate.

Mimi Burbank (1996)

Supercomputer Computations Research
Institute,
Florida State University
Tallahassee, FL 32306-4052 USA
Phone: (904) 644-2440
FAX: (904) 644-0098
Internet: mimi@scri.fsu.edu

Biography:

My job over the past 8 years at the Supercomputer Computations Research Institute (SCRI) has evolved from technical typing to coordinating all publications efforts for a large research institute. We support a large community of research scientists (and their international group of collaborators), university faculty and administrative staff. We maintain a \TeX publications database, coordinate

conferences and publish proceedings, and coordinate the distribution of informational material to remote users at a large number of international sites. I also have had the opportunity to work with a large number of people from widely diverse cultural and scientific backgrounds.

My association with the \TeX Users Group began in 1985. I've attended quite a number of classes, have sponsored classes here at Florida State University (FSU), and for the past three years have worked as an editor/co-editor of the TUG Annual Proceedings issues. During the past year I've served on the TUG Conference Planning Committee and the Publications and Documentation Committee. I've corresponded with many TUG members electronically, and have met and talked with many of you at Annual TUG Meetings since 1986.

Personal statement:

As a board member I would be interested in actively involving members of our \TeX community in

- improving the lines of communication between users;
- establishing databases of information/sources of instructional information for new users, as well as \TeX wizards;
- promoting the dissemination of information to Local User Groups;
- the growth of \TeX and *TUG NEWS*;
- most importantly, the active and aggressive recruitment of new members; and
- providing a source of support for new users.

Jackie Damrau (1996)

[Editor's note: With the demise of the Super Collider project, Jackie has a new job. Her biography is what would have appeared on the ballot last fall, but the address below is new.]

P. O. Box 875
Red Oak, TX 75154-0875 USA
Phone: (214) 617-2323
Internet: damrau@amber.unm.edu

Biography:

I've been working at the Superconducting Super Collider Laboratory since 1989, where I directly support \TeX on various computer platforms. This includes collecting public domain and commercial \TeX and \TeX -related packages. I also have taught several in-house classes and brought outside consultants in to teach specialized \TeX training classes. As the \TeX support person, I answer staff questions on the use of \TeX , as well as locate specialized macro files for our users.

I have been active with the T_EX Users Group since 1985, and have attended seven of the last eight conferences. Currently I serve on the TTN editorial staff (where I'm one of four panel judges in the A-in-L^AT_EX Contest), and am Associate Editor of the L^AT_EX column for *TUGboat*. As well, I chair the Conference Planning Committee, which is working on providing documentation for site proposal bids, and for running meetings.

Personal statement:

As a Board member, I would be interested in:

- expanding the scope of T_EX users to other users groups, such as DECUS (DEC Users' Society) and STC (Society for Technical Communication);
- providing guidelines for future conference site proposals for successful users meetings;
- supporting the continuing growth of TTN; and
- recruiting existing members into volunteering their hidden talents to make the T_EX Users Group a continued success.

Luzia Dietsche (1997)

Universität Heidelberg
Im Neuenheimer Feld 293
D-69120 Heidelberg, Germany
Internet: x68@vm.urz.uni-heidelberg.de

Biography:

Luzia S. Dietsche was born in Freiburg/Germany where she received her education. In 1982 she began to study German and History, first in Freiburg then in Heidelberg. Since 1988 she is working in the computing center of the University of Heidelberg responsible for T_EX support, T_EX installation and documentation with and about T_EX. She teaches classes and advises users at the University.

In 1989, when DANTE e.V. (the German speaking T_EX users group) was formed, she was one of the founding members and elected first secretary, a position she has held ever since. She is also doing the job of a business manager for DANTE e.V., organizing management of membership, software distribution and working groups. Besides this she is involved in editing the journal of the association. DANTE e.V. increased rapidly since its foundation (more than 2500 members) and Luzia is the driving force behind this development.

Since 1991 she is a member of the first *Board of Directors* ever been elected by TUG members. She is involved in various committees, two of which are working on "Special Vice Presidents", and the "Promotions Committee" which tries to advance T_EX and TUG.

She is well known among T_EXies throughout the world for her knowledge as well as for her ability to carry through her plan and to accomplish her objective. Over the last years she has taken an important part in organizing all major T_EX-related events in the German speaking countries, and some outside as well.

Besides all that she is working for various publishers as a consultant and macro writer and helps to spread the knowledge about T_EX in the commercial world.

Personal statement:

Being in the board for two years, trying to help TUG to become reorganized, I feel it is time to establish a new TUG. This will lead to an increasing number of members, what is still most urgently needed. Another way to achieve a wider acceptance are better public relations and promotions of TUG and T_EX. This would not only help TUG, but also T_EX as a software product, in a market which is more and more dominated by other typesetting systems. To force this development it is necessary to make a better and versatile documentation available, for new as well as for advanced users. There is a need to provide more information on new developments, more services, coordination of existing and new interest groups, and getting new contacts to publishers and computer companies, ...

Michael Doob (1995)

Department of Math & Astronomy
University of Manitoba
Winnipeg R3T 2N2, Manitoba, Canada
204-474-9796
Internet: mdoob@ccu.umanitoba.ca

Personal statement:

I am a professor in the Mathematics Department. I do research in algebraic graph theory, and in the course of writing my first book in that area I became interested in mathematical typesetting. After an aesthetically disappointing first printing, I learned about T_EX and, after some exposure, what an extraordinarily beautiful tool it is. After writing two more books and editing two others, I believe that we have still just seen the tip of a fundamental change in mathematical communication. The presence of TUG, particularly on the Internet, can be of great value.

