Computer technology has far to go to duplicate the spirit and excitement of art. When such a day is reached, however, it will require an artist to do the programming.

Gerald Oster
"Optical Art" (1965)

TUGBOAT

THE TEX USERS GROUP NEWSLETTER
EDITOR BARBARA BEETON

VOLUME 8, NUMBER 1 • APRIL, 1987
PROVIDENCE • RHODE ISLAND • U.S.A.

TUGboat

The communications of the TEX Users Group are published irregularly at Providence, Rhode Island, and are distributed as a benefit of membership both to individual and institutional members. Three issues of TUGboat are planned for 1987.

Submissions to TUGboat are for the most part reproduced with minimal editing, and any questions regarding content or accuracy should be directed to the authors, with an information copy to the Editor.

Submitting Items for Publication

The deadline for submitting items for Vol. 8, No. 2, is May 18, 1987; the issue will be mailed in July.

Manuscripts should be submitted to a member of the TUGboat Editorial Committee. 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.

Contributions in camera copy form are encouraged, as is electronic submission of items on magnetic tape or diskette, via electronic mail, or transferred directly to the AMS computer; for instructions, write or call Barbara Beeton.

TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call Ray Goucher.

Other TUG Publications

TUG is interested in considering for publication manuals or other documentation that might be useful to the TEX community in general. If you have any such items or know of any that you would like considered for publication, contact Ray Goucher at the TUG office.

TUGboat Editorial Committee

See page 3 for addresses.

Addresses

Jacques André
IRISA/INRIA–Rennes
Campus de Beaulieu
F-35042 Rennes Cedex, France
uucp: ...mcvax!inria!irisa!jandre

Lawrence A. Beck
Grumman Data Systems
R & D, MS D12-237
Woodbury, NY 11797
516-682-8478

Nelson H. F. Beebe
Center for Scientific Computing
220 South Physics
University of Utah
Salt Lake City, UT 84112
801-581-5254
Arpanet: Beebe@Utah-Science, Beebe@Utah-CS, BeebeN@Utah-RUAC

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
Arpanet: bb@Sail.Stanford.Edu, bnb@xx.lcs.MIT.Edu

Edwin V. Bell, II
Department of Physics & Astronomy
University of Kansas
Lawrence, KS 66045
913-864-3610
Bitnet: Bell@UKANVAX

Neenie Billawala
2014 Colony #8
Mountain View, CA 94943
415-965-0643

Malcolm Brown
ACIS/IRIS
Stanford University
Cypress Hall, Rm E7
Stanford, CA 94305
415-723-1055
Arpanet: MBB@Portia.Stanford.Edu

Lance Carnes
c/o Personal TEX
12 Madrona Avenue
Mill Valley, CA 94941
415-388-8853

James R. Celoni, S.J.
Jesuit Community
Santa Clara University
Santa Clara, CA 95053
Arpanet: Celoni@Score.Stanford.edu

S. Bart Childs
Dept of Computer Science
Texas A & M University
College Station, TX 77843-3112
409-845-5470

Maria Code
Data Processing Services
1371 Sydney Dr
Sunnyvale, CA 94087
408-735-8006

John M. Crawford
Computing Services Center
College of Business
Ohio State University
Columbus, OH 43210
614-292-1741
Arpanet: Crawford-J@Ohio-State
Bitnet: TSO135@OHSTVMA

Lawrence D. Cutter
121 Mohawk Drive
Schenectady, NY 12303
518-387-6247

Jackie Damrau
Dept of Math & Statistics
Univ of New Mexico
Albuquerque, NM 87131
505-277-4623
Bitnet: damrau@unmb
UUCP: damrau@unmvax

Jennifer L. Dyck
School of Natural Sciences
Department of Psychology
California State University, Fresno
Fresno, CA 93740-0011
209-294-2691

Maureen Eppstein
Administrative Publication Manager
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-723-9254
Arpanet: MVEppstein@Score.Stanford.Edu

Jim Fox
Academic Computing Center HG-45
University of Washington
3737 Brooklyn Ave NE
Seattle, WA 98105
206-543-4320
Bitnet: fox7632@uwacdc

David Fuchs
1775 Newell
Palo Alto, CA 94303
415-323-9436

Richard Furuta
Department of Computer Science
Univ of Maryland
College Park, MD 20742
301-454-1461
Arpanet: furuta@mimsy.umd.edu

Edgar Fuß
Tools GmbH
Kaiserstraße 48
5300 Bonn
Federal Republic Germany
uucp: ....unido!bnu!fuss

Raymond E. Goucher
TEX Users Group
P. O. Box 9506
Providence, RI 02940
401-272-9500 x232

Dean Guenther
Computer Service Center
Washington State University
Computer Science Building,
Room 2144
Pullman, WA 99164-1220
509-335-0411
BITnet: Guenther@WSUVM1

Judy Hawkins
Center for Academic Computing
McConnell Hall
Smith College
Northampton, MA 01063
Bitnet: JHawkins@Smith

Doug Henderson
Division of Library Automation
Univ of California, Berkeley
186 University Hall
Berkeley, CA 94720
Bitnet: dlatex@ucbcmsa

Amy Hendrickson
57 Longwood Ave #8
Brookline, MA 02146
617-738-8029

Alan Hoenig
17 Bay Avenue
Huntington, NY 11743
516-385-0736

Helen S. Horstman
Lowell Observatory
Mars Hill Road, 1400 West
Flagstaff, AZ 86001
602-774-3358

Patrick D. Ion
Mathematical Reviews
416 Fourth Street
P. O. Box 8604
Ann Arbor, MI 48107
313-996-5273

Helmut Jürgensen
Dept of Computer Science
Univ of Western Ontario
London N6A 5B7, Ontario, Canada
519-661-3560
Bitnet: A5O5@UWOCC1
UUCP: helmut@deepthot

Arthur Keller
University of Texas at Austin
Department of Computer Science
Austin, TX 78712-1188
512-471-7316
Arpanet: ARK@SALLY.UTexas.Edu

Donald E. Knuth
Department of Computer Science
Stanford University
Stanford, CA 94305
Arpanet: DEK@Sail.Stanford.Edu

Linda Krick
Anglo-Australian Observatory
P. O. Box 296
Epping, N.S.W. 2121 Australia
+61 2 868-1666
uucp: {seismo,hplabs,mcvax,ukc,nttlab}
!munnari!aaoepp.oz!lk

Leslie Lamport
Systems Research Center
Digital Equipment Corp
130 Lytton Ave
Palo Alto, CA 94301
415-853-2170
Arpanet: lamport@SRC.DEC.COM

Silvio Levy
Math Department, Fine Hall
Princeton University
Washington Road
Princeton, NJ 08544
uucp: seismo!princeton!fine!levy

Tor Lillqvist
VTT/ATK
Lehtisaarentie 2
SF-00340 Helsinki, Finland
Bitnet: tml@fingate

Pierre A. MacKay
Northwest Computer Support Group
University of Washington
Mail Stop DW-10
Seattle, WA 98195
206-543-6259; 545-2386
Arpanet: MacKay@Ward.CS.Washington.edu

Rick Mallett
Computing Services
Room 1208 Arts Tower
Carleton University
Ottawa (K1S 5B6), Ontario, Canada
613-231-7145

Laurie Mann
Stratus Computer
55 Fairbanks Boulevard
Marlboro, MA 01752
617-460-2610
uucp: harvard!anvil!es!Mann

Graeme McKinstry
Computing Services Centre
University of Otago
P. O. Box 56
Dunedin, New Zealand
(024) 771-640

William A. McWorter
Mathematics Department
Ohio State University
231 W. 18th
Columbus, Ohio 43210

Boyd Michailovsky
CNRS/LACITO
44 rue de l'Amiral Mouchez
75014 Paris, France

David Nash
Center for Cognitive Science
20B-225 MIT
Cambridge, MA 02139
617-253-7355
Arpanet: lex.nash@SPEECH.MIT.EDU,
nash@COGITO.MIT.EDU

Richard S. Palais
Department of Mathematics
Brandeis University
Waltham, MA 02154
617-647-2667

Mitch Pfeffer
Suite 90
148 Harbor View South
Lawrence, NY 11559
516-239-4110

Arnold Pizer
Department of Mathematics
University of Rochester
Rochester, NY 14627
716-275-4428

Craig Platt
Dept of Math & Astronomy
Machray Hall
Univ of Manitoba
Winnipeg R3T 2N2, Manitoba, Canada
204-474-9832
platt%cc.uofm.cdn@ubc.csnet

Raymond A. Ryan
Department of Mathematics
University College Galway
Galway, Ireland
(091) 24411

Tony Siegman
Stanford University
Ginzton Lab
Stanford, CA 94305
415-723-0222
Arpanet: Siegman@Sierra.Stanford.edu

Barry Smith
Kellerman & Smith
534 SW Third Ave
Portland, OR 97204
503-222-4234; TLX 9102404397

Ralph Stromquist
MACC
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706
608-262-8821

Rilla Thedford
Intergraph Corporation, MS HQ013
One Madison Industrial Park
Huntsville, AL 35807
205-772-6494

Anders Thulin
ContextVision AB
Teknikringen 1
S-583 30 Linköping, Sweden

Georgia K.M. Tobin
The Metafoundry
OCLC Inc., MC 485
6565 Frantz Road
Dublin, OH 43017
614-764-6087

Andrew Trevorrow
University of Adelaide
Computing Services
Box 498, G.P.O. Adelaide
South Australia 5001, Australia
ACSnet: akt@uacomsci.ua.oz
Arpanet: munnari!uacomsci.ua.oz!akt
@seismo.css.gov

Joey K. Tuttle
I P Sharp Associates
220 California Avenue, Suite 201
Palo Alto, CA 94306
415-327-1700

Samuel B. Whidden
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500

Kurt Bernardo Wolf
IIMAS-UNAM
Apdo. Postal 20-726
0100 México, D.F., México
(73) 17-3077

Reinhard Wonneberger
Drachenstieg 5
D 2000 Hamburg 63
Federal Republic Germany
040/592354
Bitnet: BO3WBG at DHHDESY3

Ken Yap
Dept of Computer Science
University of Rochester
Rochester, NY 14627
Arpanet: Ken@Rochester
Usenet: ..!{allegra,decvax,seismo,
cmcl2,harvard,topaz}!rochester!ken

Hermann Zapf
Seitersweg 35
D-6100 Darmstadt
Federal Republic Germany

Maurizio Zocchi
c/o TECOGRAF
via Plinio 11
20100 Milano Italy

General Delivery

From the President

Bart Childs

I would like to call your attention to several items, some of which should appear elsewhere in this issue.

First, our financial position has improved greatly. This is primarily due to the efforts of Ray Goucher, Karen Butler, and Stephan v. Bechtolsheim.

Second, we grew considerably during the past year and this has also contributed to our success. The support of the TEX vendors has been significant in this regard too.

Third, the schedule for 1987 is set. Our annual meeting is set for Seattle and we look forward to it. The courses Ray has scheduled are realistic. The membership as a whole will gain by encouraging others to attend these courses. (Calendar, page 79.)

Fourth, TUG is looking for a permanent employee of the TEX and systems type. Hopefully, that individual will also be able to teach some of the courses. (See cover 3.)

Fifth, I have asked Jackie Damrau, University of New Mexico, to chair the Donald E. Knuth Scholarship Committee. (Page 6.)

Finally, Patrick Ion, Michael Ferguson, and Calvin Jackson have agreed to serve as the nominations committee. This year we have to find a replacement for Sam Whidden and myself. We owe a great deal to Sam and he has given a polite ultimatum that it is time for new blood in the Treasurer's office.

Acknowledgment of Contributions

The Officers and Steering Committee gratefully acknowledge receipt of royalties and other contributions to TUG from several sources during 1986:

- From the sale of Don Knuth's *TEXbook*, METAFONT*book* and related software, royalties of $6,757. (Total royalties from 1984–1986 amount to nearly $14,000.)
- From David Kellerman and Barry Smith, of Kellerman & Smith, royalties of $4,820 from fees paid to them for distribution of the VAX/VMS version of the WEB sources of TEX. David and Barry also contributed $4,000 toward the cost of the experimental issue, *TUGboat* 7, No. 1 (1986), which they guest edited. (This was in addition to several thousand dollars of editorial and design costs incurred by them directly.) An additional $1,000 was contributed to the Donald E. Knuth Scholarship Fund to support the Scholarship winner's attendance at the Annual Meeting.
- Proceeds from the sale of publications made available to TUG:

  Arthur Samuel's First Grade TEX: 1,200 copies, $7,208.

  LATEX Command Summary, by Lloyd Botway and Chris Biemersderfer: 640 copies, $3,208.

  Michael Urban's An Introduction to LATEX: 466 copies, $3,728.

  Lear Siegler's VAX Language-Sensitive Editor (LSEDIT) Quick Reference Guide for use with the LATEX Environment, with the LATEX Style Templates: 50 sets, $3,265. Thanks are due to Kathy Hornbach for her efforts in arranging to make this available.
- Contributions of $2,000 were received from David Rodgers on behalf of ArborText, Inc.
- Special thanks to TUG supporters at Addison-Wesley Publishing Company, the American Mathematical Society, and Personal TEX for arranging for TUG to purchase publications at discounted prices. Also, we wish to express our appreciation to Kellerman & Smith, Personal TEX, Addison-Wesley, and a large number of our member organizations who have shown their strong support for TUG by purchasing thousands of publications through TUG.
- Lastly, we wish to acknowledge the special and unique contribution Barbara Beeton has made in her capacity as the Editor of *TUGboat*. Her tireless efforts and dedication have resulted in a highly informative and indispensable newsletter year after year. In 1986 she donated nearly 300 hours of her own time, thereby saving TUG many thousands of dollars.

TUG sincerely appreciates these very generous contributions.

Samuel B. Whidden, Treasurer

Donald E. Knuth Scholarship

TUG is pleased to announce the Second Annual "Donald E. Knuth Scholarship" competition. This year **two** Scholarships will be awarded. The awards consists of an all-expense-paid trip to TUG's 1987 Annual Meeting and the Short Course offered immediately following the meeting. Funding for one of the scholarships will be provided by Personal TEX, Inc., Mill Valley, California, and the other by the TEX Users Group. The competition is open to all 1987 TUG members holding support positions that are secretarial, clerical or editorial in nature.

To enter the competition, applicants should submit to the Scholarship Committee by May 15, 1987, the input file and final TEX output of a project that displays originality, knowledge of TEX, and good TEXnique. The project may make use of a macro package, either a public one such as LATEX or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge. Along with the TEX files, each applicant should submit a letter stating his/her job title, with a brief description of duties and responsibilities, and affirming that he/she will be able to attend the Annual Meeting and Short Course at the University of Washington, Seattle, Washington, August 24–28, 1987.

Selection of the scholarship recipient will be based on the TEX sample. Judging will take place May 16 – June 15, and the winner will be notified by mail after June 16.

The Scholarship Committee consists of Jackie Damrau, University of New Mexico, chairperson; Laurie Mann, Stratus Computer; Anne Smith, Fairchild CIM Engineering; and Nelson Beebe, University of Utah. All applications should be submitted to the Committee at the following address:

Jackie Damrau
Department of Mathematics and Statistics
University of New Mexico
Albuquerque, New Mexico 87131

It happened

Don Knuth
26 January 87

After 14 months with no bug reports, I was getting pretty confident that we could declare TEX to be bug-free. But Saturday night I tried something different (for me) and guess what: There will be a version 2.1.

If you have a tendency to type a space after a hyphen or a dash, you might not have been getting optimum line breaks, since TEX didn't compute the correct 'badness' of the line following such a break. The incorrect badnesses tended to be masked off because they were associated with an extra penalty; but they could give strange behavior. I'm so sorry.

The reward for the next bug will be $40.96.

Editor's note: This item appeared first in TEXhax, and is reprinted with permission. The changes for version 2.1 are included in the current edition of *Computers & Typesetting: Errata and changes*, among the changes to Volume B, dated 1/28/87. Don had sent the details "to all the main implementors" by the end of January, so by the time you read this, the new version should be available for most implementations; check with your supplier or site coordinator for details.

News of TEX Users in Mexico

Kurt Bernardo Wolf
Universidad National Autónoma de México

The first national meeting of TEX users in Mexico, "Grupo de Usuarios de TEX", was held in Cuernavaca on December 15, 1986. The 57 participants represented academic institutions, private enterprises and government agencies; the National University (UNAM) was represented by 32 participants.

There are presently 30 TEX stations in Mexico. It was agreed that the Publications Department of the UNAM School of Science would serve as information clearing house and link with the TEX Users Group Headquarters.

Program of the Users Group meeting

Mesa redonda:
- Estado actual de la tipografía en México;
- Tipografía convencional y TEX;
- Tipografía científica, problemas y logros.

Conferencia:
- Sistemas TEX: presente y futuro, por Max Díaz.

Exposición de equipo por parte de **Colorimetría, S.A.**

Mesa redonda:
- TEX y los sistemas de información;
- Normas tipográficas y el lenguaje científico;
- Formación de un Grupo de Usuarios de TEX en México.

A Manual of Scientific Typography for Spanish

In May 1986, the first edition of *Manual de lenguaje y typografía científica en castellano* was published by Editorial Trillas SA. This manual was written in TEX at the Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, UNAM. As far as we know, it is the first TEX book in Spanish.

This volume covers, for Spanish authors of scientific manuscripts, grammar, scientific notation and terminology, as well as the duties of author, editor and typographer in bringing such manuscripts into print. Use of the book is not bound to the use of TEX, but is applicable to TEX manuscripts in the same way as Ellen Swanson's book, *Mathematics into Type*. The intended audience for this book is the professional mathematician plus the graduating student population.

Further information on this manual can be obtained from the publisher:

Editorial Trillas, S. A. et C. V.
Av. Río Churubusco 385
Col. Pedro María Anaya
Deleg. Benito Juárez
03340, México, D. F.

Attention, Australian TUG Members

Linda Krick
Anglo-Australian Observatory

I wish to inform the Australian members of TUG that copies of John Sauter's METAFONT files and procedures to build them (TUGboat Vol. 7, No. 3, pp. 151–152) are available from me or the DECUS librarian here in Sydney.

John sent me a tape and permission to copy it for any TUG members who request it. This should eliminate high postage costs and waiting time. I have passed a copy to the DECUS librarian, who has agreed to copy it for any DECUS members.

I have also obtained a tape and permission to copy James Alexander's Tib (TUGboat Vol. 7, No. 3, pp. 138–140). Distribution arrangements will be similar to those for the METAFONT tape.

Copies will be made onto user-supplied magnetic tapes at either 800, 1600 or 6250 bpi. Members who wish to obtain a copy of one or both of these packages should send one tape for each package desired (1200′ for 6250, 2400′ for the smaller densities) to either the DECUS librarian at DECUS, in Chatswood, or to me, at the address below, stating what density to make the copy.

Linda Krick
Anglo-Australian Observatory
P. O. Box 296
Epping, N.S.W. 2121 Australia

Scientific Book Publishing Using TEX

Tony Siegman
Stanford University

A few months ago my publisher and I completed the final tasks in the production of a very large scientific textbook (31 chapters, 1300 pages, 600 figures, 2000 display equations) using TEX. This included preparation of both the source files and the final typeset pages using computer-based text-editing and typesetting facilities at Stanford University. In response to suggestions from Eric Berg of Stanford and Ray Goucher of TUG, this note is an attempt to set down a few of the lessons I learned about producing a book using TEX, including a few of the things I wish I'd thought about earlier in the project. These comments are focused primarily on organizational and procedural questions, rather than on technical problems involving TEX coding, macro packages, or computer hardware.

Book publishing is an old and highly refined art, with a highly developed and widely standardized set of procedures as to how an author's work goes from initial manuscript to bound books. Traditional book publishing is also an industry marked by a great deal of subcontracting. Book design, copy editing, preparation of art work, typesetting, printing and binding, and even publication coordination are very often subcontracted out by the primary publisher to firms who supply those services to many different publishers. Most major publishers, in fact, have probably never owned a printing press or even a typesetting machine.

This widespread use of subcontracting, along with the traditional character of the publishing industry, means that all the procedural steps from manuscript to final book — the way the work flow is organized and scheduled, the way records are kept and objects are labelled, the proofreaders' and printers' marks that are used, the understandings as to who does what and in what order — have become highly standardized and widely understood by all concerned. Technological developments which interrupt or short-circuit this understanding have a disturbing effect.

(Widespread subcontracting of the production steps has also had a second more subjective effect, in my opinion. It seems to me that many traditional publishers have not, at least in the past, developed a technologically oriented view of their industry. Over many years, when I wanted to converse with editors and publishers about word processors, text-editing programs, holograms, floppy disks, laser printers, and their impact on publishing technology, the publishing people seemed to me to focus only on things like printers' union demands, and whether it was better to send material to England or Asia for low-cost typesetting, They seemed to me to have little interest in technological developments in their industry. This disinterest in the technological aspects of publishing is of course now rapidly changing.)

Publishing a book using TEX, especially when this involves direct author input of the TEX source code, as well as author (or at least noncommercial) generation of the final typeset pages, obviously throws many of the traditional procedures and organizational arrangements of book publishing into confusion. To gain maximum advantage from book preparation in TEX, while retaining traditional book quality, new and different organizational procedures will have to evolve. I would not want to say myself, from my limited experience to date, how these new procedures should develop. It seems clear that this evolution has yet to be completed, however, or even to have gotten very well started.

In my own case, the source file for my book was prepared, and then expanded, revised, reshuffled, and rearranged through innumerable versions over a 10 year period, largely by myself and my secretary, Judy Clark, working on home and office terminals connected to various Stanford mainframe computers. Early versions started out being formatted in Script on the campus IBM 360/370 mainframes. Things really began moving about 5 years ago when the source files up to that point were transferred first to a university and then a departmental DEC-20 machine, and TEX became available. Five or six years of steadily evolving class notes and draft output were generated, printed first on a Benson Varian 9211 printer and then on one of the earliest Canon/Imagen laser printers (which is still running magnificently today). If we began again today, of course, we would probably carry out all of this manuscript preparation using a PC-based TEX on our Macintoshes, obtaining draft output from a LaserWriter or something similar.

The final book, *Lasers*, is being published by University Science Books, headed by Bruce Armbruster, in Mill Valley, California, and will be distributed overseas by Oxford University Press. The book design was done by Robert Ishi of Oakland, California, and the copy editing by Aidan Kelly of Alameda, California. The art work was drawn by John Choi, draftsman in the Stanford Chemistry Department. TEX macros to implement Bob Ishi's design, along with the massaging of the source file into final shape, were largely done by

Stanford TEXpert Laura Poplin, with assistance from Arthur Ogawa and Eric Berg. Final pages were typeset on the Stanford Computer Science Department's Autologic APS-μ5 by Eric Berg, with counsel from Dave Fuchs. The art work was shot and pasted onto the typeset pages, and the book printed and bound, by the Maple-Vail Book Manufacturing Group. Publication coordination of all these people was accomplished, despite the confusions of a new and unfamiliar approach, by Bernie Scheier of Miller/Scheier Associates, Palo Alto.

Even when I began to use TEX, in its early days, I knew enough to delineate the main structural elements of the book using a simple limited set of macros ("`\chapter`", "`\section`", "`\subsection`", "`\sectionend`", "`\figure`", "`\problemsbegin`", "`\problem`", "`\problemsend`", and so forth), putting as little native TEX code in the source file as possible. I wrote simple definitions of these macros myself for producing draft output, assuming that a hired TEXpert would modify and extend these macros when the final book design was developed. Any special symbols used in the laser field, such as the quantum-mechanical symbol "h-bar", along with any mathematical symbols for which I thought I might later want to change notation, were also represented by control sequences.

I might of course have looked for some standard TEX macro package to employ in doing this. Few TEX macro packages existed at the time I began, however. In addition, my experience with macro packages, in several different situations, has always been that any macro package powerful enough to provide sophisticated capabilities seems inevitably to acquire a complicated syntax. Once the syntax of the macro package becomes sufficiently complicated, it becomes easier to learn the basic language and then name and define your own macros — stealing as many clever wrinkles from other packages as you can — rather than trying to learn the existing package syntax and make it do what you want.

One of the primary concepts in traditional publishing is of course the idea that the production of a book begins with an author's finished manuscript. Publishers assume there's going to be one. Copy editors assume they will have a stack of typewritten pages, on which they will write mysterious editing marks in red, which will eventually be read by the printer and must never be erased, along with editorial corrections in blue, and questions to the author in green, which must be answered in the margins in soft black pencil, and the circled question marks crossed out in purple, and so. This master copy of the manuscript becomes a primary vehicle of communication between all those involved in the production process.

The idea that there is no paper manuscript, only a mysterious (and changeable) source file in a computer somewhere, is immediately upsetting to people in publishing, at least to the traditionalists. In my initial dealings with many of the production people I had the strong impression that what they really wanted most was to get the manuscript out of the g-d d-m computer and into their hands, so that they could start working on it in the way that God intended. (The copy editor did eventually end up sitting at one of our terminals, doing much of his editing of the source file on line.)

Traditional publishers — at least the ones I've encountered — also have a very strong reluctance to begin production work on a book until the author has turned in an absolutely complete, finished version of the manuscript and the associated art work. One can understand this viewpoint. It is probably based in part on well-founded experience that it always takes several times longer than the author promises for a manuscript to go from "virtually complete" to "really complete". Authors are probably also notorious for telling publishers that the first half of their manuscript is absolutely finished and the publisher can start production work on it, while the author cleans up just a few unfinished bits in the second half. The author then comes back later on with expensive last-minute changes in the first half of the manuscript after it is halfway through the production process, thereby throwing everything into confusion.

Setting the production process in motion also has cash flow implications for the publisher — he must start paying the various subcontractors, with sales revenue still far in the future. Production people can also obviously work more efficiently, and follow schedules more effectively, if they have a complete and finished block of work handed to them all at once. Nonetheless, as I'll argue below, making plans as early as possible at least for how the production is to be accomplished is one step that can improve the efficiency and speed of the whole process.

To allow for maximum future flexibility, I had also adopted an approach in which every illustration in the book was given a symbolic name — e.g., "`\pwrflow`" for a diagram of the power flow in a laser system. Each figure was located in the source file by a `\figure` macro with syntax like

```
\figure{\pwrflow}{18}{20}{sidecap}
    {Power flow in a laser system.}
```

The arguments include the figure's symbolic name, its width and height, information as to whether it is a side-caption or bottom-caption figure, and the text for the figure caption. Encountering this macro in the source file bumps the current figure number up by one; assigns that number to the control sequence `\pwrflow`; reserves suitable space in the TEX output (perhaps as a `\topinsert`) for the figure; and inserts a suitably formatted caption. I also wrote the symbolic name ("`pwrflow`") in the upper right corner of the original art work for that figure. The figure was then referred to in the text by its symbolic name, e.g., "The flow of power between different elements in a laser is shown in Figure `\pwrflow`.". With this approach I could obviously insert a new figure anywhere in the book, or move a whole section of the book with its imbedded figures to some other place in the manuscript, at any time before the final TEX run of the source file. So long as I kept all the symbolic names unique, the figure captions, references and renumbering would all be taken care of.

The production people were unable to cope, however, with the idea of illustration copy that was identified by names instead of numbers; and the copy editor and everyone else expressed great unhappiness at a manuscript which referred to "Figure `\pwrflow`" rather than "Figure 8.11". Each piece of art work for a book is supposed to have a figure number, written in red and circled, in the upper right-hand corner. The production coordinator wants to count these, and make sure they're all there; the copy editor wants to see that the figure number written on the art work matches what is in the text. It was easiest eventually to go back, after the final manuscript was frozen, and hand-convert each symbolic name into the corresponding figure number, both on the art work copy and in the source file.

I did not want to write the final macro definitions and TEX coding myself to convert the book designer's specifications into typeset output pages, nor did I want to massage the final page markup and run the phototypesetter myself. Hence it was necessary to find local free-lance TEXperts with publishing skills to carry out these tasks. We were fortunate eventually in finding skilled help in the Stanford community to do this work. People with advanced ability in TEX, and especially people who combine high-level TEX skills with a genuine eye for quality book layout, are not yet thick on the ground, however.

Minor complications also arose in getting the final typeset pages for the book prepared on the typsesetter belonging to the Computer Facilities group in the Stanford Computer Science Department. There were the usual purely technical problems of fonts and communication between machines, which were eventually overcome. Beyond this, however, there were administrative difficulties of the sort that easily arise when commercial and academic worlds meet. Stanford's Computer Science Department views its typesetter as a typical academic facility. It's there and available to faculty, with some staff help available in using it, and its operating costs must be paid for more or less on a cost-of-use basis; but it is not a commercial enterprise. The publication coordinator, concerned with schedules for getting the book printed, bound, and marketed, wanted to be able to make firm commitments to deliver the typeset pages to the printer on a scheduled date. Hence he wished to have a definite business commitment from the typesetting people that the work could be done by a certain date, for a certain fixed price. The Computer Science people, reasonably enough, were not in a position to give that kind of firm guarantee. Things did, of course, all work out in the end. Incidentally, after a small amount of looking we did not find any commercial typesetting firms, at least in the San Francisco Bay Area, that offered commercial-grade book-quality typesetting service starting from TEX source files.

What are some of the recommendations I would make, or steps I would do differently, if I had to do the whole job over again?