I am the T_EX editor for the Canadian Mathematical Society. Our journals are done completely in T_EX, sometimes from author manuscripts, sometimes from author-submitted T_EX source files. Some

of the most extraordinary but unfortunate constructions are used by authors. Many authors put in a great deal of (wasted) time and effort to format their documents; it all has to be undone, much to their disappointment. There is an educational job that needs to be done, and I would like to see TUG help with this.

The installation of \TeX can be painful; to some extent this is necessary with a complex program in a complex environment. I would like to see TUG help with scripts for generic installations of the freely available packages.

I attended my first TUG meeting in 1982. At that time the IBM PC was just a year old, there were no Macintoshes, there was no PostScript, there were very few UNIX workstations and almost no laser printers. The fact that \TeX is the major mathematical typesetting tool a dozen years later is a tribute to its flexibility. TUG has contributed to the adaptation of \TeX to the changing environment; I want it to continue to do so.

Michel Goossens (1995)

Text Processing Section
MI Group - AS Division
CERN
CH-1211 Geneva 23, Switzerland
Internet: goossens@cernvm.cern.ch

Biography:

After obtaining a PhD in high energy physics at the University of Brussels (Belgium), I joined CERN, the European Laboratory of Particle Physics in Geneva (Switzerland) in February 1979 as a research physicist. Realizing the importance of good documentation, I soon became active in the field of text processing and documentation.

In the course of my work I have come into contact with several text processing systems, from DCF/Script to groff, from mainframes to Macs—and, of course, \LaTeX . As a scientific institute, CERN physicists and engineers use mainly \LaTeX for publishing papers and writing documentation. Therefore, since 1988 I have become more and more involved in solving problems and developing tools related to \TeX , and especially \LaTeX . Realizing the importance of training I have taught several courses on \LaTeX at CERN and at \TeX conferences, and have written several articles on \LaTeX , both for CERN and in various journals. Together with Frank Mittelbach and Alexander Samarin I recently published "The \LaTeX Companion", which describes $\LaTeX 2_{\epsilon}$.

Interesting developments have taken place recently in the area of using \LaTeX document sources for generating hypertext documentation, and my present interest includes developing tools to easily translate \LaTeX into SGML-like tagging schemes, which form the basis of most hypertext systems, such as the popular WWW.

At the same time the issue of efficient, powerful and platform-independent graphics has become an important issue, and I also have an ongoing interest in general PostScript support to use that language as an efficient and portable transport format for all graphics and text processing applications.

Personal statement:

As a collaborator in a large international scientific organization, and thanks to my daily contacts with users from many countries on all continents, I feel I can make essential contributions in the areas of:

- real multinational and multi-language support for (L^A) \TeX ;
- training and documentation (in English and national languages);
- follow the developments of e- \TeX and NTS, especially the 16-bit variants to support non-Latin languages;
- cross-platform tests of new developments;
- interfaces to SGML/HTML/multimedia products;
- stimulate the use of PostScript as graphics lingua franca;
- make \TeX better known in the non-scientific sectors of activity (administration, humanities).

Tom Rokicki (1995)

Box 2081
Stanford, CA 94309 USA
Phone: (415) 322-6442
Internet: rokicki@cs.stanford.edu

Biography:

I was introduced to \TeX in 1983 at Texas A&M University, where I was studying electrical engineering. Under the guidance of Norman Naugle, I wrote several drivers and utilities, and wrote $\TeX 2_C$, the basis for the current popular web2c system on which most ports of \TeX are based.

In 1985, I joined the \TeX project at Stanford, where I designed the PK font file format and developed the PK utilities. As the proud owner of an Amiga in 1986, I implemented Amiga \TeX and its previewer and printer drivers, including dvips. The NeXT was another irresistible attraction, so I began NeXT \TeX in 1988. More recently, I had the

opportunity to assist Arvind Borde with \TeX Help, an on-line \TeX reference.

I am presently a member of the technical staff at Hewlett-Packard Laboratories in Palo Alto, California.

Personal statement:

Computing and printing environments have changed drastically since the inception of \TeX . As computer speeds and screen and printer resolutions have risen, so have the expectations of users. Where once users were awed by simple ligatures and kerns, now users expect four-color separations with fountains, chokes, and spreads. With \TeX essentially frozen, any new features must derive from preprocessors, postprocessors, and drivers. The establishment and adoption of implementable, extensible, powerful standards for these new features is essential to maintaining the portability of \TeX . As a board member of TUG, I intend to use my experience with the technical aspects of \TeX to help encourage the design, development, and adoption of standards for specials, graphics, color, media, pagination, font encoding, and other important extensions.

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by e-mail and on diskette.

In addition to text and various coded files processable directly by \TeX , the input to this issue includes several encapsulated PostScript files. More than 60 files were required to generate the final copy; over 60 more contain earlier versions of articles, auxiliary information, and records of correspondence with authors and referees. These numbers represent input files only; $.dvi$ files, device-specific translations, and fonts ($.tfm$ files and rasters) are excluded from the total.

Most articles as received were fully tagged for *TUGboat*, using either the plain-based or \LaTeX conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385). The macros are available from CTAN (the Comprehensive \TeX Archive Network); see *TUGboat* 14, no. 2, p. 100. The TUG office will provide copies of the macros on diskette to authors who have no electronic access.