1) First of all, I would bring the copy editor and the author together as early as possible in the manuscript preparation, after only a few chapters have been finished. If the author likes to write "A is soft, while B is hard", and the copy editor is to going to insist that this be written "A is soft, whereas B is hard", the author might as well know this as early as possible in the source file preparation. Questions of style, such as how figures and equations are to be numbered and referred to, should also be settled early on, thus avoiding tedious changes in the source file later on.

2) I would also put the book designer and the TEX people in direct contact as early as possible in the production cycle, before either has begun their work. The book designer has primary responsibility for the aesthetic appearance of the book. Not all designers are yet familiar with the capabilities, and limitations, of TEX; some of them may still regard TEX in the same light as more primitive word-processing or text-formatting systems — OK for in-house technical reports, but not really capable of serious bookmaking. Having the book designer

specify the book format in isolation, then giving this format to the TEXpert and asking that it be implemented, does not make the most effective use of the talents on either side.

3) Obtaining aesthetically pleasing page breaks, which also produce the most effective relationship between subheadings, illustrations, tables, and text references to these elements, seems to be the most difficult and tedious part of the whole book production process in TEX. A TEXpert, a publishing person, and potentially the author must work together on many successive drafts, shifting the positions of figure macros, tables and other inserts, juggling headings, and perhaps cutting or expanding the text itself, until TEX yields a layout for each pair of facing pages that is both aesthetically satisfactory and functionally effective.

Since any change at an earlier point in the source file can propagate forward and change subsequent page breaks, a tool for freezing the page breaks as one slowly works forward through the book is essential. TEX can in fact do quite unexpected things in its attempt to minimize penalties on each page. Its behavior can be reminiscent of a *Scientific American* article from some years ago about work scheduling. The article described a situation in which a group of workers successively picked up tasks of different lengths from the top of a pile of work to be done, each worker coming back for a new task as soon as his or her previous task was completed. Under certain circumstances, increasing the number of workers in the team could actually increase the time required to finish the same pile of work. Similarly, shortening a paragraph by a few words on an early page in a chapter, in order to pull back an orphan line that had spilled onto the following page, would sometimes trigger TEX to make readjustments in the page breaks on later pages that actually made the chapter a full page longer.

4) I did not attempt during the writing of this book to mark potential index entries or index terms in the source file, though I thought of doing this by using some kind of "`\indexentry{...}`" macro which could then write a preliminary index file for computer sorting and hand refinement. By the time the book went into production it seemed simpler and quicker to generate the index "by hand", rather than attempting to define such a macro and put the entries into the source file. If I did the whole thing again from scratch, I would probably include some sort of macro tool for marking the starting and stopping points for index terms in the source file. I would want this macro to note the starting and stopping points for the index term by a small marginal note in draft printouts, and to write the index term and the relevant page numbers to an index file in the final TEX run. This index file could then of course be computer-sorted to become the basis for quick preparation of the final index.

All in all, I would certainly produce any future book of this type using TEX, as compared to any other way. Important benefits for author, publisher, and readers that came from preparing this book in TEX included:

1) The ability to generate (easily) many draft chapters as typeset class notes which were available years before the complete book was finished, so that I could obtain student feedback and error checking on them.

2) The ability to make major revisions and rearrangements of the material all through the writing stages, and the ability to make smaller changes, including adding late references and results, revising paragraphs, and correcting errors, up to a very late stage in the final production process.

3) Freedom for the author from the terrible drudgery, after the manuscript has been finished, of having to again reread and correct in minute detail first the galley proofs and then the page proofs. Having been through that labor twice on earlier books, I'd never want to do it again.

4) Speed of publication from finished manuscript to printed book. With better advance organization we might have done things even faster, but the process still took considerably less than the traditional 9-month gestation period from completed manuscript to printed books. It is particularly helpful to be able to obtain instant draft output from a local laser printer at every stage throughout the production process, rather than having to send marked-up manuscript, or galley proofs, off to a distant typesetter and not receive galleys, or corrected page proofs, until weeks or months later. This gives an immediacy to the production process which keeps attention focused on the project.

5) Of great importance, economy. At a time when technical books are typically priced at 10¢ to 12¢ per page, this relatively specialized book with 1300 pages of technical mathematics will carry a retail list price under $60.

Finally, last but far from least, preparing this book in TEX gave me the opportunity to gain some appreciation of the remarkable intellectual achievement represented by the TEX language, and seemingly by everything else that Donald Knuth does.

TEX Output for the Future

Leslie Lamport

It seems clear that dvi files are a TEX idiosyncracy, and the rest of the reasonable world is going to adopt PostScript as the standard device-independent output format. TEX will be a lot more useful, and reach a much wider set of users, if it could produce PostScript instead of dvi output. This would also permit the development of standards for incorporating figures drawn with other systems into TEX documents.

What are the problems involved in doing this? The existence of dvi → PostScript converters indicates that there is no serious problem at the back end. Allowing the incorporation of other PostScript figures into a TEX document simply requires implementors of these converters to agree on a convention for the `\special` commands. Since I gather that there are now just two such dvi → PostScript programs, I urge their authors to agree on some standard that the rest of the world can use.

The more serious problem lies at the front end, with the fonts. A PostScript font comes with an .afm file to specify font metrics. TEX requires a .tfm file that has additional parameters needed to use the font in math mode. Consequently, as I understand the situation, one can use only CMR fonts in math mode. (I suspect that the .amr file also lacks parameters for the proper placement of accents.) It is my understanding that there are no PostScript versions of the CMR fonts; they must be printed by converting each character to a set of pixels, and drawing each pixel individually — a time-consuming operation.

A first solution to this problem might be a method of getting METAFONT to produce PostSript fonts with .tfm files. If METAFONT becomes the wave of the future, with lots of fonts being drawn with it, this will be a satisfactory solution. If, as I suspect will be the situation, METAFONT is ignored by most of the world, one will ultimately, want a method for producing .tfm files for fonts not produced by METAFONT.

The problem of converting TEX to the PostScript world is important to anyone who wants to see TEX survive. It seems to me that the current dvi → PostScript drivers are not a viable long-term solution. I haven't the expertise or the time to contribute much to a solution. However, I'd be happy to do what I can to act as a catalyst. There are a number of people out there who have a financial stake in the survival of TEX; I urge them to start cooperating now on solving this problem.

Software

WEB Adapted to C, Another Approach

Silvio Levy
Princeton University

I read with great interest a recent TUGboat article about a C version of WEB, by Klaus Guntermann and Joachim Schrod (October, 1986). Since I, too, have written such a CWEB program, I would like to share some of my experiences. I will concentrate on the differences between my version and Knuth's original Pascal version.

I start with TANGLE, since it is easier. I decided that TANGLE should respect the user's line breaks and insert `#line` statements into the C file, so that the compiler, debugger, etc., would print messages that refer to the CWEB file and not to the C file, which is difficult to consult. I am very happy with this arrangement, especially in terms of the debugger: I never have to refer to the C file.

Knuth endowed WEB with a macro command `@d` because the generic Pascal does not handle macros. But the C preprocessor has a standard and powerful macro capability, and between having (the traditional) WEB's treatment of macros and C's I prefer the latter option, because of WEB's limitation to one parameter and, even more annoyingly, because of the fact that in WEB you can't use a variable name (even in TEX text!) before defining it in the source file. This second constraint runs counter to the overall philosophy of WEB, which is that things should be introduced where they logically belong; e.g., an *error_message* macro should be introduced in the section that deals with error handling, and it may not be convenient, or even logical, to have that section in the beginning of the source file.

For this reason my version of TANGLE does not process macros; instead it transforms the WEB file's `@d` statements into `#define` statements that it groups at the beginning of the C file. Naturally, `#define` statements can still be interspersed in the C code, if for any reason they should not migrate to the beginning of the C file. This has the disadvantage that one cannot write macros with a variable number of parameters; but in my experience the gain in simplicity and uniformity far outweighs this drawback.

Finally, my version of TANGLE always inserts a blank space after an '=' token. This is because the C compiler, for reasons of backward compatibility,

interprets `x=-1` to mean "subtract 1 from x," which is very annoying. (The current idiom for this instruction is `x-=1`, and has been for over a decade.)

In order to "understand" input code and typeset it correctly, WEAVE's parser transforms it into a sequence of *scraps*. Each scrap has a *category* (or *cat*, in the lingo), which is like its part of speech; when several scraps with the right cats are found in sequence, they "fire" a *production rule*; for this reason I also call them *sparks*, a quasi-anagram of scraps. It turns out that C's syntax is different enough from Pascal's that I needed to rewrite the production rules from scratch. For example, WEAVE should distinguish between the use of '`*`' and '`&`' as unary or binary operators: in the common idiom `char **argv`; both `*`'s "belong" to *argv*, so the output should look something like

char ∗∗*argv*;

Here's what the TEX output of my version of WEAVE looks like:

```
\&{char} ${*}{*}\\{argv};$
```

(I'm thankful to Guntermann and Schrod for pointing out that this makes TEX treat the asterisk as an Ord atom, not as a unary operator; but then I tried making them Op symbols, and the output didn't look as good to me. Op symbols are meant for large operators, and things like log.)

Following the syntax definition of C (appendix A of Kernighan and Ritchie's *The C Programming Language*), I wrote a relatively small set of rules (fewer than in the original WEB) that correctly parses all C constructs, including variable and function definitions. (It can fail spectacularly when module names or macro names are used in unusual ways; then manual formatting is called for.) In addition all variables being defined automatically get an underlined entry in the index. This means that it is no longer necessary to insert `@!` by hand when certain variables are being defined; I only use `@!` in special circumstances.

In C, when you say `typedef double foo`, the indentifier `foo` can no longer be used to hold the value of a variable and it becomes syntactically equivalent to `double`. Thus WEAVE must give it the same syntactic treatment as a reserved word like `double`, and should also give it the same typographical treatment. Furthermore this should preferably be done automatically. Currently my version of WEAVE takes care of this by changing the *ilk* of the identifier at parsing time. This is not very elegant, and doesn't work if the **typedef** definition is in a separate file; but then one can use WEB's `@f` control sequence. There is also a new control sequence `@i` which works like `#include`, but actually does include the file into WEAVE's input.

Thanks to these changes, if I write

```
double inner_prod(vec1,vec2)
double vec1[dim],vec2[dim];
```

the variables *inner_prod*, *vec1* and *vec2* automatically get an underlined reference in the index; and if I write

```
typedef double vector[dim];
```

the word **vector** will from now on appear in boldface, and its "part of speech" becomes the same as that of **double**.

The last addition I made to WEAVE doesn't show in the output, but it simplifies the grammar a lot. In the original WEB sparks of certain cats can be printed in math mode only, others in either mode and others in non-math mode only. With the relatively more complex grammar of C this scheme would imply a great increase in the number of cats and of production rules. Guntermann and Schrod's solution (letter of December 11, 1986) was to typeset everything in math mode, and have the TEX macros for the various output tokens switch back to non-math when necessary (using the `\ifmmode` primitive). My solution is somewhat different: my sparks have a new attribute, their *mathness*, which is independent of their cat. When a production rule is fired, there is a special bit of code that inserts a '`$`' between sparks of different mathness, but the grammar itself doesn't have to contain any mathness information. This makes WEAVE run about 2% slower, but TEX'ing the WEAVE'd file is faster because TEX doesn't have to check the modes for lots of control sequences.

In conclusion, I am quite happy with CWEB, and do all my programming in it. CWEB itself is written in CWEB. Although I still consider the program experimental, I'm distributing it to interested people, and looking forward to comments and suggestions for improvements.

My heartfelt thanks to Klaus Guntermann and Joachim Schrod, for their helpful correspondence, and to Helmut Jürgensen and Barbara Beeton for inviting me to send this paper.

Mixing right-to-left texts with left-to-right texts

Donald E. Knuth and Pierre MacKay

TeX was designed to produce documents that are read from left-to-right and top-to-bottom, according to the conventions of English and other Western languages. If such documents are turned 90°, they can also be read from top-to-bottom and right-to-left, as in Japan. Another 90° or 180° turn yields documents that are readable from right-to-left and bottom-to-top, or from bottom-to-top and left-to-right, in case a need for such conventions ever arises. However, TeX as it stands is not suitable for languages like Arabic or Hebrew, which are right-to-left and top-to-bottom.

It would not be difficult to use TeX for documents that are purely Arabic or purely Hebrew, by essentially producing the mirror image of whatever document is desired. A raster-oriented printing device could easily be programmed to reflect the bits from right to left as it puts them on the pages. (This is sometimes called "T-shirt mode", because it can be used to make iron-on transfers that produce readable T-shirt messages, when English language output is transferred to cloth after being printed in mirror image.)

Complications arise, however, when left-to-right conventions are mixed with right-to-left conventions in the same document. Consider an Arabic/English dictionary, or a Bible commentary that quotes Hebrew, or a Middle-Eastern encyclopedia that refers to Western names in roman letters; such documents, and many others, must go both ways.

The purpose of this paper is to clarify the issues involved in mixed-direction document production, from the standpoint of a Western author or reader or software implementor. We shall also consider changes to TeX that will extend it to a bidirectional formatting system.

1. *Terminology and conventions.* Let us say for convenience that an *L-text* is textual material that is meant to be read from left to right, and an *R-text* is textual material that is meant to be read from right to left. Similarly we might say that English and Spanish are L-languages, while Arabic and Hebrew are R-languages.

In order to make this paper intelligible to English readers who are unfamiliar with R-languages, we shall use "reflected English", i.e., **hsilgnE**, as an R-language. All texts in reflected English will be typeset in **dednetxE dloB nredoM retupmoC** type, which is a reflected version of Computer Modern Bold Extended type. To translate from English to **hsilgnE** and back again, one simply needs to reverse the order of reading. Both English and **hsilgnE** are pronounced in the same way, except that **hsilgnE** should be spoken in a louder and/or deeper voice, so that a listener can distinguish it.

2. *The simplest case.* It's not difficult to typeset single R-language words in an L-text document. TeX will work fine if you never need to deal with R-texts of more than one word at a time; all you have to do is figure out a macro that will reverse isolated words.

Let's suppose that we want to type '`the |English| script`' in order to typeset 'the **hsilgnE** script' with TeX. All we need is a font for **hsilgnE**, called `xbmc10`, say, and the following macros:

```
\font\revrm=xbmc10       \hyphenchar\revrm=-1
\catcode'\|=\active
\def|#1|{{\revrm\reflect#1\empty\tcelfer}}
\def\reflect#1#2\tcelfer{\ifx#1\empty\else\reflect#2\tcelfer#1\fi}
```

(The `xbmc10` font can be generated like `cmbx10` with the extra METAFONT statement

```
extra_endchar := extra_endchar &
 "currentpicture:=currentpicture reflectedabout((.5[l,r],0),(.5[l,r],1))"
```

added to the parameter file. It has the same character widths as `cmbx10`.)

3. *Alternating texts.* But the simple approach sketched above does not work when there are multiword R-text phrases, i.e., **sesarhp txet-R drowitlum**, embedded in an L-text document—because of the possibility

of line breaks, i.e., **because of the possibility of line breaks**. For example, let's consider the problem of typesetting the following paragraph:*

```
Leonardo da Vinci made a sweeping statement in his notebooks:
|``Let no one who is not a mathematician read my works.''|
In fact, he said it twice, so he probably meant it.
```

Here are samples of the proper results, considering two different column widths:

Leonardo da Vinci made a sweeping statement in his notebooks: **"Let no one who is not a mathematician read my works."** In fact, he said it twice, so he probably meant it.

Leonardo da Vinci made a sweeping statement in his notebooks: **"Let no one who is not a mathemati-cian read my works."** In fact, he said it twice, so he probably meant it.

Notice that the R-text in each line is reflected; in particular, a hyphen that has been inserted at the right of an R-segment will appear at the left of that segment.

How can we get TEX to do this? The best approach is probably to extend the driver programs that produce printed output from the DVI files that TEX writes, instead of trying to do tricky things with TEX macros. Then TEX itself merely needs to put special codes into the DVI output files, in order to tell the "DVI-IVD" drivers what to do.

For example, one idea that almost works is to put '\special{R}' just before an R-text begins, and '\special{L}' just after it ends. In other words, we can change the '|' macro in our earlier example to the simple form

```
\def|#1|{{\revrm\special{R}#1\special{L}}}
```

which does not actually reverse the characters; we can also leave the '\hyphenchar' of \revrm at its normal value, so that R-texts will be hyphenated. Line breaking will proceed in the normal way, and it will be the job of the DVI-IVD driver program to reflect every segment that it sees between an R and an L.

Reflecting might involve arbitrary combinations of characters, rules, accents, kerns, etc.; for example, the R-text might be in **français**, or it might even refer to **TEX**!

4. An approach to implementation. In order to understand how DVI-IVD programs might do the required tasks, we need to look into the information that TEX puts into a DVI file. The basic idea is that whenever TEX outputs an hbox or a vbox, the DVI file gets a '*push*' command, followed by various commands to typeset the box contents, followed by a '*pop*' command. Therefore we can try the following strategy:

a) Whenever '\special{R}' is found in the DVI file, remember the current horizontal position h_0 and vertical position v_0; also remember the current location p_0 in the DVI file. Set $c \leftarrow 0$. Then begin to skim the next DVI instructions instead of actually using them for typesetting; but keep updating the horizontal and vertical page positions as usual.

b) When '\special{L}' is found in the DVI file, stop skimming instructions. Then typeset all instructions between p_0 and the current location, in mirror-reflected mode, as explained below.

c) When '*push*' occurs when skimming instructions, increase c by 1.

d) When '*pop*' occurs when skimming instructions, there are two cases. If $c > 0$, decrease c by 1. (This '*pop*' matches a previously skimmed '*push*'.) But if $c = 0$, effectively insert '\special{L}' at this point and '\special{R}' just after the very next '*push*'.

The mirror-reflected mode for DVI commands in positions p_0 to p_1 in the DVI file, beginning at (h_0, v_0) and ending at (h_1, v_1), works like this: A character of width w whose box sits on the baseline between (h, v) and $(h+w, v)$ in normal mode should be placed so that its box sits on the baseline between $(h'-w, v)$ and (h', v) in mirror mode, where h' is defined by the equation

$$h - h_0 \;=\; h_1 - h'.$$

Similarly, a rule of width w whose lower edge runs from (h, v) to $(h+w, v)$ in normal mode should run from $(h'-w, v)$ to (h', v) in mirror mode.

* After Leonardo lost the use of his right hand, he began to make lefthanded notes in mirror writing. Of course, he actually wrote in **Italian** instead of **English**.

5. Fixing bugs. We stated above that the approach just sketched will "almost" work. But it can fail in three ways, when combined with the full generality of TEX. First, there might be material "between the lines" that is inserted by `\vadjust` commands; this material might improperly be treated as R-text. Second, the suggested mechanism doesn't always find the correct left edge of segments that are being reflected, since the reflection should not always begin at the extreme left edge of a typeset line; it should begin after the `\leftskip` glue and before other initial spacing due to things like accent positioning. Third, certain tricks that involve '`\unhbox`' can make entire lines disappear from the DVI file; however, this problem is not as serious as the other two, because people shouldn't be playing such tricks.

A much more reliable and robust scheme can be obtained by building a specially extended version of TEX, which puts matching special commands into every line that has reflected material. It is not difficult to add this additional activity to TEX's existing line-breaking mechanism; the details appear in an appendix below. When this change has been made, parts (c) and (d) of the DVI-IVD skimming algorithm can be eliminated, since the case $c = 0$ will never arise in part (d).

6. L-chauvinism. We have been discussing mixed documents as if they always consist of R-texts inserted into L-texts; but people whose native script is right-to-left naturally think of mixed documents as the insertion of L-texts into R-texts. In fact, there are two ways to read every page of a document, one in which the eye begins to scan each line at the left and one in which the eye begins to scan each line at the right.

The Leonardo illustration above is an example of the first kind, and we shall call it an *L-document*. To read a given line of an L-document, you start at the left and read any L-text that you see. Whenever your eyes encounter an R-character, they skim ahead to the end of the next R-segment (i.e., until the next L-character, or until the end of the line, whichever comes first); then you read the R-segment right-to-left, and continue as before. The rules for reading an R-document are similar, but with right and left reversed.

It's usually possible to distinguish an L-document from an R-document because of the indentation on the first line of a paragraph and/or the blank space on the last line. For example, the R-documents that correspond to the two L-document settings of the paragraph about Leonardo look like this:

Leonardo da Vinci made a sweeping statement in his **"Let no one who is not a mathematician** notebooks: In fact, he said it twice, so he probably **read my works."** meant it.

Leonardo da Vinci made a sweep- **"Let** ing statement in his notebooks: **no one who is not a mathemati-** **cian read my works."** In fact, he said it twice, so he probably meant it.

We can imagine that these R-documents were composed on an R-terminal and processed by **TEX** from an **input file** that looks like this:

```
|Leonardo da Vinci made a sweeping statement in his notebooks:| ''Let
no one who is not a mathematician read my
works.'' |In fact, he said it twice, so he probably meant it.|
```

In this case it is the L-text, not the R-text, that is enclosed in |'s. (The reader is urged to study this example carefully; there *is* **method** in't!)

A poet could presumably construct interesting poems that have both L-meanings and R-meanings. when read as L-documents and R-documents.

Notice that our examples from Leonardo have used boldface quotation marks (i.e., the quotation marks of **English**), so that these marks belong to the text being quoted. This may seem erroneous; but it is in fact a necessary convention in documents that are meant to display no favoritism between L-readers and R-readers, because it ensures that the quotation marks will stay with the text being reflected. (See the examples of contemporary typesetting at the end of this paper.) If we had put the quotation marks into English rather than **English**, the R-documents illustrated above would have looked very strange indeed:

Leonardo da Vinci made a sweeping statement in his **Let no one who is not a mathematician**notebooks: " " In fact, he said it twice, so he probably.**read my works**meant it.

Leonardo da Vinci made a sweep- **Let**ing statement in his notebooks: " **no one who is not a mathemati-** **cian read my works.**" In fact, he said it twice, so he probably meant it.

7. Multi-level mixing. The problems of mixed R- and L-typesetting go deeper than this, because there might be an L-text inside an R-text inside an L-text. For example, we might want to typeset a paragraph whose TEX source file looks like this:

```
\R{Alice} said, \R{``You think English is \L{`English written backwards'};
but to me, \L{English} is English written backwards.  I'm sure \L{Knuth}
and \L{MacKay} will both agree with me.''}  And she was right.
```

An intelligent bidirectional reader will want this to be typeset as if it were an R-document inside an L-document. In other words, the eyes of such a reader will naturally scan some of the lines beginning at the left, and some of them beginning at the right. Here are examples of the desired output, set with two different line widths:

Alice said, "You think English is 'English written backwards'; but to me, English is English written backwards. I'm sure Knuth and MacKay will both agree with me." And she was right.

Alice said, "You think English is 'En- English written backwards'; but to me, English is English written backwards. I'm sure Knuth and MacKay will both agree with me." And she was right.

(Look closely.)

Multi-level documents are inherently ambiguous. For example, the right-hand setting above might be interpreted as the result of

```
...\R{... I'm sure and \L{MacKay} will both agree with} Knuth \R{me.''}...
```

and the left-hand setting would also result from a source file like this(!)

```
\indent\R{``You think English is \L{said,} Alice
\L{`English}; but to me,} written backwards'
\R{written backwards.} \R{\L{English} is English}
will both} MacKay \R{and} Knuth \R{I'm sure
\L{And she} agree with me.''} was right.
```

except for slight differences in spacing due to TEX's "spacefactor" for punctuation.

In general, we have `\R{\L{a}\L{b}}` = ba, hence any permutation of the characters on each line is theoretically possible. A reader has to figure out which of the different ways to parse each line makes most sense. Yet there is unanimous agreement in Middle Eastern countries that a mixture of L-document and R-document styles is preferable to an unambiguous insistence on L-reading or R-reading throughout a document, because it is so natural and because the actual ambiguities arise rarely in practice. The quotation marks in the example above make it possible to reconstruct the invisible `\R`'s and `\L`'s; in this way an author can cooperate with a literate reader so that the meaning is clear.

Multi-level texts arise not only when quotes are inside quotes or when R-document footnotes or illustrations are attached to L-documents; they also arise when mathematics is embedded in R-text. For example, consider the TEX source code

```
The \R{English} version of `the famous identity $e^{i\pi}+1=0$ due to Euler'
is \R{`the famous identity $e^{i\pi}+1=0$ due to Euler'}.
```

It should be typeset like this:

The English version of 'the famous identity $e^{i\pi}+1=0$ due to Euler' is 'the famous identity $e^{i\pi}+1=0$ due to Euler'.

An extension of TEX called TEX-XET, described in the appendix, properly handles multi-level mixtures including math, as well as the simpler case of alternating R-texts and L-texts.

8. Conclusions. When right-to-left and left-to-right texts are mixed in the same document, problems can arise that are more subtle than simple examples might suggest. The difficulties can be overcome by extending TEX to TEX-XET and by extending DVI drivers to DVI-IVD drivers. Neither of these extensions is extremely complex.

585. The description of DVI commands is augmented by two new ones at the end:

begin_reflect 250. Begin a (possibly recursive) reflected segment.

end_reflect 251. End a (possibly recursive) reflected segment.

Commands 250–255 are undefined in normal DVI files, but 250 and 251 are permitted in the special 'DVI-IVD' files produced by this variant of TEX.

When a DVI-IVD driver encounters a *begin_reflect* command, it should skim ahead (as previously described) until finding the matching *end_reflect*; these will be properly nested with respect to each other and with respect to *push* and *pop*. After skimming has located a segment of material to be reflected, that segment should be re-scanned and obeyed in mirror-image mode as described earlier. The reflected segment might recursively involve *begin_reflect*/*end_reflect* pairs that need to be reflected again.

586. Two new definitions are needed:

```
define begin_reflect = 250  { begin a reflected segment (allowed in DVI-IVD files only) }
define end_reflect = 251  { end a reflected segment (allowed in DVI-IVD files only) }
```

638. At the beginning of *ship_out*, we will initialize a stack of \beginL and \beginR instructions that are currently in force; this is called the LR stack, and it is maintained with the help of two global variables called *LR_ptr* and *LR_tmp* that will be defined later. The instructions inserted here (just before testing if *tracing_output* > 0) say that on the outermost level we are typesetting in left-to-right mode. The opening '**begin**' is replaced by:

```
begin LR_ptr ← get_avail; info(LR_ptr) ← 0;  { begin_L_code at outer level }
```

639. At the end of *ship_out*, we want to clear out the LR stack. Thus, '*flush_node_list*(*p*)' is replaced by:

```
flush_node_list(p); ⟨ Flush the LR stack 1382 ⟩;
```

649. The *hpack* routine is modified to keep an LR stack as it packages a horizontal list, so that errors of mismatched \beginL...\endL and \beginR...\endR pairs can be detected and corrected. Changes are needed here at the beginning of the procedure and at the end.

```
function hpack(p : pointer; w : scaled; m : small_number): pointer;
                    ⋮
  b: integer;  { badness of the new box }
  LR_ptr, LR_tmp: pointer;  { for LR stack maintenance }
  LR_problems: integer;  { counts missing begins and ends }
begin LR_ptr ← null; LR_problems ← 0;
r ← get_node(box_node_size);
                    ⋮
common_ending: ⟨ Finish issuing a diagnostic message for an overfull or underfull hbox 663 ⟩;
exit: ⟨ Check for LR anomalies at the end of hpack 1387 ⟩;
hpack ← r;
end;
```

877. Similarly, the *post_line_break* should keep an LR stack, so that it can output \endL or \endR instructions at the ends of lines and \beginL or \beginR instructions at the beginnings of lines. Changes occur at the beginning and the end of this procedure:

```
procedure post_line_break(final_widow_penalty : integer);
                    ⋮
  cur_line: halfword;  { the current line number being justified }
  LR_ptr, LR_tmp: pointer;  { for LR stack maintenance }
begin LR_ptr ← null;
⟨ Reverse the links of the relevant passive nodes, setting cur_p to the first breakpoint 878 ⟩;
```

```
        ⋮
prev_graf ← best_line − 1: ⟨ Flush the LR stack 1382 ⟩;
end;
```

880. The new actions to be performed when broken lines are being packaged are accomplished by three new steps added to this section of the program.

```
⟨ Justify the line ending at breakpoint cur_p, and append it to the current vertical list, together with
      associated penalties and other insertions 880 ⟩ ≡
  ⟨ Insert LR nodes at the beginning of the current line 1383 ⟩;
  ⟨ Adjust the LR stack based on LR nodes in this line 1384 ⟩;
  ⟨ Modify the end of the line to reflect the nature of the break and to include \rightskip; also set the
      proper value of disc_break 881 ⟩;
  ⟨ Insert LR nodes at the end of the current line 1385 ⟩;
  ⟨ Put the \leftskip glue at the left and detach this line 887 ⟩;
        ⋮
```

1090. We add '*vmode* + *LR*' as a new subcase after '*vmode* + *ex_space*' here. This means that the new primitive operations will become instances of what *The TEXbook* calls a ⟨horizontal command⟩.

1196. Math-in-text will be formatted left-to-right, because two new 'append' instructions are inserted into this section of the code.

```
⟨ Finish math in text 1196 ⟩ ≡
  begin tail_append(new_math(math_surround.before));
  ⟨ Append a begin_L to the tail of the current list 1380 ⟩;
  cur_mlist ← p; cur_style ← text_style; mlist_penalties ← (mode > 0); mlist_to_hlist;
  link(tail) ← link(temp_head);
  while link(tail) ≠ null do tail ← link(tail);
  ⟨ Append an end_L to the tail of the current list 1381 ⟩;
  tail_append(new_maih(math_surround.after)); space_factor ← 1000; unsave;
  end
```

1341. The new primitive operations put new kinds of whatsit nodes into horizontal lists. Therefore two additional definitions are needed here:

```
define LR_node = 4   { subtype in whatsits that represent \beginL. etc. }
define LR_type(#) ≡ mem[# + 1].int   { the sub-subtype }
```

1344. Here's where the new primitives get established.

```
define immediate_code = 4   { command modifier for \immediate }
define begin_L_code = 0   { command modifier for \beginL }
define begin_R_code = 1   { command modifier for \beginR }
define end_L_code = 2   { command modifier for \endL }
define end_R_code = 3   { command modifier for \endR }
define begin_LR(#) ≡ (LR_type(#) < end_L_code)
define begin_LR_type(#) ≡ (LR_type(#) − end_L_code)
⟨ Put each of TEX's primitives into the hash table 226 ⟩ +≡
  primitive("beginL", LR, begin_L_code);
  primitive("beginR", LR, begin_R_code);
  primitive("endL", LR, end_L_code);
  primitive("endR", LR, end_R_code);
  primitive("openout", extension, open_node);
        ⋮
```

1346. The new primitives call for a new case of cases here.

```
LR: case chr_code of
  begin_L_code: print_esc("beginL");
  begin_R_code: print_esc("beginR");
  end_L_code: print_esc("endL");
  othercases print_esc("endR")
  endcases;
```

1356. We also need to be able to display the newfangled whatsits.

```
LR_node: case LR_type(p) of
    begin_L_code: print_esc("beginL");
    begin_R_code: print_esc("beginR");
    end_L_code: print_esc("endL");
    othercases print_esc("endR")
    endcases;
```

1357, 1358. Copying and deleting the new nodes is easy, since they can be handled just like the `\closeout` nodes already present. We simply replace '*close_node*' by '*close_node*, *LR_node*' in these two sections.

1360. We used to *do_nothing* here, but now we must *do_something*:

```
⟨ Incorporate a whatsit node into an hbox 1360 ⟩ ≡
  if subtype(p) = LR_node then ⟨ Adjust the LR stack for the hpack routine 1386 ⟩
```

This code is used in section 651.

1366.

```
⟨ Output the whatsit node p in an hlist 1366 ⟩ ≡
  if subtype(p) ≠ LR_node then out_what(p)
  else ⟨ Output a reflection instruction if the direction has changed 1388 ⟩
```

This code is used in section 622.

1376. Most of the changes have been saved up for the end, so that the section numbers of TEX in [2] can be left unchanged. Now we come to the real guts of this extension to mixed-direction texts.

First, we allow the new primitives in horizontal mode, but not in math mode:

```
⟨ Cases of main_control that build boxes and lists 1056 ⟩ +≡
hmode + LR: begin new_whatsit(LR_node, small_node_size); LR_type(tail) ← cur_chr; end;
mmode + LR: report_illegal_case;
```

1377. A number of routines are based on a stack of one-word nodes whose *info* fields contain either *begin_L_code* or *begin_R_code*. The top of the stack is pointed to by *LR_ptr*, and an auxiliary variable *LR_tmp* is available for stack manipulation.

```
⟨ Global variables 13 ⟩ +≡
LR_ptr, LR_tmp: pointer; { stack of LR codes and temp for manipulation }
```

1378.

```
⟨ Declare functions needed for special kinds of nodes 1378 ⟩ ≡
function new_LR(s : small_number): pointer;
  var p: pointer; { the new node }
  begin p ← get_node(small_node_size); type(p) ← whatsit_node; subtype(p) ← LR_node; LR_type(p) ← s;
  new_LR ← p;
  end;
```

See also section 1379.

This code is used in section 161.

1379.

```
⟨ Declare functions needed for special kinds of nodes 1378 ⟩ +≡
function safe_info(p : pointer): integer;
  begin if p = null then safe_info ← −1 else safe_info ← info(p);
  end;
```

1380. ⟨ Append a *begin_L* to the tail of the current list 1380 ⟩ ≡
tail_append(*new_LR*(*begin_L_code*))
This code is used in section 1196.

1381. ⟨ Append an *end_L* to the tail of the current list 1381 ⟩ ≡
tail_append(*new_LR*(*end_L_code*))
This code is used in section 1196.

1382. When the stack-manipulation macros of this section are used below, variables *LR_ptr* and *LR_tmp* might be the global variables declared above, or they might be local to *hpack* or *post_line_break*.

```
define push_LR(#) ≡
        begin LR_tmp ← get_avail; info(LR_tmp) ← LR_type(#); link(LR_tmp) ← LR_ptr;
        LR_ptr ← LR_tmp;
        end
define pop_LR ≡
        begin LR_tmp ← LR_ptr; LR_ptr ← link(LR_tmp); free_avail(LR_tmp);
        end
```

⟨ Flush the LR stack 1382 ⟩ ≡

```
while LR_ptr ≠ null do pop_LR
```

This code is used in sections 639 and 877.

1383. ⟨ Insert LR nodes at the beginning of the current line 1383 ⟩ ≡

```
while LR_ptr ≠ null do
  begin LR_tmp ← new_LR(info(LR_ptr)); link(LR_tmp) ← link(temp_head);
  link(temp_head) ← LR_tmp; pop_LR;
  end
```

This code is used in section 880.

1384. ⟨ Adjust the LR stack based on LR nodes in this line 1384 ⟩ ≡

```
q ← link(temp_head);
while q ≠ cur_break(cur_p) do
  begin if ¬is_char_node(q) then
    if type(q) = whatsit_node then
      if subtype(q) = LR_node then
        if begin_LR(q) then push_LR(q)
        else if LR_ptr ≠ null then
            if info(LR_ptr) = begin_LR_type(q) then pop_LR;
  q ← link(q);
  end
```

This code is used in section 880.

1385. We use the fact that q now points to the node with `\rightskip` glue.

⟨ Insert LR nodes at the end of the current line 1385 ⟩ ≡

```
if LR_ptr ≠ null then
  begin s ← temp_head; r ← link(s);
  while r ≠ q do
    begin s ← r; r ← link(s);
    end;
  r ← LR_ptr;
  while r ≠ null do
    begin LR_tmp ← new_LR(info(r) + end_L_code); link(s) ← LR_tmp; s ← LR_tmp; r ← link(r);
    end;
```

```
    link(s) ← q;
    end
```

This code is used in section 880.

1386. ⟨ Adjust the LR stack for the *hpack* routine 1386 ⟩ ≡

```
  if begin_LR(p) then push_LR(p)
  else if safe_info(LR_ptr) = begin_LR_type(p) then pop_LR
    else begin incr(LR_problems);
      while link(q) ≠ p do q ← link(q);
      link(q) ← link(p); free_node(p, small_node_size); p ← q;
      end
```

This code is used in section 1360.

1387. ⟨ Check for LR anomalies at the end of *hpack* 1387 ⟩ ≡

```
  if LR_ptr ≠ null then
    begin while link(q) ≠ null do q ← link(q);
    repeat link(q) ← new_LR(info(LR_ptr) + end_L_code); q ← link(q);
      LR_problems ← LR_problems + 10000; pop_LR;
    until LR_ptr = null;
    end;
  if LR_problems > 0 then
    begin print_ln; print_nl("\endL␣or␣\endR␣problem␣(");
    print_int(LR_problems div 10000); print("␣missing,␣");
    print_int(LR_problems mod 10000); print("␣extra");
    LR_problems ← 0; goto common_ending;
    end
```

This code is used in section 649.

1388. ⟨ Output a reflection instruction if the direction has changed 1388 ⟩ ≡

```
  if begin_LR(p) then
    begin if safe_info(LR_ptr) ≠ LR_type(p) then
      begin synch_h; synch_v; dvi_out(begin_reflect);
      end;
    push_LR(p);
    end
  else if safe_info(LR_ptr) = begin_LR_type(p) then
      begin pop_LR;
      if info(LR_ptr) + end_L_code ≠ LR_type(p) then
        begin synch_h; synch_v; dvi_out(end_reflect);
        end;
      end
    else confusion("LR")
```

This code is used in section 1366.

Final Important Note

The extensions to TEX just described are "upward compatible" with standard TEX, in the sense that ordinary TEX programs will still run correctly (although more slowly) on TEX-XET. However, TEX-XET must *not* be called a new version of 'TEX', even though it runs all TEX programs; the reason is, of course, that TEX will not run all TEX-XET programs.

A name change is necessary to distinguish all programs that do not agree precisely with the real TEX. Anybody who runs a program called 'TEX' should be able to assume that it will give identical results from all its implementations.

Bibliography

[1] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting* (Reading, Mass.: Addison Wesley, 1986).

[2] Donald E. Knuth, *TEX: The Program*, Volume B of *Computers & Typesetting* (Reading, Mass.: Addison Wesley, 1986).

[3] Pierre MacKay, "Typesetting problem scripts," *Byte* **11**, 2 (February 1986), 201–218.

Examples of Typesetting Practice

1. From *Textus* **5** (1966), p. 12; Magnes Press, Hebrew University of Jerusalem. (Notice the Hebrew quotation marks surrounding the Hebrew title in footnote 6.)

ters adhered,[10] and which may have been similar to that adopted, by normative Jewry presumably somewhat later, during the period of the Second Temple.[10]

Frag. E. Yadin correctly states: "Sanders' cautious indication '103 (? 104)' can now be eliminated" (*ib.*, p. 5).

6 Sanders' *editio princeps* of Ps. 151 already has been discussed by various scholars. The present author deals with the text of Ps. 151, and its literary genre in: מזמורים חיצוניים בלשון העברית מקומראן, *Tarbiz* 35 (1966) 214–228.

7 See: W. Wright, "Some Apocryphal Psalms in Syriac", PSBA 9 (1887) 257–266; M. Noth, "Die fuenf apokryphen Psalmen", ZAW 48 (1930) 1–23.

2. Fragments from the third edition of William Wright's classic nineteenth century grammar of Arabic, volume 2, pages 295–297. (Notice the page break in the midst of right-to-left text, and some left-to-right brackets.)

gnawed at us; كُنْتُمْ خَيْرَ أُمَّةٍ أُخْرِجَتْ لِلنَّاسِ *ye are the best people that has been brought forth (created) for mankind;* مَشَيْنَ كَمَا ٱهْتَزَّتْ رِمَاحٌ تَسَفَّهَتْ أَعَالِيَهَا مَرُّ ٱلرِّيَاحِ ٱلنَّوَاسِمِ *they walked as spears wave, the tops of which are bent by the passing of gentle breezes;* إِنَارَةُ ٱلْعَقْلِ

296 PART THIRD.—*Syntax.* [§ 152

A مَكْسُوفٌ بِطَوْعِ هَوًى *the brightness of the intellect is obscured* (or *eclipsed*) *by obeying lust.* As the above examples show, this agreement

§ 152] *Sentence and its Parts.—Concord of Predicate & Subject.* 297

verb is placed after a collective subject (see § 148); as A وَلٰكِنَّ أَكْثَرَ ٱلنَّاسِ لَا يَشْكُرُونَ *but the greatest part of mankind are thankless;* فَرِيقٌ مِنْهُمْ يَخْشَوْنَ ٱلنَّاسَ *a part of them are afraid of men:* [ٱتْرُكُوا ٱلتُّرْكَ مَا تَرَكُوكُمْ *let the Turks alone as long as they let you alone;* لِأَنَّ جَيْشَهُ هَلَكُوا *because his army had perished*].

3. From page 233 of the same book. Here right-reading texts are equated with = signs; the left sides of each equation are to be read first.

اَلصَّلٰوةُ فِى ٱلسَّاعَةِ, i.e. صَلٰوةُ ٱلسَّاعَةِ ٱلْأُولَى = صَلٰوةُ ٱلْأُولَى ; understood
B جَانِبُ ٱلْغَرْبِىِّ Similarly, some grammarians consider (see § 77). ٱلْأُولَى
مَسْجِدُ ٱلْمَكَانِ ٱلْجَامِعِ = مَسْجِدُ ٱلْجَامِعِ, جَانِبُ ٱلْمَكَانِ ٱلْغَرْبِىِّ =
and ,بَقْلَةُ ٱلْحَبَّةِ ٱلْحَمْقَاءِ = بَقْلَةُ ٱلْحَمْقَاءِ, مَسْجِدُ ٱلْوَقْتِ ٱلْجَامِعِ or
أَفْضَلُ Here too the constructions .دَارُ ٱلْحَيَاةِ ٱلْآخِرَةِ = دَارُ ٱلْآخِرَةِ*

4. From *Bulletin of the Iranian Mathematical Society* **8** (Tehran, 1978), p. 78L. (Left-to-right mathematics in right-to-left text.)

کلمه جدید بازشود ازعبارت زیرتعیین میشود

$$\frac{n-1}{2m} + \frac{2(n-1)(n-2)}{3m^2} + \frac{3(n-1)(n-2)}{4m^3} + \ldots .$$

که تقریباً مساوی $1/(1-\alpha) + [\log_e(1-\alpha)] / \alpha$ است که در آن e=2.71828
میباشد .بنابراین اضافه کردن کلمات به جدول باکمك الگوریتم F کارچند ان

5. From *Introduction to Mathematics* [Mavo Lematematika, printed mirror-reversed] by Abraham A. Fraenkel, vol. 1 (Jerusalem, 1942), p. 38. (Page numbers are '96–90' because '90' and '96' are Hebrew numbers.)

בהשתמשנו במושג הקונגרואנציה, נוכל לכתוב את המשפט הקטן של פ ר מ ה
בצורה: $a^p \equiv a$ (mod. p), ואת משפט ו י ל ס ו ן בצורה:
$1 \cdot 2 \cdot 3 \cdots (p-1) \equiv -1$ (mod. p).

1. מן המלים הרומיות congruens = מתאים, modulus = אופן, מדה.
2. עיין בהוכחה שנתן M. Hamburger בשנת 1896 בכרך ה 116 של ה Journal f. d. reine u. ang. Mathematik (עמ׳ 90–96) לנוסחה של גאוס משנת 1802, הקובעת את התאריך הנוצרי

6. Page 200 of the same book illustrates the difference between ellipses '$\cdots$' in formulas and ellipses in the text. None of this book's math-in-text is broken between lines.

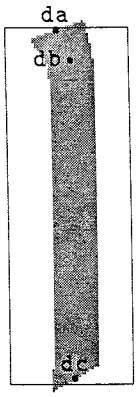
בריבוע של מספר אי־זוגי, ב) שאפשר לפתור את הבעיה אם n מתפרק למספרים
ראשוניים ש ו נ י ם $n = p_1 \cdot p_2 \cdot \cdots \cdot p_k$, ב ת נ א י שהבעיה נפתרת בשביל כל
הערכים $n = p_1$, $n = p_2$, …. $n = p_k$. נסתפק בכך שנבאר את הטענה האחרונה
ביחס לדוגמה $n = 15 = 3 \cdot 5$. ידוע בוודאי לרוב הקוראים שאפשר לבנות בעזרת

⋮

קודם כל יצויין שבמקרה זה, $n = p$, כל שרשי המשואה [1] (לשון אחר:
כל שרשי המשואה $x^p = 1$, פרט ל 1) הנם שרשים „פרימיטיביים" במובן שהוגדר
בעמ׳ 199 ; שהרי כל ערך k מתוך הסדרה $(1, 2, \ldots, p-1)$ זר ל p. יהי אפוא $n = p$,
ו ξ שורש כלשהו של המשואה [1] ; אז יהיו $\xi, \xi^2, \ldots, \xi^{p-1}$ כל שרשיה של
המשואה [1] ¹. השרשים האלה נקראים ש ר ש י ה י ח י ד ה.

Notice how natural this description is: we can specify the apparent width (which in handwriting can be controlled by pressure) and angle of our pen at each of three critical points on our stroke, and we control the curve drawn by "pulling the curve tighter" via the `tension` parameter. Notice, too, that despite all this control, there is still much flexibility here: the values of the three arguments passed to `downstroke` are determined precisely for each character at run time.

Having thus defined `downstroke`, we can describe an

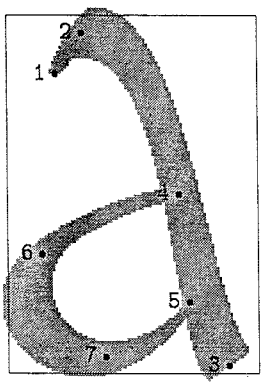


very succinctly indeed:

```
beginchar(73,4.5/60em#,m#,0); "The letter I";
  downstroke(m+verto,left_edge,0-verto);
endchar;
```

Here, by invoking `beginchar`, we assign a suitable ASCII code to the image METAFONT will create, and specify the height, depth, and width of the box in which the character sits. All we need do then is invoke `downstroke` with suitable arguments, and invoke `endchar` to wrap up all loose ends.

Creating a solid working set of subroutines for all the recurring motifs in a font is just the first, albeit rather major, step in the creation of our alphabet. As we have seen, METAFONT takes to this task like a duck to water. It is equally easy to deal with the "odd cases", the letters that are not composed in whole or part of common letter parts. A good example is an




```
beginchar(65,28/60em#,m#,0); "The letter A";
% First, pickup the appropriate pen
  pickup ucpen;
% Describe positioning of pen at
% all key points
  penpos1(uvthinp,180);
  penpos2(1.15uchorzp,90-15);
  penpos3(1.15uchorzp,35);
  penpos4(uvthinp,90+15);
  penpos5(uhthinp,15);
  penpos6(1.15uchorzp,180+15);
  penpos7(3/4uchorzp,270+15);
% Specify the locations of key points
  y4=1/2m;
  y5=1/5m;
  y6=1/3m;
  x6r=left_edge;
  y7r=0;
  x7=1/2[x6r,x5r];
  x1r=1/6w;
  y1r=5/6m;
  y2r=m+verto;
  x2r=1/5[x1r,x3r];
  x3r=right_edge;
  y3l=0-verto;
  z5l=whatever[z3l,z2r];
  z4l=whatever[z3l,z5r];
% Draw through those points
  penstroke z3e{-1,1}..
    tension 2..z2e{-6,-1}..
    tension 4..z1e
  penstroke z4e..
    tension 1.5..z6e..
    z7e..tension 1.15..z5e;
  labels(1,2,3,4,5,6,7);
endchar;
```

This code should give the reader a good feeling for just how easy METAFONT descriptions are. This is probably the most complicated code I had to write for my uncial character set, and yet I believe it is fairly straightforward and easy to follow along in conjunction with the proof.

So, my experiment gave METAFONT good marks for ease and speed of use. The positive reactions I have received thus far on the character set's appearance tend to also rate it favorably on the question of how well it emulates the model. (I welcome the readers' opinions of the appearance of the uncial sample given at the end of this article to add to the evidence here.)

Of course, we cannot expect *any* 300 dpi digitized image to perfectly echo the graceful tapers and curves we are after, but I suspect this METAFONT version could hold its own with another produced by hand for the same resolution. The acid test, though, is to print these fonts on a very high resolution device, and compare that copy to an analogue version.

The final question I hoped to answer regarded the robustness of these METAFONT characters. While I found that I could generate these characters at a good range of point sizes and a good range of resolutions without any complaints from METAFONT, the results on my low resolution printer were unacceptable at small point sizes:

abcdefghijklm
nopqrstuvwxyz
0123456789

Disturbing breaks and fadeouts in the fine lines occurred. While this is bothersome (and contrasts with the results I am used to with METAFONT79), I do not consider it an insurmountable problem. There are certain parameters that can be tweaked (namely `blacker` and `fillin`) that can handle many problems. The overriding impression I get, though, is that METAFONT is a rich enough and powerful enough and precise enough language that with time and care this problem can be dealt with. However, the production of good fonts at typical text sizes for low resolution (300 dpi and under) devices is one task that METAFONT needs to be able to do well, and is one that I intend to focus on in the early part of 1987.

As always, I welcome my readers comments on this article or on any aspect of font design.

G.K.M. Tobin
31 December 1986

Knowledge
is good, method is good
but one thing beyond all others
is necessary: and that is to have
a head, not a pumpkin,
on your shoulders and
brains, not pudding,
in your head.
— a.e. housman

abcdefghijklm
nopqrstuvwxyz
0123456789

This document printed at 300 dpi with experimental Metafoundry fonts.

Write-White Printing Engines and Tuning Fonts with METAFONT

Neenie Billawala

The wide variety of digital printers with different print characteristics presents the fact that the same font data will likely produce just as many results. There are differences in resolution, printer type (laser, photo-digital, dot matrix, screen), and in the characteristics of the marking device. Even within the range of same resolution and type of printer, e.g., 300 dpi laser printer, variations occur.

A most noticeable one occurs between those of the "write-white" and "write-black" variety.

The problem in getting a like result with write-white (ww) and write-black (wb) machines from the same pixel pattern lies in the fact that the effective size and/or shape of each pixel differs; the size is typically smaller in ww machines.

Theoretically, one pixel on a 300dpi printer has a width of 1/300 of an inch. On a ww machine, it's typically less; on a wb machine, it's usually a bit more. The gradations of thickness may look something like the following hypothetical data, actual values will vary.

number of pixels	"ideal width"	write-white width	write-black width
1	1	.8	1.2
2	2	1.8	2.2
3	3	2.8	3.2
4	4	3.8	4.2
10	10	9.8	10.2

(width values are in 1/300 of an inch)

As you can see, the greater the number of pixels, the less important the .4 pixel difference becomes. The greatest discrepancies and problems occur with lines one pixel wide, where the percentage of difference is the greatest.

>> Try the following METAFONT example on different printers. Make an image which alternates a single pixel black vertical line with a single pixel "empty" or white vertical line and print the character on your machine. Do the same with alternating single pixel horizontal lines. Notice the shades of gray, the relative darknesses and the crispness of line. Try this on a write-white printer and then again on a write-black printer. There will probably be a marked difference. If you are more ambitious, repeat the same example with different thicknesses for the black lines and/or white lines. Change the length of your test lines; overlap the horizontal and vertical tests ... Notice also the corners and edges of the lines. You may have to limit the size of your test characters to keep METAFONT from running out of memory. <<

Larger pixel representations of characters are often shown as precise patterns with a clean square representing each pixel. Upon close inspection of an individual pixel, however, you see no precise square, but rather a globular figure, or a starburst pattern, the center being darker with the pattern becoming lighter away from the center. Individual pixels may vary in shape within a range.

The square represents a hypothetical pixel (fig 1a). The circles are the areas where blackness, e.g., toner, may be found. If the outermost edges of the circle fall totally inside the square, the pixel is light, and a series of these in a row will form an unconnected or broken line (fig 1b). On the other hand, if the square falls completely within the circle, the pixels will be relatively dark, and a line of them will be darker than the target pixel size due to overlap (fig 1c). The target pixel size is one that will produce a target line thickness, e.g., 1/300 of an inch, when several of these pixels are lined up. The "ideal" solution lies somewhere in between these two, similar to fig 1d.

Figure 2 demonstrates a simplification of the idea between "write-white" and "write-black" printers. With a wb printer, the circles are blackened, in a ww printer, the circles are white and the areas between them are dark.

At some point, digital font data contains information about which pixels or bits are black and which are white. If the same font data or pixel pattern is produced with printers, each having different output characteristics, the results will vary.

If the line thicknesses or other characteristics of two printers are not identical it will be impossible to produce exactly the same output given the same font data. However, the same font can often be made recognizable and comparable between printers.

The style of a font dictates how well it will hold up, how robust or how fragile it will be. Typically fonts with very thin lines, such as the "modern" fonts, tend to fall apart with ww printers. Many of the Computer Modern fonts fall into this category.

METAFONT offers the capability of changing parameters, modifying a font. Naturally, only those parameters which are included in the design of the font can be changed. It is important to leave the vertical and horizontal width information found in the "tfm" file intact, but it is possible to make slight alterations to characters to get back lines that drop out or fade due to a one pixel thickness.

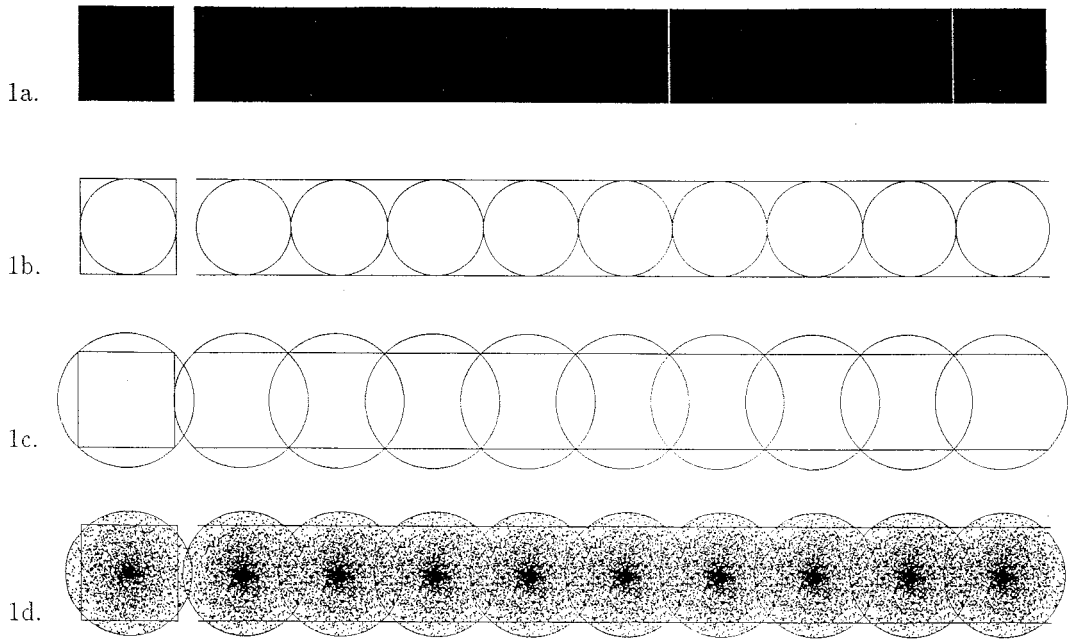


Figure 1. Area occupied by adjoining pixels

Editor's note: The small gaps in the "solid" bar of fig 1a appear to be an artifact of roundoff error; roundoff problems can sometimes be noticed in text output from laser printers, particularly when strings of monospaced type are interspersed with roman text, since the likelihood is small that the width of a monospaced letter is an integral number of pixels at all design sizes and magnifications.

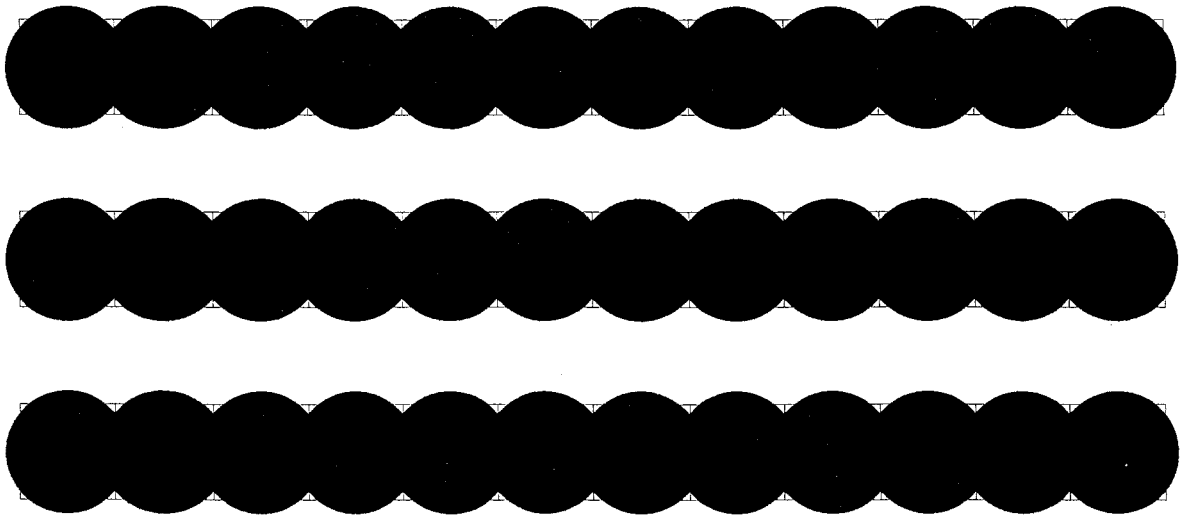


Figure 2. Alternating lines of written and unwritten pixels

At this point, one should consider a few things about the font you are trying to modify. Remember the printer characteristics, and the fact that two very different printers will give dissimilar results. Within this restriction, one generally looks for a representation that is recognizable and close to the "original" or "source" form. You should also consider

- For what purpose is the font being used? If it is for lower resolution previewing proofs for a higher resolution device, then it is essential to maintain "tfm" dimensions.
- How closely does it match its source?
- Is it important that it remain as true as possible to its source? Or is the darkness/lightness of the font more important?
- If darkness only is a consideration, think about changing to a darker font.
- How much effort do you want to put into the modification?