By number, 85% of the articles in this issue are in \LaTeX , but only about 57% of the pages. The three articles by David Salomon were all tagged for the plain-based `tugboat.sty`; one of them redefined the entire verbatim system, requiring that it be processed separately from the others (which also incorporated verbatim segments, but without affecting the *TUGboat* macros).

Test runs of articles were made separately and in groups to determine the arrangement and page numbers (to satisfy any possible cross references). A file containing all starting page numbers, needed in any case for the table of contents, was compiled before the final run. Final processing was done in 2 runs of \TeX and 2 of \LaTeX , using the page number file for reference.

In addition to the three articles by Salomon, The following material was prepared using the plain-based `tugboat.sty`:

- the TUG calendar, page 66.
- these Production notes.
- "Coming next issue".

Output

The bulk of this issue was prepared at the American Mathematical Society from files installed on a VAX 6320 (VMS) and \TeX 'ed on a server running under Unix on a Solbourne workstation. Output was typeset on the Math Society's Compugraphic 9600 Imagesetter, a PostScript-based machine, using the Blue Sky/Y&Y PostScript implementation of the CM fonts, with additional fonts downloaded for special purposes.

Photographs illustrating the article by Claudio Beccari (p. 9) were converted to halftones by traditional means. Two diagrams for the Salomon/Hendryx article on "Slanted lines" (p. 59) were provided as camera-ready copy and pasted in.

Coming Next Issue

Tools for interaction

Michael Downes describes two documentstyle options, `dialog.sty` and `menus.sty`, which provide functions for printing menus on a screen and reading users' responses. These have been written so that they are also usable with non-L^AT_EX macro packages that include `plain.tex` in their base, such as $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX or eplain.

More new books

Reviews of the following are expected:

- Norman Walsh, *Making T_EX Work*
- Christian Rolland, *L^AT_EX guide pratique*
- Stanley Sawyer and Steven Krantz,
A T_EX Primer for Scientists
- and possibly others ...

Institutional Members

The Aerospace Corporation,
El Segundo, California

Air Force Institute of Technology,
Wright-Patterson AFB, Ohio

American Mathematical Society,
Providence, Rhode Island

ArborText, Inc.,
Ann Arbor, Michigan

Brookhaven National Laboratory,
Upton, New York

Brown University,
Providence, Rhode Island

California Institute of Technology,
Pasadena, California

Carleton University,
Ottawa, Ontario, Canada

Centre Inter-Régional de
Calcul Électronique, CNRS,
Orsay, France

CERN, *Geneva, Switzerland*

College Militaire Royal de Saint
Jean, *St. Jean, Quebec, Canada*

College of William & Mary,
Department of Computer Science,
Williamsburg, Virginia

Communications
Security Establishment,
Department of National Defence,
Ottawa, Ontario, Canada

Cornell University,
Mathematics Department,
Ithaca, New York

CSTUG, *Praha, Czech Republic*

Elsevier Science Publishers B.V.,
Amsterdam, The Netherlands

Escuela Superior de
Ingenieros Industriales,
Sevilla, Spain

European Southern Observatory,
Garching bei München, Germany

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

Florida State University,
Supercomputer Computations
Research, *Tallahassee, Florida*

GKSS, Forschungszentrum
Geesthacht GmbH,
Geesthacht, Germany

Grinnell College,
Computer Services,
Grinnell, Iowa

Hong Kong University of
Science and Technology,
Hong Kong

Institute for Advanced Study,
Princeton, New Jersey

Institute for Defense Analyses,
Communications Research
Division, *Princeton, New Jersey*

Iowa State University,
Ames, Iowa

Los Alamos National Laboratory,
University of California,
Los Alamos, New Mexico

MacroSoft, *Warsaw, Poland*

Marquette University,
Department of Mathematics,
Statistics and Computer Science,
Milwaukee, Wisconsin

New techniques in METAFONT

Certain geometrical problems that arise very often in glyph design are not directly solvable by METAFONT's `plain` macros. Yannis Haralambous presents two such problems and solutions for them, along with a discussion of an approach that, although geometrically correct, does *not* work in real-world METAFONT practice and should be avoided. [Delayed by technical difficulties]

ASCII.sty

Because they needed a font to prepare a table of ASCII control codes and their associated IBM graphics characters for a book on interfacing medical equipment to an IBM PC, R. Ramasubramanian, R.W.D. Nickalls and M.A. Reed developed a new style option and encoded font containing these characters for use with T_EX and L^AT_EX. The new font is based on the public domain IBM Courier font.

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

Max Planck Institut
für Mathematik,
Bonn, Germany

Naval Postgraduate School,
Monterey, California

New York University,
Academic Computing Facility,
New York, New York

Nippon Telegraph &
Telephone Corporation,
Software Laboratories,
Tokyo, Japan

Personal T_EX, Incorporated,
Mill Valley, California

Princeton University,
Princeton, New Jersey

Rogaland University,
Stavanger, Norway

Rutgers University,
Computing Services,
Piscataway, New Jersey

Space Telescope Science Institute,
Baltimore, Maryland

Springer-Verlag,
Heidelberg, Germany

Springer-Verlag New York, Inc.,
New York, New York

Stanford Linear Accelerator
Center (SLAC),
Stanford, California

Stanford University,
Computer Science Department,
Stanford, California

Texas A & M University,
Department of Computer Science,
College Station, Texas

United States Naval
Postgraduate School,
Monterey, California

Universität Augsburg,
Augsburg, Germany

University of California, Berkeley,
Space Astrophysics Group,
Berkeley, California