A while back, someone came by with the problem that the Computer Modern Roman fonts were falling apart on his 300 dpi write-white printer. Thin parts of the characters were so thin as to make unwanted breaks in the letters. Parts of characters where arches joined stems, as in the lowercase "n" and "u", disappeared. The overall page was quite light and a photocopy of it gave a result that was very difficult to read. As the pages of output were often to be photocopied, this was an important consideration.

We tried the following in order to improve the fonts:

(1) Some higher settings of "*blacker*" and changes to "*fillin*".
(2) Selectively adding "*blacker*" to some values.
(3) Setting a minimum line thickness, 2 pixels in this case.

The Computer Modern fonts use an amount called "*blacker*", which adds to or subtracts from certain line and pen thicknesses. Increasing the amount of "*blacker*" has the effect of adding an absolute pixel amount to certain values, e.g., *blacker* := .75 would always add .75 pixels to the stem, no matter the resolution. If a line has a value of 4.1 pixels before rounding, adding .75 would make that value 4.85 and round to 5 pixels. But if "*blacker*" were .25, then this line value would be 4.35 and round to 4; in this case "*blacker*" has no apparent effect on this value. Setting the value of blacker to 0 means that no adjustment is made. The value of "*blacker*" is given in the "*mode*" setting.

"*Fillin*" is used to compensate for extra heaviness that seems to appear in diagonal lines. A positive value means that the diagonal line will be thinner, a negative value would add thickness to that line.

Both (1) and (2) often created unwanted character distortions since the "tfm" widths wanted to be the same. When each line in a character was increased by one, the shapes were often distorted; counter (inside) shapes seemed to suffer the most. A first attempt with (3) showed promise, and though the result was not as dark as with the wb machine, the feeling of Computer Modern was retained.

In the first attempt, we did not manage to change all of the appropriate thicknesses to 2 pixels. Don Knuth then made the correction shown in figure 3, which keeps minimum thicknesses to 2 pixels. Looking at the pixel patterns, this fix appears to help quite a bit.

```
1) CMBASE.MF without the modifications
2) CMBASE.MF with the two modifications

1) min_Vround:=max(fine.breadth,crisp.breadth,tiny.breadth);
2) min_Vround:=max(fine.breadth,crisp.breadth,tiny.breadth,2);%"WRITE WHITE"

    >> The addition of the value 2

1) enddef;
2) forsuffixes $=thin_join,hair,curve,flare,dot_size,cap_hair,cap_curve,
2) vair,bar,slab,cap_bar,cap_band,stem',cap_stem',vair',fudged.hair,
2) fudged.stem,fudged.cap_stem: $:=max($,2); endfor  % "WRITE WHITE" ONLY!
2) enddef;

    >> The addition of these three lines before the "enddef"
```

Figure 3. Additions to *font_setup* in `cmbase.mf`

>> Add the two changes shown in figure 3 to the "*font_setup*" macro located in your `cmbase.mf` file. You may want to keep the old version around until you are sure that you have put in the fix. And it is a good idea to make a note of the changes made and why for future reference. Then run a test, using your original *mode_setup* specifications. If the test result looks unchanged, you may not actually have the fix in. It is useful to get a pixel representation to verify this.

>> To go about looking for the "right" settings, add the minimum 2-pixel fix; this number may want to be larger with a higher resolution printer. Then run some systematic tests with new "*mode*" definitions, typically changing the amount of "*blacker*" and the amount of "*fillin*" until you find an appropriate setting. It may be different for each printer. These modifications maintain "tfm" dimensions in the Computer Modern fonts. <<

"*Mode*" information tells a few specifics about each printer and you can make a new mode to suit your printer. The existing modes typically have information about the resolution, an "*o_correction*" for the amount a curve extends past a vertical limit, a value for "*blacker*" and a value for "*fillin*". There is also information about what will show up on the screen when running the METAFONT and such, but we'll only concern ourselves with the resolution, "*blacker*", and "*fillin*".

A typical mode definition, as found in the `waits.mf` file, looks like this:

```
% imagen mode: for the Imagen 8/300 (Canon engine)
mode_def imagen =
 proofing:=0; % no, we're not making proofs
 fontmaking:=1; % yes, we are making a font
 tracingtitles:=0; % don't show titles in the log
 pixels_per_inch:=300;
 blacker:=0; % Canon engine is black enough
 fillin:=.2; % and it tends to fill in diagonals
 o_correction:=.6; %
 enddef;
```

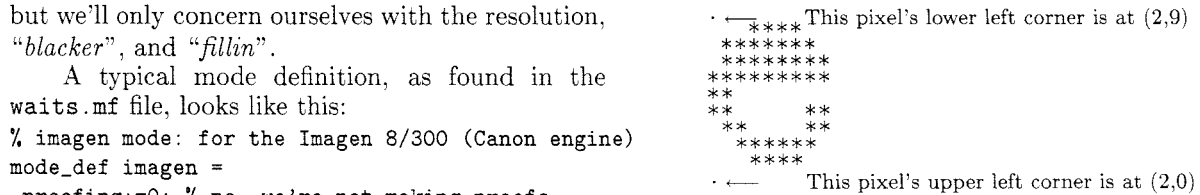
The Computer Modern fonts weren't tested on all possible print devices; you can imagine why. Some of the modes listed have conjectural values, as it wasn't possible to test extensively on these printers. When conjectural values are given, or a new machine is to be added, trying the following test method to help establish suitable settings for your printer.

(1) Check to see if a mode already exists with the same print engine or same or similar print characteristics; if so, use that mode.
(2) Look for a mode which has the same resolution as your printer. If no such mode exists, then make a mode following an existing pattern, changing the resolution to fit your printer.
(3) Set the value of "*blacker*" to 0.
(4) Set the value of "*fillin*" to 0.

Run a test with these settings; this will be your control. Then vary values of "*blacker*" and "*fillin*" systematically. For example, run 9 samples, setting "*blacker*" to 0, .5, and 1 while setting the values of "*fillin*" to $-.5$, 0, and .5. You will begin to see areas that are more successful than others. Look at the values in those ranges more closely. The next set may look like this: "*blacker*" $= 0, .2, .4$ and "*fillin*" $= -.5, 0$. Try this on a variety of the fonts you want to use on your printer to find an appropriate setting.

There are caveats to note though. The shapes may suffer, especially in the case of smaller or lower resolution fonts. In the case of a small lowercase "e", if the distance between the top of the curve of the "eye" and the bottom of the bar is 4 pixels or less, the "eye" will fill in and you will get a dark spot. Sometimes shapes produced with a single-pixel pen look better than those produced with the "minimum thickness" pen.

Here is "e" from CMR5 at a resolution of 300 dpi. Pixel locations are indicated in METAFONT coordinates.



(This example was sent by J. Sauter.)

There always was a question as to how well these fonts would work at low resolution, where often special compensations are made. In such low resolution cases, where the characters don't work well, it may be more economical to edit the character shapes directly. With some to a significant amount of extra work, you can also alter the source code to modify the fonts.

The previous suggestions are based on experiments that have already shown a degree of success within the limitations of the medium. I encourage everyone to experiment with these methods, to contemplate other possible solutions to this problem and to share their results.

METAFONT mode_def Settings for Various TEX Output Devices

Barbara Beeton

As Neenie Billawala explained in the preceding article, the marking characteristics of different print engines must be taken into account in order to assure legible, attractive output. For the Computer Modern family, this is done by tuning several parameters built into the METAFONT design. The settings for all printers used at Stanford appear in the file WAITS.MF. Other settings are frequently requested and (less frequently) communicated in TEXhax or Laser-Lovers.

Here is a typical **mode_def** setting, adapted from PLAIN.MF (*The* METAFONT*book*, page 270) for 200 dpi devices (such as the Xerox XGP, the original TEX output device); it has been augmented by the parameter *aspect_ratio* (required for non-square rasters; the default value is given).

```
mode_def lowres =
 proofing:=0;        % not making proofs
 fontmaking:=1;      % we are making a font
 tracingtitles:=0;   % don't show titles
 pixels_per_inch:=200;
 blacker:=.65;    % make pens a bit blacker
 fillin:=.2;   % adjust for diagonal fillin
 o_correction:=.4;   % less overshoot
 aspect_ratio:=1/1;  % vertical/horizontal
 enddef;
```

For all font "production", typical settings are *proofing* = 0 and *fontmaking* = 1. *tracingtitles* is usually set to 0 for low-resolution fonts (400 dpi or less) and to 1 for higher-resolution fonts, to reassure one that the computer is still in operation and to indicate how far it has progressed during a long job. The standard proof settings can be found in PLAIN.MF as already noted.

For more guidance, see *Adapting to local conditions*, *The* METAFONT*book*, page 278.

The table at the bottom of the page contains a summary of the relevant settings gleaned from available sources. The print engines cited in the table are listed below, along with an indication of whether they are write-black (wb) or write-white (ww), if known, and the names of some of the output devices into which they have been built.

Canon CX (wb)	Apple LaserWriter, Cordata, HP LaserJet, Imagen 8/300, QMS and Talaris 8 ppm printers
Canon LBP-10	Imagen 10/240
Canon (wb)	Imagen 3320, Imagen 7320
Ricoh 4080	DEC LN03; TI OmniLaser 2115
Ricoh LP4120	HP 2688A, Imagen 12/300
Xerox XP-12 (ww)	DEC LN01, QMS 1200, Talaris 1200, Xerox 2700
Xerox XP-24 (ww)	Imagen 24/300, QMS 2400, Talaris 2400, Xerox 3700

Additions and corrections to this list are solicited, as are suggestions for other subjects on which such an overview might be useful.

Typical **mode_def** parameter settings

		pixels_per_inch	*blacker*	*fillin*	*o_correction*	*aspect_ratio*
PLAIN.MF						
proof		2601.72	0	0	1	
lowres		200	.65	.2	.4	
WAITS.MF and other sources						
dover	(Xerox Dover)	384	1.2	0	.6	
imagen	(Canon CX)	300	0	.2	.6	
qms	(Xerox XP-12E)	300	.75*	0*	.5*	
qms[1]		300	.8	.2	.4	
decln[2]	(Ricoh 4080)	300	.9	.2	.5	
aps	(APS-Micro5)	722.909	.2	.2	1	
crs	(Alphatype CRS)	4000+4000/3	.4	0	1	
boise	(HP 2680A)	180	.55	.1	.3	
DD	(DataDisc terminal)	70	0	0	.2	
canon	(Canon LBP-10)	240	.2	.2	.4	
newDD	(DataDisc terminal)	70	0	0	.2	4/3
cg	(Compugraphic 8600)	1301.5	.2	.2	1	1569/1301.5
epson		240	0	0	.2	9/10

*A note in WAITS.MF states that these settings are conjectural.

All settings are from PLAIN.MF or WAITS.MF except for:

[1]Charles Karney, appeared in TEXhax 1986, issue 4

[2]Charles LaBrec, appeared in TEXhax 1986, issue 6

Typesetting on Personal Computers

A New TEX-based Book Typesetting Package for the Macintosh

From a news release datelined San Francisco, January 8, 1987, we have learned of the availability of *Page One* automated book publishing software from McCutcheon Graphics of Toronto. This software is based on FTL's MacTEX package, and uses templates to access a variety of formats, with a choice of five of the most popular book trim sizes, and a total of 50 page layouts. It requires a Macintosh Plus and hard disk.

The manuscript is created on a Macintosh using Microsoft Word formats. The publisher chooses the desired design template from a catalog and enters its number on a simple screen form, along with optional page format information. The book is then printed on any PostScript-compatible printer.

Page One was co-developed by Thad McIlroy, formerly an author, publisher, bookseller, and editor of over 40 titles, and one of the world's first book authors to use desktop publishing; noted book designer Garfield Reeves-Stevens, author, editor, and designer of over 300 books for McGraw-Hill, Doubleday, and other publishers; and Toronto-based FTL systems.

These claims are made:

- Total design and production time: a few hours.
- Skills required: ability to use a mouse.

Additional information about this package can be obtained from

McCutcheon Graphics Inc.
130 Bridgeland Avenue
Toronto, Ontario Canada M6A 1Z4
Wes Thomas: 516-266-1652
Thad McIlroy: 416-789-2993

Editor's note: TEX is now appearing with increasing frequency as a "back-end" composition system, shielded from the originator of the manuscript. Not all creators of such packages acknowledge the TEX connection, but those that do seem to feel that the TEX association will be a good advertisement of quality. We applaud their confidence.

Output Devices

TEX Output Devices

Some of the interfaces listed in these charts are considered proprietary. Most are not on the standard distribution tapes; if it is known that an interface has been submitted to the distributor, this fact is noted. To obtain information regarding an interface, if it is supposed to be included in a standard distribution, first try the appropriate site coordinator or distributor; otherwise, request information directly from the sites listed.

The codes used in the charts are interpreted below, with a person's name given for a site when that information could be obtained and verified. If a contact's name appears in the current TUG membership list, only a phone number or network address is given. If the contact is not a current TUG member, the full address, and its source, are shown.

Corrections and new information are welcome; send them to Barbara Beeton (address on page 3).

Sources

ACC Advanced Computer Communications, Diane Cast, 720 Santa Barbara Street, Santa Barbara, CA 93101, 805-963-9431 (DECUS, May '85)

Adelaide Adelaide University, Australia, Andrew Trevorrow, (08) 267 1060
(Note that Andrew Trevorrow is no longer at Adelaide university, and cannot handle requests for software; all programs cited have been sent to the appropriate coordinators for inclusion on the distribution tapes.

AMS American Mathematical Society, Barbara Beeton, 401-272-9500

Arbor ArborText Inc, Bruce Baker, 313-996-3566

A-W Addison-Wesley, Brian Skidmore, 617-944-3700, ext. 2253

Bochum Ruhr Universität Bochum, Norbert Schwarz, 49 234 700-4014

Caltech Cal Tech, Glen Gribble, 818-356-6988

Canon Canon Tokyo, Masaaki Nagashima, (03)758-2111

Columbia Columbia University, Frank da Cruz, 212-280-5126

CMU Carnegie-Mellon University, Howard Gayle, 412-578-3042

COS COS Information, Gilbert Gingras, 514-738-2191

Carleton Carleton University, Neil Holtz, 613-231-7145

DEC Digital Equipment Corporation, John Sauter, 603-881-2301

GA Tech GA Technologies, Phil Andrews, 619-455-4583

GMD Gesellschaft für Mathematik und Datenverarbeitung, Dr. Wolfgang Appelt, D-5202 Sankt Augustin, Federal Republic of Germany

HP Hewlett-Packard, Stuart Beatty, 303-226-3800, ext. 2067

IAM Institut für Angewandte Math, Univ of Bonn, Federal Republic of Germany, Bernd Schulze, 0228-733427

Imagen Imagen Corp, Dan Curtis, 408-986-9400

INFN INFN/CNAF, Bologna, Italy, Maria Luisa Luvisetto, 051-498286

Intergraph Intergraph, Mike Cunningham, 205-772-2000

JDJW JDJ Wordware, John D. Johnson, 415-965-3245

K&S Kellerman & Smith, Barry Smith, 503-222-4234

LLL Lawrence Livermore Laboratory

LSU Louisiana State University, Neal Stoltzfus, 504-388-1570

MPAE Max-Planck-Institut für Aeronomie, H. Kopka, (49) 5556-41451

MR Math Reviews, Patrick Ion, 313-996-5273

NJIT New Jersey Institute of Technology, Bill Cheswick, 201-596-2900

OCLC OCLC, Tom Hickey, 616-764-6075

OSU1 Ohio State University, John Crawford, 614-422-1741

OSU2 Ohio State University, John Gourlay, 614-422-6653

Personal Personal TEX, Inc., Lance Carnes, 415-388-8853

Procyon Procyon Informatics, Dublin, Ireland, John Roden, 353-1-791323

SARA Stichting Acad Rechenzentrum Amsterdam, Han Noot, Stichting Math Centrum, Tweede Boerhaavestraat 49, 1091 AL Amsterdam (TUGboat 5, no. 1)

Scan Laser Scan Laser, England, John Escott, +1 638 0536

Sci Ap Science Applications, L. E. Fields, 619-458-2616

SLAC Stanford Linear Accelerator Center, 415-854-3300

SRI SRI International

Stanford Stanford University

SUN Sun, Inc

Talaris Talaris, Sonny Burkett, 619-587-0787

T A&M1 Texas A&M, Bart Childs, 409-845-5470

T A&M2 Texas A&M, Ken Marsh, 409-845-4940

T A&M3 Texas A&M, Norman Naugle, 409-845-3104

Tools Tools GmbH Bonn, Edgar Fuß, Kaiserstraße 48, 5300 Bonn, Federal Republic Germany (TUGboat 8#1, page 46)

UBC University of British Columbia, Afton Cayford, 604-228-3045

UCB University of California, Berkeley, Michael Harrison, `vortex@berkeley.arpa`

UCIrv1 University of California, Irvine, David Benjamin

UCIrv2 University of California, Irvine, Tim Morgan

U Del University of Delaware, Daniel Grim, 302-451-1990

U Köln Univ of Köln, Federal Republic of Germany, Jochen Roderburg, 0221-/478-5372

U Mass University of Massachusetts, Amherst, Gary Wallace, 413-545-4296

U Md University of Maryland, Chris Torek, 301-454-7690

U Mich University of Michigan, Kari Gluski, 313-763-6069

U Milan1 Università Degli Studi Milan, Italy, Dario Lucarella, 02/23.62.441 (329)

U Milan2 Università Degli Studi Milan, Italy, Giovanni Canzii, 02/23.52.93

U Shef University of Sheffield, England, Ewart North, (0742)-78555, ext. 4307

U Utah University of Utah, Nelson H. F. Beebe, 801-581-5254

U Wash1 University of Washington, Pierre MacKay, 206-543-2386

U Wash2 University of Washington, Jim Fox, 206-543-4320 (NOS 2.2)

U Wisc University of Wisconsin, William Kelly, 608-262-9501

UNI.C Aarhus University, Regional Computer Center

Vanderbilt Vanderbilt University, H. Denson Burnum, 615-322-2357

Wash St Washington State University, Dean Guenther, 509-335-0411

Weizmann Weizmann Institute, Rehovot, Israel, Malka Cymbalista, 08-482443

Notes for tables

[1] graphics supported
[2] included on one of the standard distribution tapes

Low-Resolution Printers on Multi-User Systems — Laser Xerographic, Electro-Erosion Printers

	Amdahl (MTS)	CDC Cyber	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM	Prime	Siemens BS2000	VAX UNIX	VAX VMS
Agfa P400								IAM				
Apple LaserWriter					Utah	Adelaide[2] Arbor[1] Utah					Arbor[1] Carleton[2] Utah	Utah
Canon					Utah	Utah					Canon Utah	Utah
DEC LN01											U Wash1[2]	NJIT
DEC LN03												K&S Procyon DEC[2]
HP LaserJet					Utah	Utah					Utah	Utah
IBM 38xx, 4250, Sherpa								SLAC[2]				
Imagen	Arbor UBC		T A&M1	Stanford Vanderbilt	Columbia SRI Utah	Utah	Arbor	SLAC[2]			U Md[2] Utah	K&S[1] Utah
QMS Lasergrafix	Arbor	U Wash2[1]	T A&M1			T A&M2	GMD	Arbor	T A&M3	GMD	Arbor U Wash1[2]	Arbor GA Tech T A&M3
Symbolics					U Wash1						U Wash1[2]	U Mass
Talaris				Talaris[1]	Talaris[1]		Talaris[1]	Wash St			Talaris[1]	Talaris[1]
Xerox Dover					CMU						Stanford	
Xerox 2700II		Bochum			OSU2						OSU2	
Xerox 9700	Arbor U Mich			U Del			Arbor	Arbor U Del			U Del	ACC Arbor

Low-Resolution Printers on Multi-User Systems — Impact and Electrostatic Printers

	Amdahl (MTS)	CDC Cyber	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM	Prime	Siemens BS2000	VAX UNIX	VAX VMS
Apple ImageWriter					Utah	Utah					Utah	Utah
C Itoh												LSU
DEC LP100					OSU2							
Facit 4542												INFN[2]
Florida Data					MR							
NDK 7700								IAM				
Okidata					Utah	Utah					Utah	Utah
Printronix			T A&M1		Utah	Utah					Utah	Utah
Toshiba			T A&M1[1]		Utah	Utah					Utah	Procyon Utah
Varian												Sci Ap
Versatec		U Köln	T A&M1[1]	GA Tech Vanderbilt	U Wash1		GMD U Milan2	Weizmann	LLL		U Wash1[2]	K&S

Low-Resolution Printers on Microcomputers and Workstations

	Apollo	Apple Macintosh	Atari ST	Cadmus 9200	HP1000	HP3000	HP9000 200	IBM PC	ICL Perq	Integrated Solutions	SUN	Texas Instr PC
Apple ImageWriter		A-W[1] K&S[1]						MR Utah		Utah	Utah	
Apple LaserWriter	Arbor[1]	A-W[1] K&S[1]					Arbor	A-W[1] Pers[1] Arbor[1] Utah		Utah	Arbor[1] Utah	
Canon								Utah	GMD	Utah	Utah	
Cordata LP300								Personal				
Diablo Qume						Personal						
Epson			Tools		JDJW	U Shef		A-W, Pers U Milan1 U Shef				T A&M1
Fujitsu				U Köln								
GE 3000	COS											
HP 2680					JDJW	Personal						
HP 2688A					JDJW		Caltech HP					
HP Laserjet Plus			Tools				MPAE	Arbor[1] Utah		Utah	Utah	
Imagen	Arbor[1] OCLC							A-W[1] Pers[1] Arbor[1] OCLC, Utah		Utah	Arbor[1] Sun Utah	
Printronix								Utah		Utah	Utah	T A&M1[1]
QMS Lasergrafix	Arbor Scan Laser							A-W[1] Personal[1] Arbor[1]			Arbor U Del	
Texas Instruments 855												T A&M1[1]
Toshiba								A-W Personal Utah		Utah	Utah	
Xerox 9700	COS Scan Laser											

Typesetters

	Amdahl (MTS)	Apollo	CDC Cyber	DEC-20	HP3000	HP9000 200; 500	IBM MVS	IBM PC	IBM VM	Sperry 1100	SUN	VAX UNIX	VAX VMS
Allied Linotype CRTronic													Procyon
Allied Linotype L100, L300P	Arbor	Arbor				Arbor		A-W Arbor Personal			Arbor	Arbor	Arbor
Allied Linotype L202								Personal					Procyon
Alphatype CRS				AMS									
Autologic APS-5, Micro-5	Arbor	Arbor COS Scan Laser		Arbor	Arbor			Arbor Personal	Arbor		Arbor	Arbor	Arbor Intergraph
Compugraphic 8400					U Shef			Personal					K&S
Compugraphic 8600			UNI.C				Wash St	Personal	Wash St	U Wisc			K&S
Harris 7500												SARA	
Hell Digiset							GMD*						

* Digiset also supported at GMD on Siemens BS2000.

Video Displays

	Apollo	Apple Macintosh	Atari ST	Cadmus 9200	Data General MV	IBM MVS	IBM PC	Integrated Solutions	ICL Perq	Siemens BS2000	SUN	Texas Instr PC	VAX VMS
AED 483, 512													Adelaide*[2]
ANSI-compatible terminals													Adelaide*[2]
BBN BitGraph								Utah			Utah		Utah
DEC ReGIS													Adelaide*[2]
DEC VT100													Adelaide*[2]
DEC VT125													INFN[2]
Talaris 7800													Talaris[1]
Tektronix 4014						U Milan1							Adelaide*[2] INFN[2]
Visual 500, 550													Adelaide*[2]
VAXstation													Arbor
other screen preview	Arbor	K&S[1] A-W[1]	Tools	U Köln	T A&M1[1]	GMD	Arbor A-W Personal	UClrv1	GMD	GMD	Arbor UCB UClrv2[2]	T A&M1[1]	Adelaide*[2]

* The DVItoVDU program developed at Adelaide University has also been ported to VAX/UNIX, and has been submitted to the UNIX distribution.

Public Domain TEX DVI Driver Family

Nelson H. F. Beebe
University of Utah

At the 1985 summer TEX Users Group meeting at Tufts University, I announced the availability of a family of TEX DVI drivers. This announcement updates the information presented at the meeting.

These drivers are **public domain** and **not** copyrighted. If you get them and commercialize them, remember that I will be in competition giving them away for free.

Enhancements and ports to new systems are expected to be returned to the author for further redistribution.

This driver family is written in a highly portable subset of C, with preprocessor conditionals used to select code fragments which are of necessity dependent on the host machine, compiler, or operating system. The code supports a *family* of devices, so that each driver presents a Unix-style command-language interface to the user, and with the exception of `\special{}` support, produces identical output subject only to variations in device resolution.

Font files in `.PXL`, `.GF`, and `.PK` formats are supported, including invisible fonts (which have broken other DVI driver programs). Default font format lists and directory search paths are established at compile time, but can be overridden at run time on all systems. A font substitution mechanism is also supported to handle the common case of fonts being unavailable in a particular magnification or style.

A new feature added in December '86 is run-time selectable virtual font caching, which maps entire font files into the address space at file open time. This should improve response when font files reside on a networked file system; the many small character packet requests otherwise entail substantial real-time delay across the net.

Environments

This family is running in these environments:

Machine	Operating System	Compiler
DEC-20	TOPS-20	PCC-20
IBM-PC	PC-DOS	Microsoft V3,V4
VAX	VMS 4.x	VMS C
Unix-Box	Unix	CC

Unix systems on which this family runs include Sun, Integrated Solutions, 4.xBSD VAX, Gould, and Hewlett-Packard. On the IBM PC, compilers from Lattice, Wizard, and Aztec have also been tried, but have regrettably proved too buggy.

Devices supported

Screen displays:

`DVIBIT` Version 3.10 BBN BitGraph terminal

300dpi laser printers:

`DVIALW` PostScript (Apple LaserWriter)
`DVICAN` Canon LBP-A2
`DVIIMP` Imagen imPRESS-language family
`DVIJEP` Hewlett-Packard Laser Jet Plus

Dot-matrix and other lower-resolution printers:

`DVIJET` Hewlett-Packard Laser Jet 75 dpi
`DVIM72` Apple ImageWriter 72 dpi
`DVIMAC` Apple ImageWriter 144 dpi
`DVIMPI` MPI Sprinter 72 dpi
`DVIO72` OKIDATA Pacemark 2410 72 dpi
`DVIOKI` OKIDATA Pacemark 2410 144 dpi
`DVIPRX` Printronix 60h x 72 dpi
`DVITOS` Toshiba P-1351 180 dpi

Documentation

- 75-page manual in LATEX (primarily for installers)
- 15-page user documentation in Unix man style (but set by LATEX), TOPS-20 INFO, and GNU EMACS TEXinfo formats.

Distribution

Available on IBM PC-DOS floppy disks (about 6), or 1600 bpi 9in mag tape in TOPS-10/20 BACKUP/DUMPER format, VAX VMS BACKUP format, Unix tar format, and ANSI D-format from

Dr. Nelson H. F. Beebe
Center for Scientific Computing
220 South Physics
University of Utah
Salt Lake City, UT 84112, U.S.A.
(801) 581-5254
EMAIL: `Beebe@Utah-Science.Arpa`
`Beebe@Utah-CS.arpa`
`BeebeN@Utah-RUAC.Arpa`

Send tape or floppies for a copy. Financial contributions are always welcome, but not required. The family is also included on my `<PLOT79>` graphics distribution, since all new documentation for that uses LATEX.

This distribution is available for Internet anonymous FTP from `Utah-Science.Arpa`. Login and retrieve the file `PS:<ANONYMOUS>OOREADME.TXT`; it

contains an introduction to the system and pointers to interesting directories.

To facilitate periodic updating, FTP command files and directory listings in both alphabetic and reverse chronological order are maintained in each distribution directory. TOPS-20 sites can use the FTP `UPDATE` command instead to get changes with even less effort.

For the benefit of Unix sites, compressed `tar` files are also maintained.

Source plus documentation amounts to about 1.5Mb, and executable code for each device amounts to 80Kb–150Kb, depending on device and host machine; this is probably too large for EMAIL to Bitnet and Usenet sites.

Future work

Major pieces of work remaining to be done:

(a) merging in support of PostScript resident fonts;
(b) addition of more `\special{}` support to the laser printer drivers; currently only `DVIALW` supports a `\special{}` command;
(c) addition of support for windowing systems on bitmapped workstation displays; X-windows support is in progress locally.

Volunteer contributions are most welcome!

DVItoVDU 1.7 and PSPRINT 1.1

Andrew Trevorrow
University of Adelaide Computing Services

Introduction

DVItoVDU is an interactive page previewer that drives a variety of commonly available terminals. PSPRINT is a PostScript driver for the Apple LaserWriter that can print a DVI file, a raw PostScript program, or an ordinary text file. Both programs run under VAX/VMS and are in the public domain. Copies of the software, including source code and documentation, have been sent to Stanford and Kellerman & Smith for inclusion on their VAX/VMS distribution tapes.

Now that TEX under VAX/VMS seems to have reached its final state, further development of DVItoVDU and PSPRINT is most unlikely. (Another good reason is that I'm leaving my position at the University of Adelaide. If anybody out there needs the services of an experienced TEXnician, I can be reached at my University address or via electronic mail; my ACSNET address is `akt@uacomsci.ua.oz`. I'm particularly interested in working overseas on a short-term basis.)

DVItoVDU 1.7

Version 1.5 was described in detail in TUGboat vol. 7, no. 1. Here is a list of the significant changes that have occurred since that release:

- All font-dependent code has been moved out of the main module and into `FontReader`. This module replaces `PXLReader` and can read PXL or PK files. The `/font_directory` and `/dummy_font` values determine which format to use, as well as the location and naming convention of all font files. (I was going to handle GF files as well, but finally decided that any site using such files would eventually switch to the PK format. PK files use about a fifth of the disk space required by PXL files, and interpreting the compressed information only adds about 5% to the processing time. See Tomas Rokicki's article in TUGboat vol. 6, no. 3.)
- The `/font_directory` value is restricted to a single directory rather than a list of directories. The `/dummy_font` file must represent an unmagnified font.
- The main module's Full display routines have been changed to handle a new method of loading character bitmaps into dynamic memory on a demand basis. DVItoVDU is now slightly faster, but tends to use more memory.
- DVItoVDU can now handle all TEX character codes from 0 to 255.
- There are two new `/vdu` values. `VT220` is a new implementation by John Mann at Monash University that uses downloaded chunky graphics to get a window resolution of 132 by 100. `VIS240` is for a ReGIS VDU with a larger window region.
- The page position and first few bytes of any `\special` commands are now displayed. (Note that PSPRINT allows the use of `\special` to include a file of PostScript commands.)

PSPRINT 1.1

"PS" stands for PostScript, the printer language used by the Apple LaserWriter. The decision to replace our Imagen IMPRINT-10 with a LaserWriter was motivated by a desire to support PostScript. Upgrading to a new, high-performance PostScript printer should require only trivial editing changes to a few text files used by the PSPRINT system.

Like DVItoVDU, PSPRINT comes with a comprehensive User Guide explaining how to run the program, and a System Guide containing installation details. The following material has been extracted from these two documents. I'll start with a brief system description.

System Overview

PSPRINT is *not* a VMS print symbiont; it is actually made up of a number of components. The following sequence of events occurs when a user types '`psprint file`':

1. `psprint.exe` is invoked. This small Modula-2 program extracts the file parameter and qualifier values from the command line according to `psprint.cld`. If everything seems okay then `tex_ps:psprint.com` is activated. (`tex_ps` is the logical name for a public-access directory containing all the necessary files that make up the PSPRINT system.)
2. `psprint.com` is a command file that does most of the messy system-dependent work. It can handle three file types: a DVI file, a PostScript program, or an ordinary text file.
3. Given a DVI file, `psprint.com` uses `psdvi.exe` to create a temporary PostScript file. (PSDVI is another Modula-2 program, much of its code borrowed from DVItoVDU. It too can read PXL or PK files.) Typical documents take about 2 CPU seconds per page to translate on a VAX-11/785.
4. Each type of file is bracketed by appropriate header and trailer files before being sent as a single job to a queue called "PS". The standard VMS print symbiont then sends each job to the LaserWriter. Temporary files created in the user's current directory are automatically deleted after printing.

Much of the behaviour of PSPRINT can be changed by simply editing `psprint.com` and/or the various text files used by this command file. Such flexibility is of particular benefit to those unfortunate sites without Modula-2.

The remaining sections have been extracted from the User Guide available to people at our site.

TEX Output

PSPRINT looks for a DVI file by default. To print `foo.dvi`, simply type:

```
$ psprint foo
```

PSPRINT will translate `foo.dvi` into a temporary PostScript file called `foo.tmp`. The DVI and TEX page numbers are displayed as each page is translated. If no errors are found, the temporary file is automatically sent to the PS queue where it is printed (and then deleted).

PSPRINT translates pages in reverse order so that your document is collated automatically. Various qualifiers allow you to select particular pages for printing, or produce landscaped output, etc. See below for details.

PostScript Output

PostScript is a complete programming language with a powerful set of operators for manipulating text and graphics. A PostScript program is an ordinary text file. If you give it a name like `foo.ps` then sending it to the LaserWriter is simple:

```
$ psprint foo.ps
```

If you decide to use some other file extension you'll have to add the `/ps` qualifier.

People who want to write PostScript programs will need to get the *PostScript Language Reference Manual* by Adobe Systems Incorporated (published by Addison-Wesley).

Ordinary Text Output

To print a normal text file, just use any file extension except `.dvi` or `.ps`:

```
$ psprint foo.lis
```

The output is *not* collated. By default you get 66 lines per page, and any lines more than 80 characters long are truncated. The `/landscape` qualifier will rotate the paper so that you can have lines up to 132 characters long (but only 50 lines per page).

Type '`psprint foo.ps /text`' if you want to print a PostScript file rather than have it interpreted. FORTRAN carriage-control files and RUNOFF output can also be printed, but overprinting is not handled correctly.

PSPRINT Qualifiers

The following three qualifiers override PSPRINT's assumptions about the format of the input file based on the given file specification:

- `/dvi` — for a DVI file.
- `/ps` — for a PostScript program.
- `/text` — for an ordinary text file.

Some qualifiers can only be used with a DVI file:

- `/pages=first:final` — to select a subrange of pages for printing. `first` and `final` can select either DVI page numbers (positive integers) or TEX page numbers (in the same format as used by DVItoVDU). More than one DVI page may match a particular TEX page specification. If `first` is a TEX page then the *lowest* matching DVI page is selected, and if `final` is a TEX page then the *highest* matching page is selected. Note that `first` and `final` are optional; they default to the first and last DVI pages respectively. Omitting '`:final`' entirely sets `final` equal to `first`.
- `/magnification=n` — to override the DVI magnification. `n` is a positive integer 1000 times the desired magnification and should match one of TEX's `\magstep` values.
- `/units=xx` — to set the units of dimensions displayed in some PSPRINT messages (e.g., if the page is off the paper). `xx` can be one of `in`, `cm`, `mm`, `pt`, `pc`, `bp` or `px`.
- `/stats` — to see statistics on the number of rules/fonts/characters used on each page.

The remaining qualifiers apply to all types of input files:

- `/landscape` — to get landscaped output. If given a DVI file, PSPRINT will check each page to make sure it fits within the rotated paper.
- `/delete` — to delete the input file after printing.
- `/notify` — to be notified when the print job on the PS queue has finished.
- `/copies=n` — to specify up to 100 copies. Use of this facility is not encouraged. Note in particular that copies of a multiple-page document will *not* be collated.
- `/note=string` — to include a note on your banner page.

Error Messages

If PSPRINT detects one or more errors while translating a DVI file then the temporary PostScript file is automatically deleted and nothing is printed; not even a banner page. The sort of errors you are most likely to see include:

- `Page off paper (paper is ...`
  — the current page contains material beyond one or more paper edges. Use DVItoVDU to see where the problem lies.
- `Couldn't open font file: ...`
  — your DVI file uses a font at a non-existent size. Try a different font magnification in your source file.
- `Couldn't open \special file: ...`
  — PSPRINT could not locate the given `\special` file. Check the file's spelling and/or directory location.

Error messages apply to the most recently displayed DVI/TEX page. If your DVI file uses a large number of fonts or a few very large fonts then you might get an error page with the message "`Font_Memory_Exhausted`". Try printing just a few pages at a time or using fewer/smaller fonts. The LaserWriter has a large but finite amount of memory for storing the character bitmaps downloaded by PSPRINT. This limitation is unlikely to worry most people.

An error in a PostScript program should produce some sort of message on your printed output. If this happens a lot, go back and read the *PostScript Reference Manual* more carefully.

Merging TEX and PostScript

Let's assume you have a file called `fig.ps` that contains suitable PostScript commands for generating some sort of figure. To include this figure in a TEX document, just type '`\special{fig.ps}`' after leaving sufficient space:

```
\midinsert
   \vskip 3in
   \special{fig.ps}
\endinsert
```

PSPRINT interprets the `\special` argument as a file name and includes this file in the information sent to the LaserWriter.

It may take a little practice to position the figure correctly. When you print a raw PostScript file (e.g., '`psprint fig.ps`') the default origin is at the bottom left corner of the paper. However, when you include this file in a TEX document the origin is automatically moved to the position of the

`\special` command. You can move the figure about by shifting the `\special` position, or by including a `translate` command in the PostScript file. The latter option is faster because you don't have to run TEX again. For example, '`72 36 translate`' at the start of `fig.ps` would move the figure 1 in right and 0.5 in up (PostScript's default units are "big points" where 72 bp = 1 in).

Alternatively, you may prefer to place your figure at an absolute position on the paper (i.e., at the same position produced by '`psprint fig.ps`'). Simply include an `initgraphics` command at the start of `fig.ps`. The position of the `\special` command then becomes irrelevant; just make sure it's on the right page!

The hardest part of the scheme is creating the PostScript file in the first place. You are faced with two possibilities:

1. Learn enough PostScript to be able to draw your own figures.
2. Use a program to convert an existing graphics file into a corresponding PostScript file. This assumes you are able to produce graphics of some sort (e.g., a Zeta plot file) and have access to a suitable conversion program.

PSPRINT also allows arbitrary PostScript text to appear after the file name in a `\special` command. For example, '`\special{fig.ps 2 1 scale}`' could be used to double the width of the figure in `fig.ps`. At least one space must be used to terminate the file name; further characters are included as a new line at the *start* of the given file. There is a limit of about 200 characters. Note that it is possible to include the same PostScript file more than once, but with a different starting line each time.

Benson-Varian 9211 Looking for a Home

Until it was disconnected earlier this year, a Benson-Varian model 9211 electrostatic printer was used by the American Mathematical Society as a TEX output device. This machine is offered free to anyone who will pay the shipping costs from Providence to its new location.

This machine has a resolution of 200 dots per inch and uses liquid toner. It uses coated, roll-fed paper, 11″ wide. Physical dimensions are 24.25″ w, 27.25″ d, 40.25″ h; weight, 325 lb. The machine was obtained new in Summer 1979 and has been fully maintained under a service contract with the manufacturer.

It was connected to a DECsystem-20 via a Monolithic MSC 8004 microprocessor, which is also included in this offer. For details, contact

George M. Ogilvie
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500, ext. 279

Site Reports

Notes from TEXhax

Malcolm Brown
TEXhax Moderator

Business is booming for TEXhax. The month of January alone saw six issues distributed. It's great to have such participation, and my thanks to everyone who's been contributing (especially to Barbara Beeton for preparing an index for the 1986 issues!).

A consequence of this is that most of my time is spent maintaining the distribution list and preparing the digests. This means that I have very little time for individual responses; indeed, most of the time I cannot respond. This doesn't mean I'm ignoring anyone, I just don't have enough time.

Anyone is welcome to submit any kind of TEX-related inquiry or insight to TEXhax. Inquiries should, however, be intended for the list generally and not addressed to the moderator personally. Questions that relate to TEX distribution should not go to TEXhax, but rather to TUG.

An informal poll was conducted last December regarding the format of TEXhax and the majority of responses favored retaining the digest format. Each digest is now formatted to allow automatic parsing. I'm trying to keep each issue reasonable in size. Currently 10K of material is "critical mass" for a TEXhax issue.

Finally, a reminder: all correspondence regarding changes to the distribution list should be addressed to `texhax-request@score.stanford.edu`. Submissions to TEXhax should be mailed to `texhax@score.stanford.edu`. This directs address changes and submissions to different files and makes things a bit easier to manage.

Editor's note: Back issues of TEXhax are available online at Score (a TOPS-20 system), and can be retrieved by anonymous FTP. The present naming convention is `<TEX>texhax`*n*`.87`, the *n* being the issue number. For 1986 issues, naming is more complicated: `texhax`*n*`.86`*mmdd*, with *n* again being the issue number (always 2 digits) and the extension, the date the issue was distributed.

Malcolm hopes to keep each issue online for at least a year, but that will depend on the volume of material.

We offer our heartiest encouragement and thanks to Malcolm for his diligent attention. It's really great to have TEXhax back on the air.

TEX&Co. on the ST, Part 2

Edgar Fuß
tools GmbH Bonn

When we decided to buy a couple of ST's in the beginning of 1986, our intention was to use them as some sort of super-intelligent terminal for our UNIX machine, i.e. to let the editor run on the terminal itself; the main field of interest of our company is software tools (surprise!) and database systems for UNIX.

As I had already been a TEX addict for several years but lacked a machine to run it on, and as there were at least two implementations for the IBM PC on the market, we said something like "if they can do it on the PC, it must be possible on the ST, too" and I began to think about TEX on the ST in the middle of January, 1986. As there was no suitable Pascal compiler on the ST (C compilers were no better), the process was not too straightforward and involved several different machines (including the ST in intermediate steps as well as the final result). Nevertheless, on April 26, the ST said "`This is TeX, Version 1.5 for the ST`" and — surprisingly enough — the program passed `TRIP` at the first attempt.

I then began to write a DVI driver, and as the task is essentially the same for any pixel-oriented output device like the screen or any matrix printer, we didn't adopt the usual "one printer, one driver"-method but instead wrote *one* driver for screen and *all* printers. This includes what is called 'preview' elsewhere, but as it appears to me to be the most natural thing to see one's TEX output on the screen — at least on a machine like the ST — I just call this a screen driver.

Fortunately this could be done on the ST itself; CCD Pascal turned out to be a useful tool since it allows you to call GEM directly from Pascal, generates reasonably compact code and compiles *fast*. The driver dealt with the screen with variable output size (using all kinds of GEM stuff like pull-down menus, windows, etc.) and some common matrix printers (Prism 132; Epson MX, FX; NEC P2 and compatibles) and used `PXL` fonts

in the first two releases because this was the only form amr fonts were available in.

Having finished the driver (a project requiring a good deal of knowledge of GEM 'logic' and its bugs), we began to deliver what we called Release 0.9 (TEX, version 1.5) in August last year.

The release of TEX, version 2.0, and the cmr inputs then made it possible to use Epson printers with their true resolution of 240 * 216 dots and to generate fonts for high-resolution printers like the NEC P6. We therefore updated TEX and the driver (now including Epson LQ, NEC P5/6/7 and new low-resolution draft modes) and delivered Release 1.0 (TEX, version 2.0) in October, 1986.

Although the primary goal — providing a cheap implementation of TEX for a cheap machine — was now achieved, development continued. Astonishingly enough, there was a considerable demand among our customers for a laser printer driver (you get about 4 to 5 ST's for the price of a cheap laser printer here in Germany), so a LaserJet driver has recently begun producing its first output and adapting this driver to other laser printers seems to be only a matter of time. There were even enquiries for a phototypesetter driver, but this will probably take some more time.

I was quite surprised that our implementation has not only drawn the attention of universities — the customers I originally thought of — but also that of TEX newcomers who see they can do a little bit more with TEX than their favourite word processor was capable of...

Meanwhile, my colleague *Jürgen Keil* implemented METAFONT on the ST and somehow managed to cut down the per-font time from 2 hours to 12 minutes — we'll both have a look at ST-TEX's speed shortly (some people claim it's roughly as fast as on a VAX 780 — I can't fully believe this unless the VAX is heavily loaded, but anyway, I didn't do much about speed up to now).

We, ourselves, are curious to learn how fast we can make it, and the result of our efforts, together with the new driver now under test internally (supporting PK fonts etc.), will probably appear as Release 1.1 in the near future.

Data General Distribution News

Bart Childs

This is the status of the DG distribution:

1. We are running TEX 2.0. We just noticed from TEXhax that 2.1 is out and we will probably upgrade this week.
2. We have figured out how to preload TEX and are distributing all the packages in preloaded form.
3. METAFONT is working fine and is preloaded as well.
4. Our QMS drivers also use preloaded .TFM files.
5. We are in the process of changing all the main codes to use system calls for IO to speed things up. This is effectively using block IO.
6. We now have a driver for the DG–4558 laser printer. This is a fairly pure Canon engine, which causes some problems. It does not like characters whose size exceeds 64 pixels in either direction. We are planning on mixing bitmapping and font downloading to handle reasonable documents. It is written in WEB and I would appreciate a dialog with others who have attacked the same problem.

TEX82 and METAFONT84 Implementation for the HP1000 A-Series

Tor Lillqvist
Technical Research Centre of Finland (VTT)

I have implemented TEX82 and METAFONT84 on the HP1000 A-Series computers running the RTE-A/VC+ operating system. (This is a totally different machine from the older E-Series, on which I think the previous TEX for HP1000 was implemented.) The code runs in CDS (Code/Data Separation) mode, no manual segmentation/overlay was necessary. All data structures are kept in EMA (Extended Memory).

The TEX82 implementation is fairly complete, e.g. it is possible to interrupt TEX while it is running. It passes the TRIP test. A kind of "preloaded TEX" is also implemented, using "shared EMA", the contents of which is saved to a file and restored by a small bootstrap program.

Here are the memory sizes used:

mem_max	32000
mem_min	−32000
buf_size	1000
error_line	79
half_error_line	50
max_print_line	79
stack_size	200
max_in_open	6
font_max	99
font_mem_size	30000
param_size	60
nest_size	40
max_strings	5000
string_vacancies	15000
pool_size	40000
save_size	600
trie_size	8000
dvi_buf_size	1024
file_name_size	64

TEX uses memory quite heavily, the code segment is 271 pages (1 page = 1024 16-bit words), EMA is 256 pages and the data segment is 29 pages, so you probably don't want to run it in a very small configuration.

A small change was necessary to the format of some binary files (`DVI`, `PXL`, `GF`) because of file system restrictions on RTE-A (you cannot know the exact (logical) size of a random-access file, so the first integer in these files contain the file size).

The speed of this TEX on an A900 is comparable e.g. to a VAX-11/750 or a PC AT.

The METAFONT implementation also works OK, but for some reason I am not able to use the same values for *mem_min* and *mem_max* as in TEX (−32000 and 32000). I haven't tried the `TRAP` test, but it processes the Computer Modern fonts without problems.

The current versions are: TEX82 2.0, METAFONT84 1.0, and Computer Modern fonts "5 changes after Version 1.0". The files were obtained from the Unix TEX distribution.

I have also written a `DVI` driver for the HP LaserJet+ printer, using the `DVItype` program as a base. The driver is written for the Pascal/1000 compiler and this TEX implementation, but should be portable to other reasonable compilers and implementations. (Volunteers are welcome!)

I have sent this TEX implementation to Interex (the International Association of HP Users) for redistribution, and Alan Whitney at the MIT Haystack Laboratory has it running. He might be willing to make tapes for Interex members. Requests can naturally also be directed to me; send a 2400 foot tape (or CS80 cartridge) and enough international reply coupons to cover the return postage. (Tapes are written in TF or TAR format.)

My address is:

Tor Lillqvist
VTT/ATK
Lehtisaarentie 2
SF-00340 Helsinki
Finland

You can also reach me through electronic mail at the address `tml@fingate.bitnet` or `mcvax!santra!tml` by phone `+358~0~4566132`, or telex `122972~vttha~sf`.

Prime 50 Series Site Report

John M. Crawford
Ohio State University

Since my last site report, our TEX distribution tape has been altered to reflect the latest from Stanford. Namely, TEX 2.0 and METAFONT 1.0 are currently available, as well much of the other TEX project software.

We still don't have a wide variety of device drivers available. I believe most TEXing Prime sites are using the Texas A&M device drivers which work with the QMS QUIC laser-printers. I hope that by the end of the year, we will see a few more device drivers running on Prime hardware. If anyone has written or ported a device driver to Primos, and wants to have it added to the tape, please contact me.

Macros

Multiple, Independent Marks

Jim Fox
University of Washington

TEX provides marks to help synchronize output routines with macro expansion. They are necessary because macros are expanded before text is fitted onto a page — and therefore are expanded before page numbers are assigned or page boundaries are known. The TEXbook has numerous examples of elementary mark usage. And, as long as only one kind of information, chapter numbers, for example, needs to be communicated, the marks as described work well. However, some implementations need to keep track of a couple of things, or need to use all of the marks on a page. A more flexible approach is required.

Knuth considers this problem in the TEXbook when he suggests passing several 'independent' items of information with "`\a \or \b \or \c ...`" constructions, but this is really a deception. The items in the `\or` constructs are not very independent — they all have to change values at the same time.

The macro package presented here provides multiple, independent marks for output routine synchronization. It solves the concurrency problem by keeping all mark text in lists — with seperate lists for independent marks. A continually incrementing sequence number identifies which elements of the list correspond to the conventional top, first, and bot marks. This sequence number constitutes the only real mark in the system.

The mark lists are both an advantage and an encumberance. Since all marks on a page are recorded, they are all available to the output routine and can be used for seemingly un-mark related functions. (An example is given at the end of the article.) The marks also take up macro space. Documents such as dictionaries that use a lot of marks should be carefully coded to use the shortest mark text possible. The TEX primitive `\noexpand` can be very useful here.

How the marks are used

Since each of the multiple marks is completely independent of the others, their usage is quite simple. First define a new mark, `\abc`, for example, by

```
\newmark\abc
```

Then use `\abc` as if it were `\mark`

```
\abc{mark text}
```

or

```
\abc{top text \else bot text}
```

In the output routine extract the `abc` marks with

```
\getmarks\abc
```

This defines three control sequences: `\topabc`, `\firstabc`, and `\botabc`.

You can then use these just as you would use the conventional `\topmark`, `\firstmark`, and `\botmark`.

How the macros work

List macros are discussed in Appendix D of the TEXbook. Mark lists are similar to those in the book, but have a sequence number in addition to the list marker and text. The lists have the form:

`\m@rker`⟨seq⟩`{`⟨text⟩`}` `\m@rker`⟨seq⟩`{`⟨text⟩`}` ...

where `\m@rker` is undefined until the list is expanded, ⟨seq⟩ is an increasing decimal number, and ⟨text⟩ is the mark text.

The argument of `\newmark` is defined to add a new item to the corresponding list. Note that the sequence number is shared by all lists. For example:

```
\newmark\x \newmark\y
\x{a} \y{b} \x{c}
```

defined the mark lists to be

```
\list@x={\m@rker0{}\m@rker1{a}\m@rker3{c}}
\list@y={\m@rker0{}\m@rker2{b}}
```

When the lists are expanded in an output routine, via `\getmarks`, each sequence number is compared to the numbers in the real `\topmark` and `\botmark`. Let ⟨top⟩ and ⟨bot⟩ be those numbers, respectively, and let the list we are processing be `\list@xxx` (from `\newmark\xxx`). Then the text of the list element with the greatest ⟨seq⟩ where ⟨seq⟩ $\leq$ ⟨top⟩ becomes the topmark (`\topxxx`). The text of the first list element where ⟨seq⟩ $>$ ⟨top⟩ becomes the firstmark (`\firstxxx`). Finally, the text of the last list element where ⟨seq⟩ $\leq$ ⟨bot⟩ becomes the botmark (`\botxxx`).

An example that uses all of the marks

Here is a partial crossreference capability that makes use of mark lists.

Define an alternate mark list scanner

```
\catcode'@=11
\def\m@rksdef@#1{%
   \ifnum\curm@rk>\bm@rk \let\m@rker=\m@rkrecover
     \t@b=\e@{\e@\m@rker\the\curm@rk{#1}}%
   \@else \ifnum\curm@rk>0
     \m@rk@csafter{\xdef}{#1}{\number\pageno}\@fi
     \t@a={\m@rker0{}}\@fi}
\catcode'@=12
```

At the beginning of your file initialize the cross-reference mark list

```
\newmark\xref % marker for cross references
```

and in the output routine

```
\begingroup\let\m@rkscan@=\m@rksdef@
 \getmarks\xref
 \endgroup
```

Then a cross-reference definition, `\xref{alfa}`, for example, will define the control sequence `\alfaxref` to expand to the page number of the page containing the "`\xref{alfa}`". The output routine must have processed the page in question, of course.

The marks macro package

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% multi-marks macro package
% by Jim Fox, Oct 1986
%
\catcode'\@=11
%
%     \expandafter is used a lot, so we use a short name for it
%
\let\e@=\expandafter
%
\toksdef\t@a=0 \toksdef\t@b=2 \toksdef\t@c=4 % temporary registers
%
%     \list@cs makes a control sequence by adding the preface 'list@'
%     to its argument;  e.g., \list@cs\xx --> \list@xx
%     Note that it requires a double expansion
%
%     \list@csafter reaches over its argument to create a control sequence
%     via \list@cs; e.g., \list@csafter\def\xx --> \def\list@xx
%
\def\list@cs#1{\csname list@\string#1\endcsname}
\def\list@csafter#1{\e@\e@\e@#1\list@cs}
```

```
%
%     \newmark\xx defines a new mark.  It initializes the mark's
%     list macro, \list@xx, and defines \xx.
%
%     The list macros, \list@xx in this example, have the form:
%         \m@rker<seq>{<text>}
%         \m@rker<seq>{<text>}.....
%     where \m@rker is undefined until the list is actually expanded,
%     <seq> is a continually incrementing, decimal sequence number,
%     and <text> is the actual mark text.
%
%     The new mark, \xx in this example, is defined to add its argument,
%     along with a \m@rker and sequence number to the end of the
%     corresponding list.  It also sets a 'real' mark containing the
%     new sequence number.
%
\newcount\m@rkcounter\m@rkcounter=0
\def\newmark#1{\begingroup\escapechar=-1
  \list@csafter\gdef#1{\m@rker0{}}% start list
  \long\gdef#1##1{\begingroup\escapechar=-1\setm@rks
    \list@csafter\addm@rk#1{\the\m@rkcounter{##1}}\endgroup}\endgroup}
%
%     \addm@rk puts the \m@rker, sequence number, and new mark text
%     at the end of a 'list' macro.
%
\long\def\addm@rk#1#2{\t@a=\e@{#1}\xdef#1{\the\t@a\noexpand\m@rker#2}}
%
%     \setm@rks increments the mark counter and sets the mark
%
\def\setm@rks{\global\advance\m@rkcounter by1\mark{\the\m@rkcounter}}
%
%     In an output routine \getmarks\xx will extract from \list@xx
%     the appropriate \topxx, \firstxx, and \botxx marks.
%
\newcount\tm@rk \newcount\bm@rk  % top and bot mark numbers
\newcount\ecs@ve  % saves the value of \escapechar
\newif\ifnofirstm@rk  % true when \firstxx has been defined
\newcount\curm@rk  % records the sequence number from the mark list
%
%     \m@rk@csafter reaches over its first argument to build
%     one of the control sequences; \topxx, \firstxx, or \botxx
%     e.g., if the text was \getmarks\xx, then \m@rkname expands to 'xx'
%     and  \m@rk@csafter\show{top} --> \show\topxx
%     and  \m@rk@csafter{\long\e@\def}{bot} --> \long\def\botxx
%
\def\m@rk@csafter#1#2{\e@#1\csname#2\m@rkname\endcsname}
```

```
%
%     \getmarks scans the mark list and sets:
%
%        \topxx   = the text of the last list item whose sequence number
%                   is less than or equal to \topmark
%        \firstxx = the text of the first list item whose sequence number
%                   is greater than \topmark
%        \botxx   = the text of the last list item whose sequence number
%                   is less than or equal to \botmark
%
\def\getmarks#1{\tm@rk=0\topmark \bm@rk=0\botmark
  \t@b={}\nofirstm@rktrue
  \ecs@ve=\escapechar \escapechar=-1
  \edef\m@rkname{\string#1}%  used by \m@rk@csafter
  \let\@fi=\fi\let\fi=\relax \let\@or=\or\let\or=\relax
  \let\@else=\else\let\else=\relax % so mark text can contain \else,\or,\fi
  \let\m@rker=\m@rkscan \list@cs#1\m@rkend \m@rkrestore % 'do' the list
  \let\else=\@else \let\fi=\@fi \let\or=\@or
  \escapechar=\ecs@ve}
%
%     \m@rkscan looks at the next list item and defines the appropriate
%     \topxx, \firstxx, and \botxx control sequences
%     They are \long\def'd to handle the unusual case of marks with \par's
%
\def\m@rkscan{\afterassignment\m@rkscan@\curm@rk=}
\long\def\m@rkscan@#1{%
  \ifnum\curm@rk>\tm@rk\@else\m@rk@csafter{\long\e@\def}{top}{#1}\@fi
  \ifnum\curm@rk>\bm@rk \let\m@rker=\m@rkrecover
       \t@b=\e@{\e@\m@rker\the\curm@rk{#1}}%
  \@else \ifnofirstm@rk\m@rk@csafter{\long\e@\def}{first}{#1}%
            \ifnum\curm@rk>\tm@rk\nofirstm@rkfalse\@fi\@fi
      \m@rk@csafter{\long\e@\def}{bot}{#1}%
     \t@a=\e@{\e@\m@rker\the\curm@rk{#1}}\@fi}
%
%     at the end some of the list must be restored
%
%     \m@rkrecover is used when there are unscanned list items, otherwise
%     \m@rkend is reached and stops the scan
%
\long\def\m@rkrecover#1\m@rkend{\t@c={\m@rker#1}}
\def\m@rkend{\t@c={}}
%
%     \m@rkrestore rebuilds the list (with at least a new \topxx}
%
\def\m@rkrestore{%
   \m@rk@csafter{\long\e@\xdef}{list@}{\the\t@a\the\t@b\the\t@c}}
%
\catcode`\@=12
%
```

Form Letter Macros

John S. Garavelli
University of Illinois at Chicago

In reference to the query from Mr. John Lee concerning a program for generating form letters, here is a fairly straightforward and unsophisticated TEX program which I have written. This program, which I call FORMLETTER.TEX, asks the user to provide two file names. The first file contains a list of file names, one file name per line, each of which is an address file to be described later. The second file contains the TEX form letter. The TEX form letter must end with \vfill\eject rather than \bye and may freely use the following macros.

\title	for example Dr. or Ms.
\firstname	the first name
\middle	the middle name or initial
\lastname	the last name
\firstline	the first line from the address
\secondline	the second line from the address
\thirdline	the third line from the address
\fourthline	the fourth line from the address
\fifthline	the fifth line from the address

The address files must, unfortunately, have a standard format. The first line of each address file must have four entities delimited by spaces corresponding to the first four macros listed above, and they must occur in that order. If one of these entities does not exist in an address file it must be replaced by an empty box; for example, the line

```
Dr. John {} {von Neumann}
```

would be parsed to yield the equivalent of

```
\def\title{Dr.}
\def\firstname{John}
\def\middle{}
\def\lastname{von Neumann}
```

There must be at least one line, the first line, in each address file, and if any of the other lines are empty then the corresponding macros are empty boxes. Non-address information, such as phone numbers, can be stored in these address files on any lines after the fifth.

This is the file FORMLETTER.TEX.