University of California, Irvine,
Information & Computer Science,
Irvine, California

University of Canterbury,
Christchurch, New Zealand

University College,
Cork, Ireland

University of Delaware,
Newark, Delaware

University of Groningen,
Groningen, The Netherlands

University of Heidelberg,
Computing Center,
Heidelberg, Germany

University of Illinois at Chicago,
Computer Center,
Chicago, Illinois

Universität Koblenz-Landau,
Koblenz, Germany

University of Manitoba,
Winnipeg, Manitoba

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway

University of Salford,
Salford, England

University of South Carolina,
Department of Mathematics,
Columbia, South Carolina

University of Southern California,
Information Sciences Institute,
Marina del Rey, California

University of Stockholm,
Department of Mathematics,
Stockholm, Sweden

University of Texas at Austin,
Austin, Texas

Università degli Studi di Trento,
Trento, Italy

Uppsala University,
Uppsala, Sweden

Villanova University,
Villanova, Pennsylvania

Vrije Universiteit,
Amsterdam, The Netherlands

Washington State University,
Pullman, Washington

Wolters Kluwer,
Dordrecht, The Netherlands

Yale University,
Department of Computer Science,
New Haven, Connecticut



Individual Membership Application

Complete and return this form with payment to:

TeX Users Group
Membership Department
P.O. Box 869
Santa Barbara, CA 93102 USA
Telephone: (805) 963-1338
FAX: (805) 963-8358
Email: tug@tug.org

Membership is effective from January 1 to December 31 and includes subscriptions to TUGboat, The Communications of the TeX Users Group and the TUG newsletter, TeX and TUG NEWS. Members who join after January 1 will receive all issues published that calendar year.

For more information ...

Whether or not you join TUG now, feel free to return this form to request more information. Be sure to include your name and address in the spaces provided to the right.

Check all items you wish to receive below:

- Institutional membership information
- Course and meeting information
- Advertising rates
- Products/publications catalogue
- Public domain software catalogue

Name _____

Institutional affiliation, if any _____

Position _____

Address (business or home (circle one)) _____

City _____ Province/State _____

Country _____ Postal Code _____

Telephone _____ FAX _____

Email address _____

I am also a member of the following other TeX organizations:

Specific applications or reasons for interest in TeX:

There are two types of TUG members: regular members, who pay annual dues of \$60; and full-time student members, whose annual dues are \$30. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

- Regular at \$60 Full-time student at \$30

Amount enclosed for 1994 membership: \$ _____

Check/money order payable to TeX Users Group enclosed
(checks in US dollars must be drawn on a US bank; checks in other currencies are acceptable, drawn on an appropriate bank)

Bank transfer:

TeX Users Group, Account #1558-816,
Santa Barbara Bank and Trust, 20 East Carrillo Street,
Santa Barbara, CA 93101 USA

your bank _____

ref # _____

Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____



Institutional Membership Application

Institution or Organization _____

 Principal contact _____
 Address _____

 City _____ Province/State _____
 Country _____ Postal Code _____
 Daytime telephone _____ FAX _____
 Email address _____

Complete and return this form with payment to:

TeX Users Group
 Membership Department
 P.O. Box 869
 Santa Barbara, CA 93102
 USA

Membership is effective from January 1 to December 31. Members who join after January 1 will receive all issues of *TUGboat* and *TeX* and *TUG NEWS* published that calendar year.

For more information ...

Correspondence

TeX Users Group
 P.O. Box 869
 Santa Barbara, CA 93102
 USA
 Telephone: (805) 963-1338
 FAX: (805) 963-8358
 Email: tug@tug.org

Whether or not you join TUG now, feel free to return this form to request more information.

Check all items you wish to receive below:

- Course and meeting information
- Advertising rates
- Products/publications catalogue
- Public domain software catalogue

Each Institutional Membership entitles the institution to:

- designate a number of individuals to have full status as TUG individual members;
- take advantage of reduced rates for TUG meetings and courses for *all* staff members;
- be acknowledged in every issue of *TUGboat* published during the membership year.

Educational institutions receive a \$100 discount in the membership fee. The three basic categories of Institutional Membership each include a certain number of individual memberships. Additional individual memberships may be obtained at the rates indicated. Fees are as follows:

Category	Rate (educ./non-educ.)	Add'l mem.
A (includes 7 memberships)	\$ 540 / \$ 640	\$50 ea.
B (includes 12 memberships)	\$ 815 / \$ 915	\$50 ea.
C (includes 30 memberships)	\$1710 / \$1810	\$40 ea.

Please indicate the type of membership for which you are applying:

Category _____ + _____ additional individual memberships

Amount enclosed for 1994 membership: \$ _____

Check/money order payable to TeX Users Group enclosed
(payment in US dollars must be drawn on a US bank; payment in other currencies is acceptable, drawn on an appropriate bank)

Bank transfer: your bank _____
 ref # _____

TeX Users Group, Account #1558-816,
 Santa Barbara Bank and Trust, 20 East Carrillo Street,
 Santa Barbara, CA 93101 USA

Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

Please attach a list of individuals whom you wish to designate as TUG individual members. Minimally, we require names and addresses so that TUG publications may be sent directly to these individuals, but we would also appreciate receiving the supplemental information regarding phone numbers, email addresses, and TeX interests as requested on the TUG Individual Membership Application form. For this purpose, the latter application form may be photocopied and mailed with this form.

North America

Abrahams, Paul

214 River Road, Deerfield, MA
01342; (413) 774-5500

Composition and typesetting of high-quality books and technical documents. Complete production services using any PostScript fonts. Assistance with book design and copy. I am a computer consultant with a computer science education.

American Mathematical Society

P. O. Box 6248, Providence, RI
02940; (401) 455-4060

Typesetting from DVI files on an Autologic APS Micro-5 or an Agfa Compugraphic 9600 (PostScript). Times Roman and Computer Modern fonts. Composition services for mathematical and technical books and journal production.

Anagnostopoulos, Paul C.

433 Rutland Street, Carlisle, MA
01741; (508) 371-2316

Composition and typesetting of high-quality books and technical documents. Production using Computer Modern or any available PostScript fonts. Assistance with book design. I am a computer consultant with a Computer Science education.

ArborText, Inc.

1000 Victors Way, Suite 400,
Ann Arbor, MI 48108;
(313) 996-3566

TEX installation and applications support. TEX-related software products.

Archetype Publishing, Inc.,

Lori McWilliam Pickert
P. O. Box 6567, Champaign, IL
61821; (217) 359-8178

Experienced in producing and editing technical journals with TEX; complete book production from manuscript to camera-ready copy; TEX macro writing including complete macro packages; consulting.

The Bartlett Press, Inc.,

Frederick H. Bartlett
Harrison Towers, 6F, 575 Easton
Avenue, Somerset, NJ 08873;
(201) 745-9412

Vast experience: 100+ macro packages, over 30,000 pages published with our macros; over a decade's experience in all facets of publishing, both TEX and non-TEX; all services from copyediting and design to final mechanicals.

Cowan, Dr. Ray F.

141 Del Medio Ave. #134,
Mountain View, CA 94040;
(415) 949-4911

Ten Years of TEX and Related Software Consulting: Books, Documentation, Journals, and Newsletters

TEX & LATEX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

Hoenig, Alan

17 Bay Avenue, Huntington, NY
11743; (516) 385-0736

TEX typesetting services including complete book production; macro writing; individual and group TEX instruction.

Magus, Kevin W. Thompson

P. O. Box 390965, Mountain View
CA 94039-0965;
(800) 848-8037; (415) 940-1109;
magus@cup.portal.com

LATEX consulting from start to finish. Layout design and implementation, macro writing, training, phone support, and publishing. Can take LATEX files and return camera ready copy. Knowledgeable about long document preparation and mathematical formatting.

NAR Associates

817 Holly Drive E. Rt. 10,
Annapolis, MD 21401;
(410) 757-5724

Extensive long term experience in TEX book publishing with major publishers, working with authors or publishers to turn electronic copy into attractive books. We offer complete free lance production services, including design, copy editing, art sizing and layout, typesetting and repro production. We specialize in engineering, science, computers, computer graphics, aviation and medicine.

Ogawa, Arthur

920 Addison, Palo Alto, CA 94301;
(415) 323-9624

Experienced in book production, macro packages, programming, and consultation. Complete book production from computer-readable copy to camera-ready copy.

Pronk&Associates Inc.

1129 Leslie Street, Don Mills,
Ontario, Canada M3C 2K5;
(416) 441-3760; Fax: (416) 441-9991

Complete design and production service. One, two and four-color books. Combine text, art and photography, then output directly to imposed film. Servicing the publishing community for ten years.

Quixote Digital Typography,

Don Hosek

555 Guilford, Claremont,
CA 91711; (909) 621-1291;
Fax: (909) 625-1342

Complete line of TEX, LATEX, and METAFONT services including custom LATEX style files, complete book production from manuscript to camera-ready copy; custom font and logo design; installation of customized TEX environments; phone consulting service; database applications and more. Call for a free estimate.

Richert, Norman

1614 Loch Lake Drive, El Lago, TX
77586; (713) 326-2583

TEX macro consulting.

TEXnology, Inc.,

Amy Hendrickson

57 Longwood Ave., Brookline, MA
02146; (617) 738-8029

TEX macro writing (author of MacroTEX); custom macros to meet publisher's or designer's specifications; instruction.

Type 2000

16 Madrona Avenue, Mill Valley,
CA 94941; (415) 388-8873;
Fax: (415) 388-8865

\$2.50 per page for 2000 DPI TEX camera ready output! We have a three year history of providing high quality and fast turnaround to dozens of publishers, journals, authors and consultants who use TEX. Computer Modern, Bitstream and METAFONT fonts available. We accept DVI files only and output on RC paper. \$2.25 per page for 100+ pages, \$2.00 per page for 500+ pages.

Outside North America

TypoTEX Ltd.

Electronical Publishing, Battyány
u. 14. Budapest, Hungary H-1015;
(036) 11152 337

Editing and typesetting technical journals and books with TEX from manuscript to camera ready copy. Macro writing, font designing, TEX consulting and teaching.

Information about these services can be obtained from:

TEX Users Group

P. O. Box 869
Santa Barbara, CA 93102-0869
Phone: (805) 963-1388
Fax: (805) 963-8538
Email: tug@tug.org

L^AT_EX₂_ε . . . Knuth . . . *Mathematica*[®]

Leading the way in scientific computing. Addison-Wesley.