```
\def\reveal#1{\immediate\write1{#1}}
\def\lineup#1 #2 #3 #4 {%
    \def\title{\string #1}
    \def\firstname{\string #2}
    \def\middle{\string #3}
    \def\lastname{\string #4}}
%                   Redefine \loop
\long\def\loop#1\repeat{%
    \def\body{#1}\iterate}
\newread\afile
\newread\listfile
\newread\letfile
\newif\ifnotdone

\reveal{}
\message{Filename of file list: }
\read-1 to\filename
\openin\listfile=\filename
\ifeof\listfile \message{The file
    \filename does not exist.}\end\fi
\message{Filename of letter file: }
\read-1 to\letfile
\loop
  \read\listfile to \filename
  \ifeof\listfile \notdonefalse
  \else \notdonetrue \fi
  \ifnotdone
    \openin\afile=\filename
    \ifeof\afile \message{The file
        \filename does not exist.}\end\fi
    \reveal{Reading \filename}
    \read\afile to \firstline
    \read\afile to \secondline
    \read\afile to \thirdline
    \read\afile to \fourthline
    \read\afile to \fifthline
    \closein\afile
    \expandafter\lineup\firstline
    \input \letfile
\repeat
\closein\listfile
\bye
```

AutoLetter: A TEX form letter procedure

Edwin V. Bell, II
Department of Physics and Astronomy
University of Kansas

The inquiry from John Lee (TUGboat, October 1986) concerning form letters was particularly interesting to me as I had recently completed such a procedure at the request of our office staff. The procedure (called AutoLetter) is a "plain" TEX procedure which requires two files, one containing a delimited listing of names and addresses, the other a generic form letter.

Each entry in the address file is followed by a line containing a single asterisk. The last entry must also be followed by this line, which in turn *must* be followed by the end-of-file mark. Each individual entry consists of six lines for the addressee's title, name, etc., followed by as many address lines as desired. The lines for the addressee's name are:
(1) Title (Prof., Mr., Ms., etc.)
(2) First name/initial
(3) Name to be used as a familiar form of address
(4) Middle name(s)/initial(s)
(5) Last/family name
(6) Additional name information (Jr., II, Esq., etc.)
These lines are then used to reconstruct the person's name for various forms of address (including for the address block) and to address the person formally (Sir, Madam, Sir or Madam), familiarly (by item (3) above or, if (3) is blank, by first name), or by title and last name. If any of these lines are left blank, AutoLetter doesn't mind, but if the entry is irrelevant or absent from the person's name, *it must still be left blank.* An example of an address entry might be:

```
Mr.
Edwin
Ed
V.
Bell
II
Department of Physics \& Astronomy
University of Kansas
Lawrence, KS\ \ \ 66045
*
```

The letter file is quite simple and uses macros that are the same as or slight variations of those listed in Appendix E of *The TEXbook.* AutoLetter takes care of the placement of the date and address block (and could as well the letterhead, although I have not yet implemented that here), so the letter file might appear as:

```
\letterbody
Dear \person--

..text...

\closing
Sincerely,
..

\annotations...
\ps...

\endletter
```

AutoLetter consists of two files, one the TEX procedure itself, the other containing the letter-formatting macros (and local macros, if desired). Below are the AutoLetter procedure and our local letter-formatting macros.

The AutoLetter procedure

```
\input letter_format

\message{+-----------------------------------------+}
\message{| AutoLetter Version 1.1 -- 9 October 1986 |}
\message{+-----------------------------------------+}

\newread\addressfile

\message{Enter name of file containing the addresses:}
\read-1 to\addfile
\openin\addressfile=\addfile

\message{Enter name of file containing the body of the letter:}
\read-1 to\letfile
```

```
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       Now read in an entry containing the person to whom to send the letter
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

\endlinechar=-1
\read\addressfile to\text \let\title=\text

\loop
    \read\addressfile to\text \let\firstname=\text
    \read\addressfile to\text \let\nickname=\text
    \read\addressfile to\text \let\middlename=\text
    \read\addressfile to\text \let\lastname=\text
    \read\addressfile to\text
        \ifx\text\blank\def\jr{}\else\def\jr{, \text}\fi
        \endlinechar='015
    \message{Now processing letter to \addressee.}
    \getaddress
    \doletter
      \endlinechar=-1
    \read\addressfile to\text \let\title=\text
    \ifeof\addressfile\endfalse\else\endtrue\fi
    \ifend
\repeat

\bye
```

The letter-formatting macros

```
\input whatever local macro package you wish here

\twelvept    % 12-point size is so much easier to read

\hsize=6.5truein
\vsize=8.0truein
\hoffset=1.0truein
\voffset=1.5truein

\raggedbottom
\interlinepenalty=1000
\parindent=0pt
\parskip=0pt

\nopagenumbers
```

```
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%      Define some stuff so that the headlines will come out the way
%      we want them to be.
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

\newbox\headbox
\headline={\ifnum\pageno>1
   {\twelverm
      \global\setbox\headbox=\vbox\bgroup%
         \leftline{\addressee}
         \leftline{\today}
         \leftline{Page \folio}\egroup}\copy\headbox
   \else\hfil\fi}

\def\makeheadline{\vbox to 0pt{\vskip-70.6pt
   \line{\vbox to30.6pt{}\the\headline}\vss}
   \nointerlineskip}

\newif\ifend

%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       Macros for reading in the address from the file.
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

\def\blank{}

\def\endletter{\endmode\vfill\eject\pageno=1}

\newdimen\longindentation \longindentation=10truecm
\newbox\theaddress
\newif\ifast
\def\aster{* }
\def\getaddress{{\global\setbox\theaddress=\vbox\bgroup\raggedright%
   \hsize=\longindentation
   \everypar{\hangindent2em}
      \line{\vbox to10.2pt{}\addressee\hss}\egroup%
   {\loop{\read\addressfile to\text
      \ifx\text\aster\astfalse\else\asttrue\fi
      \ifast\global\setbox\theaddress=\vbox\bgroup\unvbox\theaddress
         \line{\vbox to 10.2pt{}\text\hss}\egroup}\repeat}}}
```

```
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       Macros for formatting the letter.
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

\def\letterbody{\beginparmode}
\def\closing{\beginlinemode\getclosing}
{\obeylines\gdef\getclosing #1
   #2
      {#1\nobreak\bigskip \leftskip=\longindentation #2
      \nobreak\bigskip\bigskip\bigskip\bigskip % space for signature
      \def
      {\endgraf\nobreak}}}
\def\annotations{\beginlinemode\def\par{\endgraf\nobreak}\obeylines\par}
\def\ps{\beginparmode\nobreak
   \interlinepenalty5000\def\par{\endgraf\penalty5000}
   P.~S.\ }

\def\address{\beginlinemode \copy\theaddress \endgroup}
\def\doletter{{\leftskip=\longindentation%
      \rm\today\bigskip\address\bigskip}
   \input \letfile}

%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       Macros for setting up forms of salutation.
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

\def\mr{Mr.}
\def\ms{Ms.}
\def\mrs{Mrs.}
\def\miss{Miss}

\def\addressee{\ifx\blank\title\else\title\space\fi%
   \firstname\space\ifx\blank\middlename\else\middlename\space\fi%
   \lastname\jr}
\def\person{\ifx\blank\title Sir or Madam\else\title\space\lastname\fi}
\def\familiar{\ifx\blank\nickname \firstname\else\nickname\fi}
\def\formal{\ifx\mr\title Sir\else{\ifx\ms\title Madam\else%
   {\ifx\mrs\title Madam\else{\ifx\miss\title Madam\else{\person}\fi}%
   \fi}\fi}\fi}
```

The AutoLetter procedure prompts the user for the names of the address and letter files and then goes to work. A similar procedure (called LABELS) can also be used with the address file to produce 33 labels/page (3 columns of 11 labels). This enables us to produce form letters with or without labels or to maintain mailing lists without requiring letters (for abstract mailings, for example).

If anyone is interested in having these procedures, drop me a line. If response is high, I would be more than willing to provide them to the TEX community at large. Mail may be sent to:

Bitnet: Bell@UKANVAX
SPAN: KUPHSX::Bell

LaTeX

Contents of LaTeX Style Collection as of 15th February 1987

Ken Yap
University of Rochester

The following files are available for anonymous ftp from `Rochester.Arpa` in directory `public/latex-style`. You should retrieve the file `00index` first to obtain a brief description of current directory contents.

File	Description
`00directory`	
`00index`	
`00readme`	
`a4.sty`	Set page size to A4
`acm.bst`	ACM BibTeX style
`agugrl.sty`	AGU Geophysical Research Letters style
`agujgr.sty`	AGU Journal of Geophysical Research style
`amssymbols.sty`	Load AMS symbol fonts
`biihead.sty`	Underlined heading
`cyrillic.sty`	Load cyrillic font
`deproc.sty`	DECUS Proceedings style
`deprocldc.tex`	Paper that describes the above
`docsty.c`	Program to convert .doc to .sty by stripping comments
`docsty.readme`	
`doublespace.sty`	Double spacing in text
`drafthead.sty`	Prints DRAFT in heading
`dvidoc.shar1`	Sh archive of `DVIDOC`, DVI to character device filter for Unix BSD systems
`dvidoc.shar2`	
`epic.shar1`	Sh archive of extended picture environment
`epic.shar2`	
`format.sty`	Print FP numbers in fixed format
`fullpage.doc`	Get more out of a page
`fullpage.sty`	
`geophysics.sty`	Geophysics journal style
`ieeetr.bst`	IEEE Transactions BibTeX style
`layout.readme`	Prints nice diagram showing page parameters
`layout.tex`	
`lfonts_ams.readme`	Use AMS symbols in LaTeX
`lfonts_ams.tex`	
`lgraph.shar`	Sh archive of data to graph command filter in Pascal
`natsci.bst`	natural sciences generic BibTeX style
`newalpha.bst`	Modified alphabetic BibTeX style
`nopagenumbers.doc`	Remove page numbers
`nopagenumbers.sty`	
`siam.bib`	SIAM BibTeX style
`siam.bst`	
`siam.doc`	SIAM LaTeX style
`siam.sty`	
`siam.tex`	
`siam10.doc`	
`siam10.sty`	
`siam11.sty`	
`siam12.sty`	
`slem.doc`	Change `\sl` to `\em`
`slem.sty`	
`spacecites.doc`	Modified to give spacing between citations
`spacecites.sty`	
`suthesis.doc`	Stanford U thesis style
`suthesis.sty`	
`texindex.doc`	Style file and processor for index entries. Works under VMS.
`texindex.pas`	
`texindex.sty`	
`texnames.doc`	Define a couple more TeX names
`texnames.sty`	
`threepart.sty`	Three part page headers
`uct10.doc`	U of California thesis style
`uct11.doc`	
`uct12.doc`	
`ucthesis.doc`	
`ucthesis.readme`	
`vdm.doc`	Vienna Development Method LaTeX style
`vdm.sty`	
`vdm.tex`	
`ws87.p`	Wordstar 8 bit filter
`wsltex.c`	Wordstar to LaTeX filter, C version
`wsltex.p`	Wordstar to LaTeX filter

More submissions are very welcome. Send them to

Ken
`LaTeX-Style@Rochester.Arpa`
`LaTeX-Style@cs.rochester.edu`
`..!rochester!latex-style`

Editor's note: People sending future submissions should note that some gateways to Bitnet strip off everything beyond 80 columns, and perhaps corrupt some other data as well (ASCII tabs may or may not remain intact). Please structure your file so that it will survive.

For Internet users: how to ftp

An example session is shown below. Disclaimer: ftp syntax varies from host to host. Your syntax may be different. The syntax presented here is that of Unix ftp. Comments in parentheses.

Non-Internet users: how to retrieve by mail

An archive server for LaTeX files has been installed. Send a piece of mail to `LaTeX-Style` (`@rochester.arpa`, `@cs.rochester.edu`, via uucp or your favourite gateway) in the following format.

- Subject line should contain the phrase "`@file request`".
- The body of the mail should start with a line containing only an `@` (at) sign.

Important! The first line following the "at" line should be a mail address **from** Rochester **to** you. (Undeliverable mail will be silently dropped on the floor.)

- Follow your return address by the names of the files you want, either one to each line, or many to each line, separated by spaces.
- End with a line containing only an `@` sign.
- Case is not significant.

For example, if you are user at `site.bitnet`, this is what you should send:

```
To: latex-style@rochester.arpa
Subject: @file request

@
user%site.bitnet@wiscvm.wisc.edu
00readme
00index
@
```

A word to the wise: it is best to fully qualify your mail address. Our mailer knows about some gateways but not all. Examples:

```
user%site.bitnet@wiscvm.wisc.edu
user%site.csnet@relay.cs.net
```

Do not include any messages in the mail. It will not be seen by human eyes. Be patient as the server is actually a batch program run once a day. Files will be sent in batches, each not exceeding 100kbytes in size.

Editor's note: Traffic on the network servers and gateways has been very high recently, and in order to provide improved service, there have been some volunteers to maintain local "slave" repositories of the LaTeX style collection. There is usually a geographic or network restriction requested, since the idea is to cut down traffic, not add to it. The following areas will be covered by the volunteers listed.

- Bitnet users: Texas A&M maintains a list- and file-server which is already handling (with TEX-L) much of the Bitnet distribution of TeXhax. An inquiry via listserv will retrieve a list of all TeX-related files:
  `tell listserv at tamvm1 get tex filelist`
- United Kingdom, for users of JANET or uucp: Stephen Page, `sdpage@uk.ac.ox.prg` or `...!ukc!ox-prg!sdpage`
- European users of BITnet: Christoph Gatzka, `zrgc002@dtuzdv5a.Bitnet`

Additional volunteers should contact Ken.

Sample FTP session for Internet users

```
% ftp cayuga.cs.rochester.edu   (a.k.a. rochester.arpa, a.k.a. 192.5.53.209)
 ...                            (general blurb)
user: anonymous
password: <any non-null string>
ftp> cd public/latex-style      (where the files are)
ftp> ls                         (to see what is there)
 ...                            (lots of output)
ftp> get 00index
 ...                            (more blurb)
ftp> quit
```

Form Letters

Graeme McKinstry
University of Otago

The October TUGboat (vol. 7, no. 3) contained an enquiry, from John Lee, regarding merging a standard letter with a separate file containing addresses. I have written a few TEX macros which enable addresses and *opening lines* (`\opening`) to be merged with a standard letter.

The style called *merge* is a substyle under LETTER.

A letter is set up as follows:

```
\documentstyle[merge]{letter}

\address{...}      % your address
\signature{...}    % your signature

\begin{document}
\begin{merge}{myaddresses}
        % file MYADDRESSES.TEX
        % contains addresses + opening
With regard to ...   % letter starts

\closing{...}   % closing
\ps{...}        % \ps, \encl, \cc, etc.

\end{merge}
\end{letter}
```

The file, MYADDRESSES.TEX, would contain:

```
{University of Otago,   % address
 Dunedin,
 New Zealand}
{Dear Graeme,}   % \opening
{...}            % next address
{...}            % next \opening
   ...           % etc.
```

Notes:

- Opening line-braces are not necessary if only 1 line.
- It is important to leave *no blank lines* at the end of the file.

Finally the macros:

```
\typeout{Merge substyle-release 6 October 1986 by Graeme McKinstry}

\newbox\store
\long\def\contents{\global\setbox\store=\vbox\bgroup}  % store the contents
\long\def\endcontents{\egroup}                         % of the letter

\def\sendaddress{test}
\def\openingtext{test}
\newif\iffirsttime
\firsttimetrue

\newread\addrfile   % allocate an input stream

\def\@openfile{\openin\addrfile=\mergef@le   % open the address file
\ifeof\addrfile                             % i.e., didn't open successfully
     \loop
          \immediate\write16{Could not open file \mergef@le}
          \closein\addrfile                 % close the input stream
          \read16 to \mergef@le             % get another file name
          \openin\addrfile=\mergef@le       % open up input stream
          \ifeof\addrfile
     \repeat                            % repeat until successfully opened
\fi}

\def\merge#1{\def\mergef@le{#1 }\@openfile\readfile \contents}

\def\readfile{\global\read\addrfile to\sendaddress      % get the address
              \ifeof\addrfile
              \else
                 \global\read\addrfile to\openingtext   % get the opening line
                 \iffirsttime
                 \else
                    \endmerge
                 \fi
              \fi}

\newbox\l@tterbox
\def\endmerge{\iffirsttime\endcontents\global\firsttimefalse\fi %
                                        % end contents if it is the first time
              \begin{letter}{\sendaddress}
              \opening{\openingtext\vskip2\parskip}
              \setbox\l@tterbox=\copy\store  % copy the contents of the letter
              \unvbox\l@tterbox
              \end{letter}
              \ifeof\addrfile
                 \message{End of file}
              \else
                 \readfile                  % loop round yet again
              \fi}
```

LaTeX's Index Processing

Maurizio Zocchi

This note suggests an easy way to process a LaTeX index. As known, index files produced by the LaTeX system involve a lot of manual work to obtain the final form of an index.

In order to do that operation automatically we need a program able to: (1) sort the index's entries alphabetically, (2) remove possible duplicate keys, (3) collect all page numbers related to each entry.

Such a program will permit a simple two-step method to correctly construct a document's index. When LaTeX processes a document, it can produce index files, which may be adjusted by the program mentioned above. During the next processing, LaTeX will find a ready-to-use index file.

Obviously, sorting of an index's entries can be performed by a standard utility routine, but data must be provided with a specified form. Sort routines can accept specification on key positions, lengths and other information to 'direct' the process, but in the simplest case there is one key, as long as the entire record. In this case, the index's lines have different lengths so the sort routine is not directly applicable. The index is a normal ASCII file and consists of several lines in the form

```
\indexentry{title}{page no.}
```

and processing the variable length key '`title`' may lead to wrong results, because of the criteria by which characters are sorted. A way to avoid this is to fix the initial key position and length to standard values. Of course, this solution occupies much memory space and limits the generality of the application.

Another solution is to estimate the maximum key length and then fill all the smaller title record keys with blanks; the page numbers must also be adjusted with leading zeros. In this way, the sort routine will compare the data keys exactly, without any misplacement.

Furthermore, accented letters need particular care, because every accent is composed by calling the `\accent` primitive or an alignment command and their presence in the index can cause an incorrect result of the sort.

It is supposed here that accents are written in the index in their unexpanded form, so it is possible to invert the position of the characters composing every accent to partially preserve the alphabetical order. For example, the sequence '`\'E`' becomes '`E\'`'; this is particularly important when the accent appears in the first position of the key.

After the sort it is necessary to merge all the page numbers related to each title key, and then remove blanks and zeroes. For example, two entries that appear in the index as

```
\indexentry{Guess     }{0020}
\indexentry{Guess     }{0122}
```

need to be 'compacted' into the form

```
\indexentry{Guess}{20, 122}
```

that is ready to be processed by LaTeX. Of course, all accent control sequences are inverted again and the result is the initial situation.

Everything described above is implemented under MS-DOS and VAX/VMS. Let me define more precisely what kind of programs were developed.

Under the VAX/VMS system there was developed a program called INDTEX that can execute the three steps mentioned, provided that sorting is on system charge, too. The sort routine is called within the program.

Under MS-DOS a Pascal program was developed for each step of the process; these programs are:

- INDTEX — performs 'expansion' of input lines.
- COMPACT — provides 'reformatting' of files after sort.

It seems important to remember here that the MS-DOS sort cannot handle files with sizes larger than 64KB; so INDTEX splits output into different files, which are sorted separately. Then, a MERGE program establishes the correct situation. However, users can deal with simple batch commands.

INDTEX was applied during the construction of a publishing house catalog of 65 pages. The catalog includes a title index, index of arguments, and author index. The index files were constructed by using the `\contentsline` form, and the line structure was expanded to include the volume code and the author's name. The final document was 97 pages.

The total time required was about 45 minutes for processing text and about an hour for sorting and adjusting the indexes.

In the future, ASCII accented characters may be used to replace a LaTeX command by a single character. Something will be done to improve the execution. Special thanks to A. Mattasoglio of Cilea and G. Canzii of TECOGRAF.

For further information or suggestions (which are welcome), please contact

M. Zocchi
c/o TECOGRAF
via Plinio 11
20100 Milano Italy

Typesetting 'Normaltext'*

Reinhard Wonneberger
Hamburg†

Abstract

To improve their readability, ancient texts can be displayed in a way known from mathematical formulas. This kind of display is backed by modern linguistic research, which has revealed the intrinsic structuring of texts. This revelation has produced the need for a congenial form of display. As a compromise between linguistic accuracy and practical needs, the concept of *Normaltext* has been developed and has proved helpful especially with the study of biblical and similar texts. Typesetting such texts in TEX or LATEX can save a lot of tedious work, but requires some special macros that are presented here.

1 On Readability

No one would think of typesetting long and complicated mathematical formulas in the shape of an ordinary paragraph , i.e. as a continuous stream of symbols. To display them as a separate block of text contributes to their *readability*, and vertical alignment can be used to clarify the internal structure of the formula.

The same holds true for ancient texts. Their words or, sometimes, even their single letters have as much weight for interpretation as mathematical symbols. One such text that is hard to understand, when read in paragraph mode, is the so-called *Trostwort an Baruch* in *Jeremiah 45*, which we quote here from the *The New English Bible*:[1]

> THE WORD WHICH THE PROPHET JEREMIAH SPOKE to Baruch son of Neriah when he wrote these words in a book at Jeremiah's dictation in the fourth year of Jehoiakim son of Josiah, king of Judah: These are the words of the LORD the God of Israel concerning you, Baruch: You said, 'Woe is me, for the LORD has added grief to all my trials. I have worn myself out with my labours and have had no respite.' This is what you shall say to Baruch, These are the words of the LORD: What I have built, I demolish; what I have planted, I uproot. So it will be with the whole earth. You seek great things for yourself. Leave off seeking them; for I will bring disaster upon all mankind, says the LORD, and I will let you live wherever you go, but you shall save your life and nothing more.

If we can find a sound method of applying the principles of display and vertical alignment to such texts, we can be sure of advancing their readability a great deal.

2 The Method

To achieve a method of text display that is not just left over to our intuition, we have to work out a linguistic model of segmentation. A method meeting these requirements is described in my article on *Normaltext (N)*,[2] and a text that is segmented and displayed according to this model will be called *Normaltext* to make clear that this form of display should represent the standard.

The method of *Normaltext* is based upon the linguistic model of text structuring developed by *Elisabeth Gülich* and *Wolfgang Raible*. The kernel of that theory can be summarized by the list of *Textgliederungssignale* given in Fig. 1.

To mark all the elements from the list in an existing text is a task too complicated to provide an undisputed text display at the beginning of research. From some working experience, however, it will become clear soon that there is a basic distinction be-

---

* This paper is dedicated to my teacher in the Old Testament, Prof. Dr. Klaus Koch, on the occasion of his 60^{th} birthday on October 4, 1986.

† The present article and also the macros contained in it were developed as a test case for the new ST-TEX (cf. *Klaus Guntermann: Porting TEX to the ATARI ST. TUGboat 7,3 (1986) p. 164.*) and to gain experience in the possibilities of 'job-sharing' and data transfer between the PC and the host at *DESY, Notkestraße 85, D 2000 Hamburg 54, FRG.* I should like to thank *P. K. Schilling* and *P. Stackhouse* for their help with corrections and English style. Comments should be sent to *R. W., Drachenstieg 5, D 2000 Hamburg 63* or through *Bitnet/Earn* to *B03WBG* at *DHHDESY3*.

1 New York: Oxford University Press 1971, p. 972. — The word *Lord* is capitalized when used for the name of God (JHWH).

2 *Normaltext und Normalsynopse. Neue Wege bei der Darstellung alttestamentlicher Texte. Zeitschrift für Sprachwissenschaft 3 (1984) 203–233.*

1. Metakommunikative Sätze
2. Substitution auf Metaebene
3. Episoden- und Iterationsmerkmale
4. Veränderung der Konstellation der Handlungsträger
5. Renominalisierung
6. Satzkonjunktionen und Satzadverbien

Figure 1: Prioritätenliste der Textgliederungssignale. This table is taken from *Normaltext* p. 213; more detailed information can be found there.

tween the first two and the remaining items. While the latter denote properties of one and the same text, the former describe the *embedding* of one text into another. Text embedding is normally beyond dispute and thus can provide a sound basis for our text display. On the other hand, a consensus will easily be reached on the smallest parts of a text, i.e. parts with a sentence-like character.[3]

Thus, with our method of text display, we shall not get lost in the complexity of elements, but concentrate on the two fundamental features of *basic lines* [*Basiszeilen*] and *text embedding* [*Texteinbettung*].

For an outline of that theory, we can refer to our paper on Normaltext. Suffice it here to sketch the main rules governing the display form of a *Normaltext:*

1. The text is broken into basic lines [*Basiszeilen*] according to the so-called signals of structure [*Textgliederungssignale*]. For Hebrew texts, the most common signal is the narrative form.
2. Each line is prefixed by a verseline label, showing
   (a) the chapter number in bold, if it is the beginning of a new chapter;
   (b) the verse number, if it is the start of a new verse;
   (c) the letter 'b', if it is the start of the second part of the verse after the *atnach* (applicable to Hebrew only);
   (d) one of the letters 'A B C ...' showing that it is the n-th line relative to the current verse.
3. These lines are indented *after* the label if they are from an embedded speech [*Redewiedergabe*], and this is done recursively if another speech is embedded.
4. Redactional additions can be indented *in front of* the label by a larger amount.

These rules, when applied to our example text *Jeremiah 45*, yield the display of Fig. 2. To force the material to stay together we have put it into a figure so that it can float to an appropriate place. This will help us to get an impression of the *shape* of the text which is expressed through indentation.

3 Fields of Application

The ideas of *display* and *vertical alignment* are especially useful in high level text research, and the method of *Normaltext* and *Normalsynopse* was developed for the research done by my teacher in the Old Testament, *Prof. Dr. Klaus Koch*, and his *Arbeitsstelle für Profetenforschung* in the first place. Some examples of the resulting kind of text display are given in the previously mentioned paper on *Normaltext (N)* as well as in my *Leitfaden (L)*.[4]

1. A synopsis of the Massoretic, Septuagint, and Tagum text of *Isaias 19,25b* (*L: Tafel 1*; *N: Abb. 18*).
2. A synopsis in German of *Judges 6,8f* and *1. Samuel 10,18* (*N: Abb. 14*).
3. A 'Normaltext' in German of *Isaias 7,1–9* (*N: Abb. 9*).
4. A synopsis of a reconstructed Massoretic text and its Septuagint counterpart for *1. Samuel 14,41* (*L: Tafel 3*).
5. A synopsis of the Septuagint and Theodotion Greek texts of *Daniel 14,27* (*N: Abb. 17*) and *Daniel 14,40* (*L: Tafel 4*).

[3] The standard case for Hebrew will be marked very clearly by a *Narrativ* at the very beginning. There are other clear cases, too, and they are assembled in *N: Abb. 8*. There remain only some cases that require decision (cf. the discussion in *N: p.214*).

[4] Leitfaden zur Biblia Hebraica Stuttgartensia. *Göttingen: Vandenhoeck & Ruprecht* 1984.

1	**45**	THE WORD
2	*B*	WHICH THE PROPHET JEREMIAH SPOKE
3	*C*	to Baruch son of Neriah
4	*b*	when he wrote these words in a book at Jeremiah's dictation
5	*E*	in the fourth year of Jehoiakim son of Josiah, king of Judah:
6	2	These are the words of the LORD the God of Israel *b* concerning you, Baruch:
7	3	You said,
8	*B*	'Woe is me,
9	*C*	for the LORD has added grief to all my trials.
10	*b*	I have worn myself out with my labours
11	*E*	and have had no respite.'
12	4	This is what you shall say to Baruch,
13	*B*	These are the words of the LORD:
14	*C*	What I have built, I demolish;
15	*D*	what I have planted, I uproot.
16	*b*	So it will be with the whole earth.
17	5	You seek great things for yourself.
18	*B*	Leave off seeking them;
19	*b*	for I will bring disaster upon all mankind,
20	*D*	says the LORD,
21	*E*	and I will let you live
22	*F*	wherever you go,
23	*G*	but you shall save your life and nothing more.

Figure 2: Jeremiah 45 as Normaltext.

Our method makes it easy to display original texts together with their translation, to relate scientific results to the text [*Beilisten*] and to combine several texts into a synopsis.

While making normalized synopses is admittedly tedious, the effort that has to go into a normalized text will soon pay off. This was clearly shown by our students of Old Testament exegesis using it as a standard in writing their first exegetical paper of their academic career.

The task of producing the *Normaltext*, usually with scissors and glue (the real one), brings the student into close contact with his text, the decisions to be made on its structure draw his attention, from the beginning, to certain exegetical questions,[5] and the displayed Hebrew text in the left column accompanied by its symmetrically shaped translation in the right column provides excellent access both to the text as a whole and to its parts, helping a great deal to control hypotheses found in the learned literature and to develop one's own.[6]

But the method is even worthwhile when used only to provide better access to some modern translation of the ancient text; rf. to the examples contained in *Verheißung und Versprechen*,[7] *1. Samuel 1,9–11* (p. 170), *Genesis 15,1–6* (p. 180), and *Numeri 6,22–27* (p. 201).

It is quite obvious from the examples mentioned so far that producing a normalized text by hand is a very tedious task, and so we developed a special macro for the typesetting of our *Leitfaden* to handle this kind of text. In connection with the table macros of DCF, we were even able to typeset our synopses rather automatically.[8] In this article we are going to discuss a corresponding approach for TEX and LATEX.[9]

[5] Two problem examples *1. Samuel 10,17–20* and *1. Samuel 10,24* are shown in *N*: *Abb.11* and *Abb.13*.

[6] It is often interesting to compare the segmentation according to our model with the conventional segmentation, rf. the example of *1. Samuel 6,21–7,3* in *Tafel 7* of the *Leitfaden*.

[7] R. W. / Hans Peter Hecht: Verheißung und Versprechen. Eine theologische und sprachanalytische Klärung. *Göttingen: Vandenhoeck & Ruprecht* 1986.

[8] The general background of the problem is presented in our article *"Verheißung und Versprechen" — A third generation approach to theological typesetting. TEX for Scientific Documentation, Proceedings of the Second European TEX Conference, Strasbourg, June 19–21, 1986. Lecture Notes in Computer Science. Heidelberg / Berlin: Springer (forthcoming)*, and we should like to refer the reader to this article both for the general background to the present paper and for the discussion of further details.

[9] Rf. Leslie Lamport: LATEX. A Document Preparation System. *Reading, Massachusetts etc. Addison-Wesley.* 1985.

4 Making a LaTeX environment

Our basic idea is to take advantage of the list making capabilities of LaTeX. Though LaTeX allows us to define a new list environment by using the `list` environment inside a `\newenvironment` command, our task is not as trivial as it might look at first sight, since we have to find a way to interfere with LaTeX's `list` environment in a controlled way.

The main task of the normal `list` environment is to give us control over line numbering, and we declare a `\newcounter{linecount}` to that end. This counter will be advanced automatically by every `\item` or replaced by an optional argument of an `\item[arg]` or left blank by an empty argument `\item[]`. To restart the counter in each new environment, we shall include in the definition a `\usecounter{linecount}`. To gain a more coherent display, we also have to reset the `\itemsep`.

4.1 Defining a new list environment

Next, we want our additional formatting to take place before the text of the item is processed, but not at the cost of rewriting the original macros. A little trick will help us here. It is built on the fact that the `\@item` macro which does the real work is finished with an `\ignorespaces` command. Mapping this control sequence to some other macro, we regain control at the end of this macro, just before the text of the the item is processed. Of course it is our first duty to remap `\ignorespaces` to its original state in the substitute macro.

In addition to the `\item` command already provided by the `list` environment, we want a similar command for headings, that do not belong to the source we are going to display, but help to denote the contents of the text or hold certain comments we want to make. Our `\head[arg]` command will also have an optional argument, which will allow us to control whether headlines are counted along with source lines or get some other marking. Unlike the normal `\item` command, it should read the text of the heading as an argument, so that we have it available for further use in running headings, in the table of contents etc., should the need arise some day.

Sometimes our linebreaking will not correspond to the inherited verse or chapter breaks. For these rare cases, we use the `\vl` (verselabel) macro, which will read the argument in the same way as `\item`, but put the verselabel in the text and link it with what follows by a tie.

At the end of our environment we will incorporate a test to make sure that the process of rushing up and down the levels of our staircases has properly come to a rest at ground level. Since that tends not to be the case in real life, we shall oblige ourselves with resetting any open levels explicit=ly with the aid of a `\reset` macro. Apart from ending the previous paragraph, this macro only has to contribute its argument so that the macros contained in it will execute by themselves.

Now we have already finished with the things that will allow us to start our new environment. But before the first item or head can be processed, we have to modify the paragraph making of the `list` environment. To understand the corresponding macro is rather complicated, but the crucial point can be easily seen looking at *Figure 5.3 'The format of a list'* from LaTeX *User's Guide & Reference Manual.* The drawing shows clearly that list items are based on a one-line-`\parshape`, the label being placed by other means. Since we want our source lines to be clearly marked as continuation lines by an indent of `\hangwidth` relative to the starting position of the text, and since that position depends on the constant width of our verse labelling and the varying width of the levels of speech and redaction, we have to use a two-line-`\parshape`. When beginning the environment, the constant parts of indentation must be calculated; these will be modified later by the varying parts.

Now we have at hand all the elements to define our new environment:

```
\newcounter{linecount}

\newenvironment{normaltext}{%begin of env.
\begin{list}{%default label:
\arabic{linecount}%
}{% begin of list declarations:
\usecounter{linecount}%
%
\def\item{\let
\@tempignorespaces= \ignorespaces
\let \ignorespaces=\makeverseline
\@ifnextchar [{\@item }{\@noitemargtrue
\@item[\@itemlabel]}}%
%
\def\head{\let
\@tempignorespaces= \ignorespaces
\let \ignorespaces= \makeversehead
\@ifnextchar [{\@item }{\@noitemargtrue
\@item[\@itemlabel]}}%
%
\let\reset=\@ureset
%
```

```
\let\vl=\makeverselabel
%
}% end of list declarations and env
% find out total width for indent
\tw@totalleftmargin= \@totalleftmargin
\tw@linewidth= \linewidth
\indentdiff= \versewidth
\advance \indentdiff by \versesep
\advance \indentdiff by \rlevel\rwidth
\advance \indentdiff by \llevel\lwidth
\advance \indentdiff by \hangwidth
\adjustparshape
\setlength{\itemsep }{\z@}
\setlength{\parsep }{\z@}
\setlength{\parskip }{\z@}
}%end of newenv begin parameter
{% test if red and lev are reset:
\ifnum\llevel= \z@
\else \adjusterror {l}\fi
\ifnum\rlevel= \z@
\else \adjusterror {r}\fi
\end{list}%
}%end of newenv end parameter
```

4.2 Extensions to the item command

We now have to specify what should be done after we wrestled control from the `\@item` command. First we shall fulfil our pledge to restore `\ignorespaces`. Then we shall advance our special `\verselinecount` and give it a chance to produce the 'A B C ...' numbering, unless it is drowned by something hidden in the `#1` argument that will be executed next. Its main purpose is to process chapter and verse count and level information, as we shall soon see.

By `\noindent` we make sure that the new item is started before we produce our own label. We then use our little trick of macro substitution another time to determine what is to be done at the end of the `\makenormlabel` macro to be executed next, because the same macro is also used for our headlines, but with a different continuation.

To make sure that no harm is done to `\ignorespaces` in case our macro is called directly, we initialize `\@tempignorespaces`.

The counterpart of the `\reset` macro can be specified much easier. To make sure that our last item cannot be affected any more by the parameters to come, we issue a `\noindent` and then simply use a LaTeX 'hack' to execute whatever will be contained in the following parameter.

```
\newcommand{\makeverseline}[1]{\let
  \ignorespaces=\@tempignorespaces
  \advance\verselinecount by \@ne
  \def\verselinelabel{$^{\em
  \@Alph{\verselinecount
  }}$}#1% execute parameters
  \let\makenormlabelend=\ignorespaces
  \noindent
  \makenormlabel}

\let \@tempignorespaces= \ignorespaces

\def\@ureset{\noindent
  \@iden} %LaTeX hack
```

The macro for headlines will insert a chapter mark if one was specified before, otherwise the field will be left blank. There is a subtlety not obvious at first glance: a chapter printed here is inherited from the previous context, while a chapter starting at this line would print in bold through the `#1` parameter. Finally, a macro to read the text of the heading will be called.

```
\newcommand{\makeversehead}[1]{\let
  \ignorespaces=\@tempignorespaces
  \def\verselinelabel{{\em \kapitel
  \/}}#1% execute parameters
  \let\makenormlabelend=\readversehead
  \noindent
  \makenormlabel}

\newcommand{\readversehead}[1]{%
  {\em #1\/}\ignorespaces}
```

The `\makenormlabel` macro will make sure that we are in horizontal mode (cf. *The TEXbook* Ex. 13.1), then produce redaction indentation, follow it with the verse line label together with a separator congenial to the label separator of the standard list environment, and finally append speech level indentation. Though in most cases both types of indentation will be just blank space, we have introduced two macros that will allow control of what goes into the indented space. As a default for testing purposes, we use `\dotfill` and `\hrulefill`. But these macros could also be used to specify names for redactional levels or vertical lines to show the depth of speech embedding. Right adjustment will be achieved using the standard `\makelabel` macro.

In some rare cases our choice of basic lines will not coincide with the cutting of verses in the source. In such a case, we will use a `\vl{\v{b}}` sequence in the text and execute it as `\makeverselabel`. First,

the parameter is read, then the label is put into the running text and connected to what follows with a tie. The tie (~) will ensure that the label is not separated from the following text by a possible linebreak.

```
\newcommand{\makenormlabel }{\leavevmode
  \hbox to \rlevel\rwidth {\rfill}\hbox
  to \versewidth{%
\makeverselinelabel {\verselinelabel}}%
\hskip \versesep
\hbox to \llevel\lwidth{\lfill}%
\makenormlabelend}

\let\makeverselinelabel= \@mklab

\newcommand{\makeverselabel }[1]{#1%
\verselinelabel ~\ignorespaces}
```

4.3 Declarations

Now we come to the boring part of the whole thing, making declarations and setting initial values. It should be noted however, which of our variables are counts, which are dimensions, and which are macros.

The shape of our display is mainly influenced by the different types of indent lengths. `\versewidth` depends mainly on the maximum width of verse or chapter numbers to be displayed, but the other parameters should be chosen according to readability considerations. A good strategy seems to start with the most important value of `\lwidth`, which indents embedded speech. A value of `1em`, representing roughly the width of two lowercase letters, seems to be sufficient. Then `\hangwidth` can be set to half that value, so that embedded text can still be distinguished from continuation lines of the governing text. `\rwidth`, the indentation for redactional parts of the text, should be made wide enough to mark the difference from text embedding very clearly, but not so wide that lines become too narrow and the reading connection to the governing text goes astray. The fields of indentation will be blank normally; for testing purposes, however, they can be filled with something, e.g. dots or rules.

```
\newcount\rlevel
 \newcount\llevel
 \newcount\verselinecount
\newdimen\versewidth
  \newdimen\versesep
  \newdimen\rwidth
  \newdimen\lwidth
  \newdimen\hangwidth
  \newdimen\indentdiff
  \newdimen\tw@totalleftmargin
  \newdimen\tw@linewidth

% set initial values
\rlevel=\z@ \llevel=\z@
\verselinecount=\z@
  \versewidth=1.2em
  \versesep=\labelsep
\lwidth=1em
  \hangwidth=0.5\lwidth
  \rwidth=4.0\lwidth
\let\lfill=\@empty
  % \def\lfill{\hrulefill}
\let\rfill=\@empty
  % \def\rfill{\hfill $\mid$}
  % \def\rfill{\dotfill}
```

4.4 Parameter macros

Now we come to the macros that will appear inside the optional arguments of `\item` and `\head`. The verse macro will make sure that our verselinecount 'A B C ...' will be replaced by the verse number or the 'b' in the starting line of the verse and its second half after the *atnach*. Note that chapters are not treated in terms of counts but of macros. This allows them to contain also letters.

The control sequences to switch the two different kinds of levels are gained by redefining those used to enclose math displays in LaTeX. Parentheses `\(...\)` are used for speech levels, square brackets `\[...\]` are used for redaction levels. All these macros can take an optional argument giving the number of level steps to be taken. This argument might also be used in a later stage of development to identify redaction levels by name.

```
\def\versehalf{b} % Text after Atnach, Hebrew
\def\versenumber{\z@}
\def\verselinelabel{\@empty}
 % \v is an accent
\renewcommand{\v}[1]{\if #1\versehalf \def
  \verselinelabel{$^{\versehalf }$}\else
  \verselinecount= \@ne \def
  \versenumber{#1}\def
  \verselinelabel{$^{\versenumber }$}\fi}

\def\kapitel{\@empty}
\newcommand{\k}[1]{\def \kapitel{#1}\def
```

```
  \verselinelabel{{\bf \kapitel}}}
\renewcommand{\(}{\adjustcount {l}{\@ne}}
\renewcommand{\)}{\adjustcount {l}{\m@ne}}
\renewcommand{\[}{\def \@redlsign{+}%
\@ifnextchar [{\@redl }{\@noitemargtrue
  \@redl[\@ne]}}
\renewcommand{\]}{\def \@redlsign{-}%
\@ifnextchar [{\@redl}{\@noitemargtrue
  \@redl[\@ne]}}
\def\@redl[#1]{%
  \adjustcount{r}{\@redlsign #1}}
\renewcommand{\(}{\@switch {l}{+}}
\renewcommand{\)}{\@switch {l}{-}}
\renewcommand{\[}{\@switch {r}{+}}
\renewcommand{\]}{\@switch {r}{-}}
\newcommand{\@switch}[2]{%
  \@ifnextchar [{\@adjust{#1}{#2}}{%
    \@adjust {#1}{#2}[\@ne]}}
\def\@adjust#1#2[#3]{\adjustcount {#1}{#2#3}}

\newcommand{\adjustcount }[2]{\def
  \thiscount{\csname #1level\endcsname
  }\advance \thiscount by #2\ifnum
  \thiscount < \z@ \adjusterror {#1}\else
  \indentdiff= #2\csname #1width\endcsname
  \adjustparshape \fi}

\newcommand{\adjusterror }[1]{\message
  {**** wrong #1-level \the\thiscount in
  \normposition}%
  \csname #1level\endcsname= \z@}
\newcommand{\normposition }{list-line
  \thelinecount, Ch.\kapitel, V.\versenumber
  \@Alph{\verselinecount} on page \thepage}
```

4.5 Adjusting the parshape

Having executed all our level switching parameters, we can now specify what our paragraphs should look like. The main difference is, that our paragraphs will have a twofold shape; the first line is the same as in LaTeX lists, but the following ones are different, and we have chosen our variables accordingly.

```
\newcommand{\adjustparshape}{%
  \advance \tw@totalleftmargin by \indentdiff
  \advance \tw@linewidth by -\indentdiff
  \parshape \tw@
    \@totalleftmargin \linewidth
    \tw@totalleftmargin \tw@linewidth }
```

5 Examples

Now that we have understood the macros, we are going to explore some of their possibilities in the following examples.

5.1 Showing Redaction

If the unity of a text is disputed, we can show which parts of the text are due to redactional work. For our text *Jeremiah 45*, a corresponding hypothesis has been formulated by *Winfried Thiel.*[10] It is not our concern here to discuss the validity of his assumptions, but to show in one clear display (Fig. 3), what otherwise has to be gleaned from several pages of text. The translation is ours and done according to the principles of *distinktives Übersetzen*, which try to make visible the structure of the underlying Hebrew as far as possible.

5.2 The Greek text of Jeremiah 45

To display our text in Greek[11] (Fig. 4), we need no more than standard TeX.[12] There are some restrictions, however. First, we have to read this kind of Greek from input, since it implies redefining of category codes.[13] Second, there is no automatic hyphenation, though explicit hyphens can be specified. If columns become rather narrow, it will often look better and also save a lot of `underfull box` messages to use a `raggedright` environment.

5.3 A synopsis of the Greek text and a translation of Jeremiah 45

Our next step will be to combine the Septuagint text with its translation (Fig. 5), using the `minipage` environment.

Though this approach makes it easy to produce two columns, it does not guarantee a correct alignment of base lines, because extension lines due to automatic line breaking may be different for the two

[10] Winfried Thiel: Die deuteronomistische Redaktion von Jeremia 26–45. Mit einer Gesamtbeurteilung der deuteronomistischen Redaktion des Buches Jeremia. *WMANT 52. 1981.*

[11] In the Septuagint (LXX), it is placed at the end of chapter 51, thus bearing witness to a different text tradition.

[12] The macros to perform typesetting Greek are described in my article in note 8.

[13] Consequently, the `\fbox` macro to draw the frame around the figure has to be split into a begin and end part. In other cases it will be convenient to use the method of semi-parameters known from PLAIN footnotes and described in more detail in my article *Chapter mottos and optional semi-parameters for TeX and LaTeX. TUGboat 7,3 (1986) 177-185.*

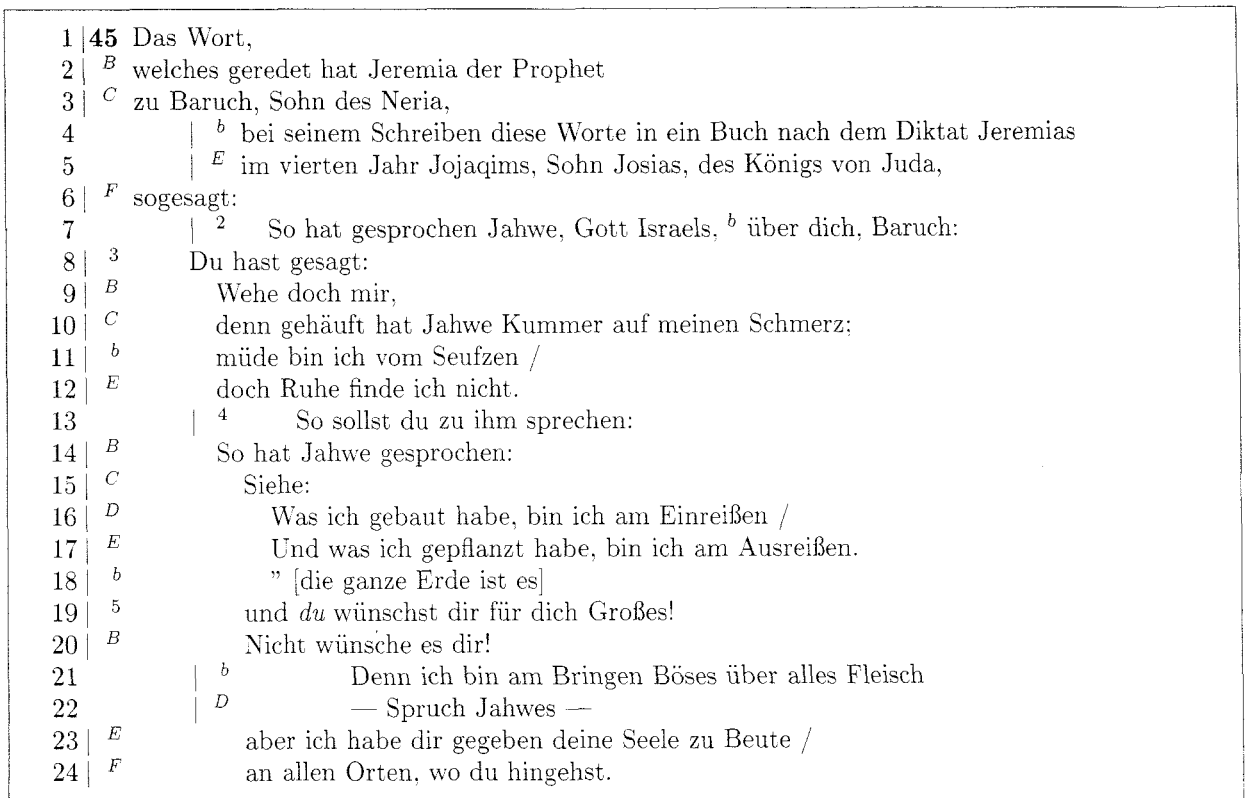


Figure 3: Vorlage und Redaktion in Jer 45 nach Thiel.

versions. The normal approach would be to regroup the synoptic texts so that the items of the different columns are presented row-wise. Though this would allow the use of `tabular`, the price would be breaking texts into pieces. On the other hand any almost automatic process of combining sequential texts will cost a lot of tricky programming and so will only be worthwhile for a larger project.

6 A verbatim of the Greek text

To show how the Greek text in Fig. 4 and Fig. 5 was produced, we give a verbatim of the corresponding input.

```
\begin{normaltext}
\item{\v{31}} {\greek \,<O lo1gov, }
\item{}  {\greek o9n e>la1lhsen Ieremiav
          o< profh1thv}
\item{}  {\greek pro3v Baroyx yi<o3n Nhrioy,}
\item{\v{b}}  {\greek o7te e4grafen toy3v
          lo1goyv toy1toyv e>n tw2j bibli1wj
          a>po3 sto1matov Ieremioy}
\item{}  {\greek e>n tw2j e>niaytw2j tw2j
          teta1rtwj tw2j Iwakim yi<w2j Iwsia
          basile1wv Ioyda}
\item{}
\item{\v{32}\()  {\greek Oy7twv ei5pen ky1riov}
 \v1{\v{b}}     {\greek e>pi3 soi1, Baroyx}
\item{\v{33}\() {\greek \,7Oti ei5pav}
\item{\() {\greek Oi4mmoi oi4mmoi,}
\item{} {\greek o7ti prose1qhken ky1riov
          ko1pon e>pi3 po1non moi,}
\item{\v{b}} {\greek e>koimh1qhn e>n
          stenagmoi2v,} /
\item{} {\greek a>na1paysin oy>x ey8ron,}
\item{\v{34}\)} {\greek ei5pon ay>tw2j}
\item{\() {\greek Oy7twv ei5pen ky1riov}
\item{\() {\greek \,>Idoy3}
\item{\() {\greek oy9v e>gw3 w>jkodo1mhsa,
          e>gw3 kaqairw2,} /
\item{}  {\greek kai3 oy9v e>gw3 e>fy1teysa,
          e>gw3 e>kti1llw;}
\item{\v{b}} %('\,' [die ganze Erde ist es])
\item{\v{35}\)}  {\greek kai3 sy3 zhtei2v
          seaytw2j mega1la?}
\item{}  {\greek mh3 zhth1shjv,}
\item{\v{b}}  {\greek o7ti i>doy3 e>gw3
          e>pa1gw kaka3 e>pi3 pa2san sa1rka,}
```

```
1  31 ῾Ο λόγος,
2  B  ὃν ἐλάλησεν Ιερεμιας ὁ προφήτης
3  C  πρὸς Βαρουχ υἱὸν Νηριου,
4  b  ὅτε ἔγραφεν τοὺς λόγους τούτους ἐν τῷ βιβλίῳ ἀπὸ στόματος Ιερεμιου
5  E  ἐν τῷ ἐνιαυτῷ τῷ τετάρτῳ τῷ Ιωακιμ υἱῷ Ιωσια βασιλέως Ιουδα
6  F
7  32    Οὕτως εἶπεν κύριος b ἐπὶ σοί, Βαρουχ
8  b  b    ῞Οτι εἶπας
9  C         Οἴμμοι οἴμμοι,
10 D         ὅτι προσέθηκεν κύριος κόπον ἐπὶ πόνον μοι,
11 b         ἐκοιμήθην ἐν στεναγμοῖς, /
12 F         ἀνάπαυσιν οὐχ εὗρον,
13 34      εἶπον αὐτῷ
14 B         Οὕτως εἶπεν κύριος
15 C           ᾽Ιδοὺ
16 D             οὓς ἐγὼ ᾠκοδόμησα, ἐγὼ καθαιρῶ, /
17 E             καὶ οὓς ἐγὼ ἐφύτευσα, ἐγὼ ἐκτίλλω·
18 b
19 35            καὶ σὺ ζητεῖς σεαυτῷ μεγάλα;
20 B             μὴ ζητήσῃς,
21 b             ὅτι ἰδοὺ ἐγὼ ἐπάγω κακὰ ἐπὶ πᾶσαν σάρκα,
22 D             λέγει κύριος,
23 E             καὶ δώσω τὴν ψυχήν σου εἰς εὕρεμα /
24 F             ἐν παντὶ τόπῳ, οὗ ἐὰν βαδίσῃς ἐκεῖ.
```

Figure 4: The Greek text of Jer 45 (Jer 51,31–35 LXX).

```
1  31 ῾Ο λόγος,                                  1  45 Das Wort,
2  B  ὃν ἐλάλησεν Ιερεμιας ὁ προφήτης            2  B  welches geredet hat Jeremia der Prophet
3  C  πρὸς Βαρουχ υἱὸν Νηριου,                   3  C  zu Baruch, Sohn des Neria,
4  b  ὅτε ἔγραφεν τοὺς λόγους τούτους ἐν τῷ      4  b  bei seinem Schreiben diese Worte in ein
      βιβλίῳ ἀπὸ στόματος Ιερεμιου                    Buch nach dem Diktat Jeremias
5  E  ἐν τῷ ἐνιαυτῷ τῷ τετάρτῳ τῷ Ιωακιμ υἱῷ    5  E  im vierten Jahr Jojaqims, Sohn Josias,
      Ιωσια βασιλέως Ιουδα                            des Königs von Juda,
6  F                                             6  F  sogesagt:
7  32    Οὕτως εἶπεν κύριος b ἐπὶ σοί,           7  2     So hat gesprochen Jahwe, Gott
         Βαρουχ                                          Israels, b über dich, Baruch:
8  b  b    ῞Οτι εἶπας                           8  3     Du hast gesagt:
9  C         Οἴμμοι οἴμμοι,                      9  B        Wehe doch mir,
10 D         ὅτι προσέθηκεν κύριος κόπον         10 C        denn gehäuft hat Jahwe Kummer
              ἐπὶ πόνον μοι,                                  auf meinen Schmerz;
11 b         ἐκοιμήθην ἐν στεναγμοῖς, /          11 b        müde bin ich vom Seufzen /
12 F         ἀνάπαυσιν οὐχ εὗρον                12 E        doch Ruhe finde ich nicht.
```

Figure 5: The Greek text of Jer 45 (Jer 51,31-35 LXX).