**When looking for the best in scientific computing, you've come to rely on Addison-Wesley.
Take a moment to see how we've made our list even stronger.**

The L^AT_EX Companion

Michael Goossens, Frank Mittelbach, and Alexander Samarin
This book is packed with information needed to use L^AT_EX even more productively. It is a true companion to Leslie Lamport's users guide as well as a valuable complement to any L^AT_EX introduction. Describes the new L^AT_EX standard.
1994 (0-201-54199-8) 400 pp. Softcover

L^AT_EX: A Document Preparation System, Second Edition

Leslie Lamport
The authoritative user's guide and reference manual has been revised to document features now available in the new standard software release—L^AT_EX₂_ε. The new edition features additional styles and functions, improved font handling, and much more.
1994 (0-201-52983-1) 256 pp. Softcover

The Stanford GraphBase: A Platform for Combinatorial Computing

Donald E. Knuth
This book represents the first fruits of Knuth's preparation for Volume 4 of *The Art of Computer Programming*. It uses examples to demonstrate the art of literate programming, and provides a useful means for comparing combinatorial algorithms.
1994 (0-201-54275-7) 600 pp. Hardcover

The CWEB System of Structured Documentation (Version 3.0)

Donald E. Knuth and Silvio Levy
CWEB is a version of WEB for documenting C and C++ programs. CWEB combines T_EX with two of today's most widely used profes-

sional programming languages. This book is the definitive user's guide and reference manual for the CWEB system.
1994 (0-201-57569-8) approx. 240 pp. Softcover

Concrete Mathematics, Second Edition

Ronald L. Graham, Donald E. Knuth, and Oren Patashnick
With improvements to almost every page, the second edition of this classic text and reference introduces the mathematics that supports advanced computer programming.
1994 (0-201-55802-5) 672 pp. Hardcover

Applied Mathematica[®]: Getting Started, Getting It Done

William T. Shaw and Jason Tigg
This book shows how *Mathematica*[®] can be used to solve problems in the applied sciences. Provides a wealth of practical tips and techniques.
1994 (0-201-54217-X) 320 pp. Softcover

The Joy of Mathematica[®]

Alan Shuchat and Fred Shultz
This software product provides easy-to-use menus for Macintosh versions of *Mathematica*[®]. Its accompanying book is an exploration of key issues and applications in Mathematics.
1994 (0-201-59145-6) 200 pp. Softcover + disk

Look for these titles wherever fine technical books are sold.



Addison-Wesley Publishing Company
1-800-447-2226

T_EX Publishing Services



From the Basic:

The American Mathematical Society offers you two basic, low cost T_EX publishing services.

- You provide a DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. \$5 per page for the first 100 pages; \$2.50 per page for additional pages.
- You provide a PostScript output file and we will provide typeset pages using an Agfa/Compugraphic 9600 imagesetter. \$7 per page for the first 100 pages; \$3.50 per page for additional pages.

There is a \$30 minimum charge for either service. Quick turnaround is also provided... a manuscript up to 500 pages can be back in your hands in one week or less.

To the Complex:

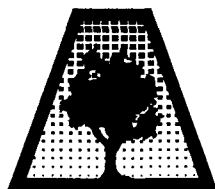
As a full-service T_EX publisher, you can look to the American Mathematical Society as a single source for any or all your publishing needs.

Macro-Writing	T _E X Problem Solving	Non-CM Fonts	Keyboarding
Art and Pasteup	Camera Work	Printing and Binding	Distribution

For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P. O. Box 6248, Providence, RI 02940, or call 401-455-4060.

Index of Advertisers

80	Addison-Wesley
81	American Mathematical Society
82	ArborText
Cover 3	Blue Sky Research
82	Micro Programs, Inc.
83	Springer-Verlag
84	Y&Y

NEW!**NEW!**

ARBORTEXT INC.

- Silicon Graphics Iris or Indigo
- Solaris 2.1
- DVILASER/HP3
- Motif and OPEN LOOK Preview

Complete T_EX packages
Ready to use, fully documented and supported.

Also Available For: Sun-4 (SPARC), IBM RS/6000,
 DEC/RISC-Ultrix, HP 9000, and IBM PC's

Call us for more information on our exciting new products!

1000 Victors Way ■ Suite 400 ■ Ann Arbor, MI 48108 ■ (313) 996-3566 ■ FAX (313) 996-3573



YOUR ONE STOP SOURCE FOR T_EX MATERIALS

Our version of T_EX supports the New Font Selection Scheme.

We have a complete selection of books on T_EX and document publishing.

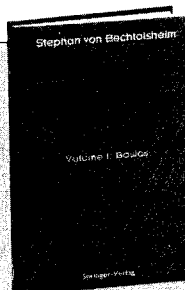
A MicroT_EX full system is \$450.00 - Ask for details to upgrade your system.

T_EX TOOL BOX

T _E XHelp \$49.95	DEMACS Editor (when ordered with MicroT _E X) \$10.00	Voyager \$25.00
SL _T T _E X & A _M S-T _E X (Call for details)	T _E Xpic \$79.00	Capture Graphics (a close-out special) \$75.00
	AmSpell Checker (when ordered with MicroT _E X) \$10.00	

Micro Programs Inc., 251 Jackson Ave., Syosset NY 11791 (516) 921-1351

GET MORE OUT OF T_EXNOLOGY WITH SPRINGER BOOKS



S. VON BECHTOLSHEIM, Integrated Computer Software, Inc., Palatine, IL

T_EX IN PRACTICE

T_EX is a computerized programmable typesetting system. It facilitates the typesetting of text, mathematical formulas and tables. Being far more powerful and flexible than any other available systems, T_EX also exhibits a fair amount of complexity. T_EX in Practice will help you to unravel the power and conquer the complexity.

This four-volume set is the ultimate reference and guide for all T_EX users. It offers a detailed analysis of all features of T_EX, many ready-to-use macros for solving a wide range of typesetting problems and numerous examples to facilitate the understanding of T_EX. Written by an acknowledged expert in the field, T_EX in Practice addresses the needs of T_EX users with different levels of experience. Readers will find it to be a valuable permanent source of information on one of the world's most powerful typesetting systems.

1993/1860 PP., 63 ILLUS./HARDCOVER
MONOGRAPHS IN VISUAL COMMUNICATION

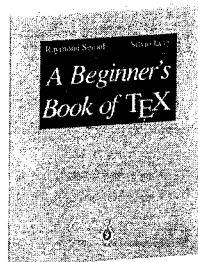
FOUR-VOLUME SET
ISBN 0-387-97296-X /\$169.00

VOLUME I: BASICS
ISBN 0-378-97595-0/\$49.00

VOLUME II: PARAGRAPHS, MATH, AND FONTS
ISBN 0-378-97596-9/\$49.00

VOLUME III: TOKENS, MACROS
ISBN 0-378-97597-7/\$49.00

VOLUME IV: OUTPUT ROUTINES, TABLES
ISBN 0-378-97598-5/\$49.00

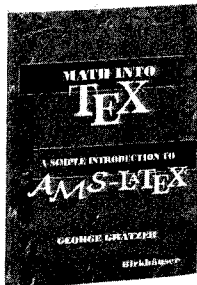


R. SEROUL, Université Louis Pasteur, Strasbourg, France and **S. LEVY**, University of Minnesota, Minneapolis, MN

A BEGINNER'S BOOK OF T_EX

A friendly introduction to T_EX, this book addresses the novice user, but contains much information that will be useful to aspiring T_EX wizards. Also presented are numerous examples and many "tricks" based on the authors' long experience with T_EX.

1991/283 PP./SOFTCOVER \$29.95
ISBN 0-387-97562-4



G. GRÄTZER, University of Manitoba, Winnipeg, Canada

MATH INTO T_EX

A Simple Introduction to AMS-L^AT_EX

This book provides beginners with a direct approach to typesetting mathematics with AMS-L^AT_EX. It is ideal for mathematicians, engineers, or technical typists who want to start immediately with writing and typesetting articles using AMS-L^AT_EX. Includes a disk with ready-to-use templates to get you up and started fast.

1993/296 PP./SOFTCOVER \$42.50
ISBN 0-8176-3637-4
PUBLISHED BY BIRKHÄUSER



M. DOOB, University of Manitoba, Winnipeg, Canada

T_EX: STARTING FROM SQUARE 1

A book for the complete newcomer to T_EX. It begins with simple exercises on typesetting text and slowly advances into more complex mathematical constructions and tables. Included are various tables for quick references.

1993/114 PP./SOFTCOVER \$25.00
ISBN 0-387-56441-1

Three Easy Ways to Order:

CALL Toll-Free 1-800-SPRINGER (NJ call 201-348-4033) or FAX 201-348-4505. Please mention S952 when ordering by phone.

WRITE to Springer-Verlag New York, Inc., Attn: J. Jeng, Dept. S952, 175 Fifth Avenue, New York, NY 10010.

VISIT your local scientific bookstore.

Payment can be made by check, purchase order, or credit card. Please enclose \$2.50 for shipping (\$1.00 each additional book) & add appropriate sales tax if you reside in CA, IL, MA, NJ, NY, PA, TX, VA, and VT. Canadian residents please add 7% GST. Foreign airmail charge is \$10.00 for the first book (\$5.00 each additional book).

Remember...your 30-day return privilege is always guaranteed!

2/94

REFERENCE #: S952



Springer-Verlag
NEW YORK

DVIWindo [newtugad.dvi] page: 1
▲ ▼

File
Preferences
Previous!
Next!
Unmagnify!
Magnify!
Fonts

Bitmap-free T_EX for Windows

Powerful, fast, flexible T_EX system for Windows

TeX Package

Y&Y T_EX Package includes:

- DVIWindo
- DVIPSONE
- Y&Y big T_EX
- Adobe Type Manager
- Acrobat Reader
- PostScript Type 1 fonts

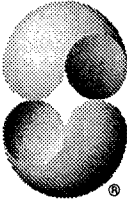
Unique Features

- Uses Type 1 *or* TrueType fonts
- Provides *partial font downloading*
- Can use *any* Windows printer driver
- Big T_EX runs in Windows *or* DOS
- Commercial grade, fully hinted fonts
- *Complete* flexibility in font encoding
- Support for EPS *and* TIFF images

Why Y&Y?

Mature products. Years of experience with Windows, PostScript printers and scalable outline fonts. We *understand* and know how to *avoid* problems with Windows, ATM, 'clone' printers, and problem fonts.

Y&Y — the experts in scalable outline fonts for T_EX



Y&Y, Inc. 45 Walden St., Suite 2F, Concord, MA 01742 USA — (800) 742-4059 — (508) 371-3286 — (508) 371-2004 (fax)

DVIWindo and DVIPSONE are trademarks of Y&Y, Inc. Windows is a registered trademark of MicroSoft Co. Adobe Type Manager is a registered trademark of Adobe Systems Inc.

Interactive T_EX ✓

WYSIWYG T_EX ✓

User-friendly T_EX ✓

Textures ✓ It's not like any other T_EX system.^[1]

When Apple introduced the Macintosh and its graphic interface, we embarked on a long-term project of research toward a T_EX system “for the rest of us,” a T_EX system that would be humanely interactive, and visibly responsive; an integrated T_EX system, designed to be radically easy for those new to T_EX, and engineered to be the best for expert T_EX users. The research continues; the product is Textures.

Textures is something of a Copernican revolution in T_EX interface. The paradigm shifts from the usual T_EX “input-process-output-repeat” mode, to a wider frame wherein the T_EX language describes a dynamic document that is continuously, quietly “realized” as you write it, with no process steps whatsoever.

This change in perspective must be experienced to be fully grasped. As you would expect, Textures is good for beginners. You might be surprised to know that it's also good for experienced T_EX users and especially good for T_EX programmers.^[2] It's not a “front-end” or an imitation, it's a full-scale live T_EX processor that's actually easy to use.

There's much more to Textures than a new perspective on T_EX, of course; too much to list here but we'll mention custom menus, Computer Modern PostScript fonts, illustrations, inter-application communication, automatic installation, genuine support,

We don't claim perfection; nor do we believe in exaggerated marketing, odd as this may seem; and we do understand our finite abilities to provide all that one might wish. But we also guarantee that you will be satisfied with Textures and with the service you receive from Blue Sky Research, or we will refund the (very finite) price you pay.

[1]
On the other hand, Textures is *exactly* like every other T_EX system. Its T_EX engine is strictly standard and up-to-date; it runs L^AT_EX, A^MS-T_EX, and all standard T_EX macros without change.

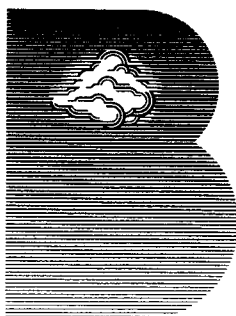
But even here it's not ordinary, with hand-tuned assembler code for maximum performance, and a transparent memory model that you can't fill until you run out of disk.

[2]
If you are a serious T_EX user on another platform, it can be worth getting a Mac just to run Textures.

For all Macintosh
processors and printers
minimum 2.5MB memory
and 5MB disk

Blue Sky Research
534 SW Third Avenue
Portland, Oregon 97204
USA

800 622 8398
503 222 9571
facsimile 503 222 1643
sales@bluesky.com



TUGBOAT

Volume 15, Number 1 / March 1994

	3	Addresses
General Delivery	5	Opening words / <i>Christina Thiele</i> Board news; The annual meeting; T _E X in new places; Announcing EuroT _E X'94; Office updates
	6	Editorial comments / <i>Barbara Beeton</i> TUGboat wish list; Another award for DEK; New magazine on type and typography; Hyphenation and exceptions
	7	The T _E X hierarchy / <i>Donald Arseneau, Raymond Chen and Victor Eijkhout</i>
Philology	9	Typesetting of ancient languages / <i>Claudio Beccari</i>
	17	Comments on the paper "Typesetting Catalan texts with T _E X" (TUGboat 14, no. 3, pp. 252-259) / <i>Claudio Beccari</i>
	17	Comments on the comments: Typesetting Catalan texts with T _E X / <i>Gabriel Valiente Feruglio and Robert Fuster</i>
Book Reviews	18	George Grätzer, <i>Math into T_EX</i> / <i>Nico Poppelier</i>
	19	Stephan von Bechtolsheim, <i>T_EX in Practice</i> / <i>T. L. (Frank) Pappas</i>
	21	Gianni Gilardi, <i>Il T_EX - Introduzione al linguaggio e complementi avanzati</i> / <i>Claudio Beccari</i>
	22	Erik Spiekermann & E. M. Ginger, <i>Stop Stealing Sheep</i> / <i>Merry Obrecht Sawdey</i>
	24	Eric van Herwijnen, <i>Practical SGML</i> / <i>Nico Poppelier</i>
	25	Donald E. Knuth, <i>Literate Programming</i> / <i>Christine Detig and Joachim Schrod</i>
Tutorials	28	Output routines: Examples and techniques, Part IV: Horizontal techniques / <i>David Salomon</i>
	40	Verbatim copying and listing / <i>David Salomon</i>
Macros	55	The bag of tricks / <i>Victor Eijkhout</i>
	57	Random bit generator in T _E X / <i>Hans van der Meer</i>
	59	Slanted lines with controlled thickness / <i>David Salomon and Matthew N. Hendryx</i>
Letters	62	On indexing and errors / <i>Jonathan Fine</i>
L^AT_EX	63	L ^A T _E X2 _ε update, dateline: 31 January 1994 / <i>Chris Rowley</i>
	64	Addenda: A suggested "operational requirement" for L ^A T _E X3's treatment of bibliographic references (TUGboat 14, no. 4, pp. 425-433) / <i>David Rhead</i>
News & Announcements	66	Calendar
	68	TUG '94— Announcement and preliminary program (Santa Barbara, California, 31 July-4 August 1994)
	69	Call for papers, EuroT _E X '94 (Gdansk, Poland, 26-30 September 1994)
Late-Breaking News	74	Production notes / <i>Barbara Beeton</i>
	75	Coming next issue
TUG Business	70	Meet the Board, Part I Barbara Beeton; Mimi Burbank; Jackie Damrau; Luzia Dietsche; Michael Doob; Michel Goossens; Tom Rokicki
	75	Institutional members
Forms	77	TUG membership application
Advertisements	79	T _E X consulting and production services
	81	Index of advertisers