```
\item{}  {\greek le1gei ky1riov,}
\item{}  {\greek kai3 dw1sw th3n uyxh1n soy
          ei>v ey7rema} /
\item{}  {\greek e>n panti3 to1pwj, oy8 e>a3n
          badi1shjv e>kei2.}
\reset{\)[4]}
\end{normaltext}
```

7 Conclusion

Perhaps it has become clear from our examples that displayed texts from ancient sources call for attention and handling similar to that of mathematical formulas, being the core and the aim of exegetic efforts. Just as with formulas, displaying them according to accepted rules will help to bring out their internal structure, thus contributing to their readability.

Because of the specific rules to be followed, it is not a trivial task to get the shape of such texts right. But while the formatting of mathematics is built into TEX, ours has to be done with macros.

Alas, our programming betrays a splendid inconsistency of mixing between TEX and LATEX commands. It is really bad style and should encourage others to do better. But what might be more interesting is our somewhat peculiar use of optional arguments to perform unforeseen actions on our layout, a feature that still waits to be explored to its full depth.

While other contributions like the *chapter mottos* mentioned above fall into the general area of typesetting, our problem is an example for the field of *philological* typesetting. Interest in such applications has grown recently, and though there are some severe restrictions in TEX, e.g. with respect to the reverted formatting necessary for semitic languages like Hebrew,[14] in general it seems to me to be a reliable base to join efforts in further development. That is why we decided at the Strasbourg conference, this past summer, to arrange for the formation of a TEX philology group.[15] Taking the present article as an example for philologic activities, I should like to stress two axioms on which it is based:

Axiom 1 *New developments should be published rapidly and with fully documented source.*

The slow progress of many projects in the field of the Bible and the computer, in my opinion, is due to the lack of any publishing of that kind. Now we have got the opportunity to do a better job.

Axiom 2 *Development should be based on or at least be compatible with* LATEX.

The use of LATEX is a good way to free oneself from many problems of general formatting that are already solved there in a clear and consistent way. LATEX programming philosophy also contains some useful techniques that may be borrowed for other purposes, thus speeding up macro development. Finally, LATEX seems to be the only macro package that has a good chance to emerge as *the* standard, sort of a counterpart to IBM's General Markup Language.

Axiom 3 *Development in philological typesetting should be based on firm linguistic ground.*

It is the aim of our article to give an example for the close relationship between exegetical work and linguistics on the one hand and linguistics and philological typesetting on the other.

---

[14] Rf. to note 8.

[15] Rf. *Barbara Beeton / R. W.: Towards a TEX Philology Group. TUGboat 7,3 (1986) 132-133.*

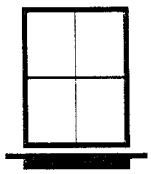
Problems

Editor's note: It seems that the time is ripe for certain ideas to pop out of the woodwork, from all directions. The two items below appeared independently, without the authors being aware that anyone else was working on the problem, although it's clear from what they say that other people are *interested*. By mutual consent, solutions to the problem are deferred until the next issue. We can assure you, however, that the solutions operate from different premises, and comparison of the macros (and perhaps using them to create a new, general, solution) should be interesting and instructive.

TEX Does Windows: A Progress Report

Alan Hoenig

We New Yorkers are a contentious lot. That's why people often approach me, chip on shoulder, to tout some mediocre typesetting program that can set windows within paragraphs. It's also why, I suppose, I rise to the bait and tout TEX back. Most of these rival programs are incapable of right justification, so what difference their ability to do windows? (I call the rectangle or other shape around which a paragraph is set a *window*.) Nevertheless, it rankles. Why can't TEX do windows? ¶Or can it? After all, TEX's commands mimic the activities a human compositor goes through to perform typesetting, and a human could certainly typeset text around windows, so why can't TEX do the same? I finally came upon one way to do it. My method is not altogether general — that's why this is only a progress report — but it does work, horizontally centering a window of some given width within a paragraph. In the next issue, I will describe these ideas and their TEX implementations, as well as the main ideas behind the macros.



Problem for a Saturday Morning

Donald E. Knuth
Stanford University

This puzzle was suggested to me by Sape Mullender, of the Centre for Mathematics and Computer Science in Amsterdam. He told me his belief that "the general design of TEX is better than that of *troff*, but the real guru can make *troff* do things that you could never do in TEX." As an example, he showed me a page on which *troff* had typeset a picture in the middle of a paragraph, with the text going around the picture. "It's not pretty, but it can be done, and that's what counts," he said. Well, I have to admit that I didn't think of a simple solution until the next Saturday morning; and I didn't finish debugging it until that Saturday afternoon. Can you guess how I typeset the paragraph you're now reading? (The answer will appear in the next issue. It doesn't demonstrate the superiority of TEX to *troff*, but it does have some interesting and instructive features.)

Queries

Editor's note: When answering a query, please send a copy of your answer to the TUGboat editor as well as to the author of the query. All answers will be published in the next issue following their receipt.

Responses to previous queries appear elsewhere in this issue: Indexing with LaTeX (Jim Ludden, Vol. 7, No. 2, page 111), see page 62. Form letters (John Lee, Vol. 7, No. 3, page 187), see pages 53 and 60 for PLAIN and LaTeX approaches.

APA Style in LaTeX

I have a specific question about LaTeX that I hope someone can answer for me. I am a cognitive psychologist using LaTeX to write my papers in APA (Americal Psychological Association) style. One of the requirements of APA style is a running head of two or three words in the upper right hand corner of the page, and then double-spaced immediately below that, the page number. For example:

```
                                    Running Head

                                                16
```

I have played around with setting my own headings for pagestyle, but can only seem to move things around on the same line.

Jennifer L. Dyck
California State University, Fresno

Chemical Formulae

I am working with the Chemistry Department at Smith College, trying to develop or unearth any TeX implementations of formulae typical of chemistry texts. I would be delighted to hear from anyone working on this!

Judy Hawkins
Smith College, Northampton, MA
JHawkins@Smith.Bitnet

Ancient Non-Roman Languages

Has anyone included Hebrew, Greek, Akkadian, Ugaritic, and/or Aramaic text in a TeX document? If so, please let me know how. (First priority is finding font files and TeX definitions; then we can fine-tune spacing, punctuation, and hyphenation.)

James R. Celoni, S.J.
Santa Clara, CA
Celoni@score.stanford.edu

Editor's note: For Greek, using the admittedly non-text-like Computer Modern math greek, see the article by Reinhard Wonneberger that begins on page 63. Requests for Hebrew fonts also turn up regularly, but we have not yet heard of any full alphabet font. The article by Donald Knuth and Pierre MacKay, beginning on page 14, discusses some problems that will be relevant once such a font is available.

International Phonetic Alphabet

Are there any TeX users who have made sets of IPA (International Phonetic Alphabet) characters for use with TeX, or who are interested in this question (and waiting for METAFONT), or who have worked on other particular problems of linguistic text (e.g. macros for aligning word-for-word interlinear translations with texts in other languages)?

Boyd Michailovsky
CNRS/LACITO, Paris

Line Numbering

How would one provide automatic line numbering for every fifth (5, 10, 15, 20, etc.) line in the left-hand margin? I would also like to be able to start and stop such numbering in the middle of a page. These features are almost essential in the preparation of patent applications.

Lawrence D. Cutter
121 Mohawk Drive
Schenectady, NY 12303

Dictionary Formatting

The Warlpiri Dictionary Project has a large document set out with R20 formatting commands. We wish to convert this to LaTeX or TeX.

(1) Does anyone know of a utility which would do the conversion?

(2) Does anyone know of a (bi-lingual) dictionary style macro package? This would recognise an entry with its repeating internal structure as the building unit, get the running heads right, possibly allow indexing of the English glosses, and so on.

David Nash
Center for Cognitive Science, MIT
lex.nash@SPEECH.MIT.EDU

Printing Out Selected Pages

As new users of TeX, we have lots of questions, but one which is particularly "niggling" is: Is there some way by which one can select only a page (or pages) of printout? In long manuscripts, there is often a need for only a few pages that have been corrected, rather than reprinting the whole paper. No one here has been able to figure it out, other than creating a separate file of just those few pages needed.

Helen S. Horstman
Lowell Observatory, Flagstaff, AZ

Editor's note: There are two ways of approaching this: through TeX, and post-TeX.

Many output drivers offer the ability to print selected pages from a DVI file, usually by specifying a starting/ending page range, or a starting page number and number of pages desired. This technique has the disadvantage that the entire manuscript must be processed by TeX (however, page numbering will be the same as in the whole job, which is not automatically true if a separate partial file is created).

Some TeX implementations accept interruptions from the terminal while TeX is running, presenting a ? prompt which can be answered by x to terminate the job cleanly, i.e. with a printable DVI file. (For example, VAX/VMS implementations can usually be interrupted by ^C; the TOPS-20 implementation can be interrupted by ^G. A carriage return will permit an interrupted job to continue with no adverse effects.) Use of this feature may be appropriate if the changed pages are near the beginning of the manuscript.

Some macros intended to generate only selected material from a larger TeX input file have recently appeared in TeXhax. We will try to track these down for inclusion in the next issue.

Letters

Wanted: Help for Beginners

To the Editor:

As a new member and a beginner at TeX, I find my membership of TUG useful. However, I feel that TUGboat makes almost no attempt to help beginners. We're not all wizards! More introductory articles please!

Raymond A. Ryan
Department of Mathematics
University College Galway
Galway, Ireland

Editor's note: We're glad that you do find TUG useful, and will continue to try hard to keep it that way.

On the subject of TUGboat, this editor has said many times that she will publish (within the bounds of relevance and decency) whatever is submitted. Three members of the Editorial Committee are particularly interested in the problems of new users and "low-level" users: Jackie Damrau of the University of New Mexico (LaTeX), Maureen Eppstein of Stanford University (applications in Plain TeX), and Laurie Mann of Stratus Computer (training issues). Articles or other items in these areas should be submitted through them. (The traffic has been very light — if we were on a desert road waiting for help, we would long since have perished of thirst.)

TUGboat is *your* journal! Help make it a useful one!

More Hyphenation Exceptions

To the Editor:

I recently received the November issue of TUGboat, and I was glad to find the Hyphenation Exception Log on page 121. I would like to contribute my own findings:

TEX	"Correct"
`tele-g-ra-pher`	te-leg-ra-pher
`schot-tis-che`	schot-tische

From the exception list and the actual hyphenations of TEX I gather that all syllable divisions are legal end-of-line breaks. This may not always be the case. I have collected some hyphenations that I think may be unacceptable in well typeset texts. I have indicated these with a ? in the examples below.

a) Words ending in -cle, -tle, -ple, -ble, -gle, -dle, etc.

   trea?cle lit?tle peo?ple am?ble sin?gle trea?dle

   (*Chicago Manual of Style*, 6.33*ff* discourages it; Merriam-Webster's *Third New International Dictionary*, vol. I, p. 22a, 11.3, seems to discourage trea-dle)

b) Words in which a single vowel constitutes a syllable.

   crit?i-cism lig?a-ture phys?i-cal sep?a-rate prej?u-dice anal?y-sis

   (*CMS*, 6.35 discourages it; *MWTNID* unclear. Anal-ysis should definitely be discouraged!)

I first reacted against these hyphenations because they differ from the (British) hyphenation practice I am used to. As it seems that American practice discourages these breaks, too, perhaps there is reason to revise the hyphenation patterns of TEX.

Anders Thulin
ContextVision AB
Teknikringen 1
S-583 30 Linköping, Sweden

Editor's note: This letter was mislaid, as is painfully obvious from the issue reference, but the information it contains is on a subject that is always up-to-date. The Editor hopes that Mr. Thulin will accept her sincere apology. The latest iteration of the Exception Log appeared in vol. 7, no. 3, page 145. We plan to make this a regular feature of the last issue each year, and solicit additions.

New Fonts for Mathematics?

To the Editor:

I think it is high time we retired math italic and use instead a font designed especially for mathematics. It needs to be highly readable (which italics isn't) and yet be very different from the Times typeface.

Without TEX I have been using the LaserWriter's Courier font for mathematics. Its thin strokes make it easily distinguishable from Times and it is quite readable (if only because we've been forced for a long time to read it in letters and first drafts of documents). No longer is it necessary for me to change fonts for things like `3`, `det`, and `cos`. With TEX it is a different story. Can anyone tell me how to replace math italic with another font (including its digits) without losing the special symbols or getting punctuation out of order?

William A. McWorter
Mathematics Department
Ohio State University
Columbus, OH 43210

Editor's note: Since the Courier font referred to by Mr. McWorter does not exist on our typesetter, we have substituted the closest font available. This is the Computer Modern `tt` font, which, although heavier than Courier, is also a monospace or fixed-pitch font.

In the particular examples cited, traditional math typography would use roman, not italic, type, and indeed it can be difficult to distinguish such an expression when it appears in the middle of text. It is also very difficult to distinguish the mathematical term "*a*" from the English article "a" in the context of a theorem, traditionally set all in italic. Some experimental styles have used slanted, rather than italic, type to make such distinctions clearer; one might also note that Computer Modern math italic was designed with letter shapes slightly different from those of text italic for presumably the same reason.

The American Math Society, when it commissioned Hermann Zapf to design the Euler Fraktur and script fonts, also received designs for an upright "cursive" font, intended to resemble notation handwritten on a blackboard. One purpose of this design was to avoid the sometimes severe problems associated with applying accents and indices (sub- and superscripts) to italic letter symbols. Some experiments have been performed using this font to set math, but none have yet reached publication. The feeling has been that most mathematicians are

quite conservative regarding the appearance of math journals, and might not accept the cursive notation readily. Perhaps Mr. McWorter is a harbinger of change in this regard.

Here is a small sample of the Euler cursive, with the same material in Computer Modern math italic for comparison.

$$\mathsf{x}^2 + \mathsf{y}^2 = \mathsf{z}^2 \qquad x^2 + y^2 = z^2$$

$$\mathsf{f}^*(\mathsf{x}) \cap \mathsf{f}_*(\mathsf{y}) \qquad f^*(x) \cap f_*(y)$$

$$\mathsf{D} \sim \mathsf{p}^\alpha\, \mathsf{M} + \mathsf{l} \qquad D \sim p^\alpha\, M + l$$

$$\frac{\mathrm{d}\mathsf{y}}{\mathrm{d}\mathsf{x}} = \mathsf{f}(\mathsf{x}\,,\,\mathsf{y}) \qquad \frac{dy}{dx} = f(x,y)$$

$$\frac{\partial \varphi}{\partial \mathsf{t}} = \mathsf{f}(\varphi\,,\,\mathsf{t}) \qquad \frac{\partial \varphi}{\partial t} = f(\varphi, t)$$

Announcements

Workshop on Font Design Systems

A workshop on Font Design Systems is being organized by INRIA to be held in Sophia-Antipolis (near Nice) on 18–19 May 1987. Papers have been solicited on the following topics:

- High, medium or low resolution font systems
- Character design algorithms
- Interactive systems
- Experiments with METAFONT, etc.
- Exotic alphabets
- 3D characters
- Logo design

The deadline for submitting papers has passed; for details on attending the workshop, please contact

Jacques André
IRISA/INRIA–Rennes
Campus de Beaulieu
F-35042 Rennes Cedex, France
`!inria!irisa!jandre`

or

Moncef Mlouka
INRIA–Sophia Antipolis
Avenue Emile Hugues
F-06565 Valbonne Cedex, France

Special Course, Spring Quarter 1987
TEX: The Program:
A case study in software design

Donald Knuth
Stanford University

This special course will be given by Donald Knuth at Stanford during the Spring Quarter, 1987. It will consist of about twenty 75-minute lectures on Tuesdays and Thursdays, 1:15–2:30 p.m., starting in early April and continuing until early June. The course designation is CS 349; this is a 3-unit course.

The implementation of TEX will be discussed as an example of the design and documentation of a medium-size software system. Also discussed will be the WEB system of structured documentation. Knowledge of the TEX and Pascal languages is a prerequisite. Students will learn enough about the innards of TEX to make extensions to the system. The text for the course will be *TEX: The Program*.

As with Knuth's previous TEX-related courses, this course will be transmitted to remote locations via the Stanford educational network, and it will be videotaped.

Videotapes of the course

If there is enough interest among the organizations represented in TUG's membership, TUG can make a special arrangement with Stanford University to lease these tapes for a year or more, with the right to sublease them. The exact details of how this would work and how much it would cost cannot be determined until TUG has an indication of the interest in setting up such an arrangement. A very rough estimate, however, would be as little as $1,000 or up to $2,000, again depending on the number of participants. The cost of leasing the tapes directly from Stanford would be at least several thousand dollars higher.

Any organization interested in participating in a TUG-sponsored lease arrangement should contact the TUG office by May 15, 1987; call 401-272-9500, ext. 232.

Call for Papers: Electronic Publishing, Document Manipulation and Typography

Nice, France
April 20–22, 1988

An international conference on Electronic Publishing, Document Manipulation and Typography will be held at Nice, France, on April 20–22, 1988. The Conference is being organized by INRIA, France, in association with a number of sponsors. This conference may be considered as a successor to the EP86 conference organized at the University of Nottingham, England, in April 1986, by the British Computer Society.

The conference will cover all aspects of computer document preparation, text processing, and printing. It will include topics such as document design, authoring systems, electronic publishing, digital typography, and it will definitely be oriented to new ideas and techniques on such matters. Papers, which should present original research work or give a comprehensive survey of a particular area, are invited on any new topic related to document processing, including (but not limited to) the following:

- Document structures (analysis and recognition)
- Document editors or formatters, integration of text, graphics, and images
- Markup languages and translation from one to another one
- Computer-based and dynamic documents
- Page description languages
- Interfaces with other software
- Expert systems for editing
- Specific documents (mathematics, chemistry, humanities, music, exotic languages, ...)
- Electronic publishing — applications and techniques
- Linguistic approaches and semantic structures of text

The chairman of the conference is Jacques André (IRISA/INRIA) and the vice-chairman is Brian Kernighan (Bell Laboratories). The program committee also includes Patrick Baudelaire (TANGRAM), Richard Beach (Xerox PARC), Charles Bigelow (Stanford University), David Brailsford (University of Nottingham), Heather Brown (University of Kent), Giovanni Coray (EPFL), R. W. Davy (Chelgraph Ltd.), Richard Furuta (University of Maryland), James Gosling (Sun Microsystems), Vania Joloboff (Bull/INRIA), Peter King (University of Manitoba), Dario Lucarella (Università di Milano), Pierre MacKay (University of Washington), Robert Morris (Interleaf/University of Massachusetts), J. Nievergelt (University of North Carolina), Vincent Quint (INRIA/IMAG), Brian Reid (DEC Western Research Center), Alan Shaw (University of Washington), and Hans Van Vliet (CWI).

Main deadlines

now	Ask to be placed on the mailing list using the form below.
July 31, 1987	Papers to be received by the Program Committee Chairman.
Oct 31, 1987	Notification of acceptance and mailing of instructions for preparation of the final paper.
Jan 31, 1988	Final paper to be received by the Proceedings Editor (Conference proceedings will be available at the conference).

To be placed on the mailing list for this conference, please supply the following information to:

Jacques André
IRISA/INRIA EP88
Campus de Beaulieu
F-35042 Rennes Cedex
FRANCE

or send relevant information by electronic mail to the Usenet address:

...mcvax!inria!irisa!jandre

I am interested in
[ ] receiving further information
[ ] submitting a paper
[ ] attending the conference
[ ] exhibiting
Name:
Address:

Calendar

1987

University of Illinois, Chicago

Mar 23 – 25 Beginning TEX (intensive)

Mar 26 – 27 Intermediate TEX (intensive)

* * * * *

Mar 25 – Apr 29 California State University, Northridge
Beginning TEX – Wednesdays, 6:00–9:30 p.m.

Mar 30 – Apr 3 University of Delaware, Newark
Intensive Beginning/Intermediate TEX

Apr 3 Journée "TEX et les Sciences Humaines", Paris, France; sponsored by GUT : Groupe francophone des Utilisateurs de TEX. For information, contact Jacques André, Rennes, France, 99-364815.

Apr – Jun Stanford University, Palo Alto
TEX: The program; A case study in software design – Tuesdays and Thursdays, 1:15–2:30 p.m. (see announcement, page 77)

Apr 15 Deadline for TEX copy for preprints, 1987 TUG Meeting: TEX for the Humanities

May 6 – Jun 10 California State University, Northridge
Intermediate TEX – Wednesdays, 6:00–9:30 p.m.

May 18 TUGboat Volume 8, No. 2: Deadline for submission of manuscripts

May 18 – 19 INRIA Workshop on Font Design Systems, Sophia-Antipolis, France (see announcement, page 77)

Texas A & M University, College Station

May 18 – 20 Beginning TEX (intensive)

May 21 – 22 Intermediate TEX (intensive)

University of New Mexico, Albuquerque

Jun 8 – 10 Beginning TEX (intensive)

Jun 11 – 12 Intermediate TEX (intensive)

Vanderbilt University, Nashville

Jun 8 – 12 Beginning TEX

Jun 8 – 12 Intensive Beginning/Intermediate TEX

Jun 15 – 19 Advanced TEX/Macro Writing

Northeastern University, Boston

Jun 22 – 26 Beginning TEX

Jun 22 – 26 Intensive Beginning/Intermediate TEX

University of Exeter, England

Jul 6 – 10 Intensive Beginning/Intermediate TEX

Jul 13 – 17 Advanced TEX/Macro Writing

University of New Mexico, Albuquerque

Jul 13 – 17 Advanced TEX/Macro Writing

Stanford University, Palo Alto

Jul 27 – 31 Advanced TEX/Macro Writing

Rijksuniversiteit Groningen, The Netherlands

Jul 20 – 24 Intensive Beginning/Intermediate TEX

Jul 27 – 31 Advanced TEX/Macro Writing

* * * * *

Jul 27 – 31 SIGGRAPH, Anaheim, Calif.. For information, contact the ACM Conference Office, Chicago, Ill.; 312-644-6610

Aug 4 – 7 Sixth International Conference on Mathematical Modelling, Washington University, St. Louis, Mo.; TEX exhibits and presentations.

Status as of 25 February 1987

TEX Users Group 1987 Conference
University of Washington, Seattle

Aug 17–21 Beginning TEX

Aug 17–21 Intensive Beginning/Intermediate TEX

Aug 24–26 **TUG Annual Meeting:**

Aug 24 General program; introduction for new TEXers

Aug 25–26 Technical program of refereed papers: TEX for the Humanities

Aug 25–26 General-interest technical program

Aug 26 Deadline for machine-readable copy of refereed papers for Proceedings

Aug 27–28 Short course: Macro Writing

* * * * *

Aug 28–Sep 4 American Chemical Society, National Meeting, New Orleans, LA; two symposia on problems of journal publication: "Methods for the electronic submission of manuscripts for publication", and "Problems and solutions in the generation of scientific manuscripts". For information on the symposia, contact Peter Lykos, Illinois Institute of Technology, 312-567-3430, Bitnet: **chempgl@iitvax**.
TEX exhibits and presentations.

Sep 14–18 University of Illinois, Chicago
Intermediate TEX
Advanced TEX/Macro Writing

University of Bergen, Norway

Sep 21–25 Intensive Beginning/Intermediate TEX

Sep 28–Oct 2 Advanced TEX/Macro Writing

* * * * *

Sep 28–Oct 1 Conference on Electronic/Desktop Publishing, San Francisco, Calif. For information, contact National Computer Graphics Association, Fairfax, Va.; 703-698-9600

Oct 9 TUGboat Volume 8, No. 3: Deadline for submission of manuscripts (tentative)

Oct 19–22 Protext IV, Boston, Mass.
TEX Seminar, Monday, Oct. 19

1988

Jan 5–9 Joint Mathematics Meeting, Atlanta, Ga. TEX Workshop, Tuesday, Jan. 5

Apr 20–22 International Conference on Electronic Publishing, Document Manipulation and Typography, Nice, France (see announcement, page 78)

For additional information on the special program, TEX for the Humanities, being offered at the 1987 annual meeting, see TUGboat vol. 7, no. 3, page 131. For additional information on other events listed above, contact the TUG office (401-272-9500, ext. 232) unless otherwise noted.

Institutional Members

Addison-Wesley Publishing Company, *Reading, Massachusetts*

The Aerospace Corporation, *El Segundo, California*

Allied-Signal Canada, Inc., *Mississauga, Ontario, Canada*

American Mathematical Society, *Providence, Rhode Island*

ArborText, Inc., *Ann Arbor, Michigan*

ASCII Corporation, *Tokyo, Japan*

California Institute of Technology, *Pasadena, California*

CALMA, *Sunnyvale, California*

Calvin College, *Grand Rapids, Michigan*

Canon, Inc., Office System Center, *Tokyo, Japan*

Carleton University, *Ottawa, Ontario, Canada*

CDS/WordWorks, *Davenport, Iowa*

Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*

Centro Internacional De Mejoramiento De Maiz Y Trigo (CIMMYT), *México, D.F., Mexico*

City University of New York, *New York, New York*

College of St. Thomas, *St. Paul, Minnesota*

Columbia University, Center for Computing Activities, *New York, New York*

COS Information, *Montreal, P. Q., Canada*

Data General Corporation, *Westboro, Massachusetts*

Digital Equipment Corporation, *Nashua, New Hampshire*

Dowell Schlumberger Inc., *Tulsa, Oklahoma*

Edinboro University of Pennsylvania, *Edinboro, Pennsylvania*

Electricité de France, *Clamart, France*

Environmental Research Institute of Michigan, *Ann Arbor, Michigan*

European Southern Observatory, *Garching bei München, Federal Republic of Germany*

Ford Aerospace & Communications Corporation, *Palo Alto, California*

Försvarets Materielverk, *Stockholm, Sweden*

FTL systems Incorporated, *Toronto, Ontario, Canada*

General Motors Research Laboratories, *Warren, Michigan*

Geophysical Company of Norway A/S, *Stavanger, Norway*

Grumman Corporation, *Bethpage, New York*

GTE Laboratories, *Waltham, Massachusetts*

Hart Information Systems, *Austin, Texas*

Hartford Graduate Center, *Hartford, Connecticut*

Hewlett-Packard Co., *Boise, Idaho*

Hobart & William Smith Colleges, *Geneva, New York*

Hutchinson Community College, *Hutchinson, Kansas*

Humboldt State University, *Arcata, California*

IBM Corporation, Scientific Center, *Palo Alto, California*

Illinois Bell Telephone, *Chicago, Illinois*

Illinois Institute of Technology, *Chicago, Illinois*

Imagen, *Santa Clara, California*

Institute for Advanced Study, *Princeton, New Jersey*

Institute for Defense Analyses, Communications Research Division, *Princeton, New Jersey*

Intergraph Corporation, *Huntsville, Alabama*

Intevep S. A., *Caracas, Venezuela*

Iowa State University, *Ames, Iowa*

Istituto di Cibernetica, Università degli Studi, *Milan, Italy*

Kuwait Institute for Scientific Research, *Safat, Kuwait*

Los Alamos National Laboratory, University of California, *Los Alamos, New Mexico*

Marquette University, *Milwaukee, Wisconsin*

Massachusetts Institute of Technology, Artificial Intelligence Laboratory, *Cambridge, Massachusetts*

Mathematical Reviews, American Mathematical Society, *Ann Arbor, Michigan*

Max Planck Institute Stuttgart, *Stuttgart, Federal Republic of Germany*

McGill University, *Montreal, Quebec, Canada*

McGraw-Hill, Inc., *Englewood, Colorado*

National Research Council Canada, *Ottawa, Ontario, Canada*

New Jersey Institute of Technology, *Newark, New Jersey*

Northeastern University, Academic Computing Services, *Boston, Massachusetts*

Online Computer Library Center, Inc. (OCLC), *Dublin, Ohio*

Pennsylvania State University, Computation Center, *University Park, Pennsylvania*

Personal TEX, Incorporated, *Mill Valley, California*

Purdue University, *West Lafayette, Indiana*

QMS, Inc, *Mobile, Alabama*

Queens College, *Flushing, New York*

Research Triangle Institute, *Research Triangle Park, North Carolina*

RE/SPEC, Inc., *Rapid City, South Dakota*

Ruhr Universität Bochum, *Bochum, Federal Republic of Germany*

Rutgers University, Hill Center, *Piscataway, New Jersey*

St. Albans School, *Mount St. Alban, Washington, D.C.*

Sandia National Laboratories, *Albuquerque, New Mexico*

SAS Institute, *Cary, North Carolina*

Schlumberger Offshore Services, *New Orleans, Louisiana*

Schlumberger Well Services, *Houston, Texas*

Science Applications International Corp., *Oak Ridge, Tennessee*

I. P. Sharp Associates, *Palo Alto, California*

Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*

Software Research Associates, *Tokyo, Japan*

Sony Corporation, *Atsugi, Japan*

Space Telescope Science Institute, *Baltimore, Maryland*

Springer-Verlag, *Heidelberg, Federal Republic of Germany*

Stanford Linear Accelerator Center (SLAC), *Stanford, California*

Stanford University, Computer Science Department, *Stanford, California*

Stanford University, ITS Graphics & Computer Systems, *Stanford, California*

State University of New York, *Stony Brook, New York*

Syracuse University, *Syracuse, New York*

Talaris Systems, Inc., *San Diego, California*

Texas A & M University, Department of Computer Science, *College Station, Texas*

Texas Accelerator Center, *The Woodlands, Texas*

TRW, Inc., *Redondo Beach, California*

Tufts University, *Medford, Massachusetts*

TV Guide, *Radnor, Pennsylvania*

TYX Corporation, *Reston, Virginia*

UNI.C, *Aarhus, Denmark*

University of Alabama, *Tuscaloosa, Alabama*

University of British Columbia, *Vancouver, British Columbia, Canada*

University of Calgary, *Calgary, Alberta, Canada*

University of California, Berkeley, Computer Science Division, *Berkeley, California*

University of California, San Diego, *La Jolla, California*

University of California, San Francisco, *San Francisco, California*

University of Chicago, Computation Center, *Chicago, Illinois*

University of Chicago, Computer Science Department, *Chicago, Illinois*

University of Chicago, Graduate School of Business, *Chicago, Illinois*

University of Delaware, *Newark, Delaware*

University of Glasgow, *Glasgow, Scotland*

University of Groningen, *Groningen, The Netherlands*

University of Illinois at Chicago, Computer Center, *Chicago, Illinois*

University of Kansas, Academic Computing Services, *Lawrence, Kansas*

University of Maryland, *College Park, Maryland*

University of Massachusetts, *Amherst, Massachusetts*

University of North Carolina, School of Public Health, *Chapel Hill, North Carolina*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

University of Ottawa, *Ottawa, Ontario, Canada*

University of Southern California, Information Sciences Institute, *Marina del Rey, California*

University of Tennessee at Knoxville, Department of Electrical Engineering, *Knoxville, Tennessee*

University of Texas at Austin, Physics Department, *Austin, Texas*

University of Texas at Dallas, Center for Space Science, *Dallas, Texas*

University of Washington, Department of Computer Science, *Seattle, Washington*

University of Western Australia, Regional Computing Centre, *Nedlands, Australia*

University of Wisconsin, Academic Computing Center, *Madison, Wisconsin*

Vanderbilt University, *Nashville, Tennessee*

Villanova University, *Villanova, Pennsylvania*

Washington State University, *Pullman, Washington*

Worcester Polytechnic Institute, *Worcester, Massachusetts*

Yale University, Department of Computer Science, *New Haven, Connecticut*

Request for Information

The TEX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX would be used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TEX and the hardware on which it runs or is being installed. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, you may indicate that member's name, and the information will be repeated.

If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* direct payment to the TEX Users Group, account #002-031375, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home / Bus. Address: _____
_____
_____
_____

QTY	ITEM	AMOUNT
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) - **North America** New (first-time): [ ] $30.00 each Renewal: [ ] $40.00; [ ] $30.00 - reduced rate if renewed before January 31, 1987	
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) - **Outside North America** New (first-time): [ ] $40.00 each Renewal: [ ] $45.00; [ ] $40.00 - reduced rate if renewed before January 31, 1987	
	TUGboat back issues, $15.00 per issue (16), circle issue(s) desired: 1980 (v. 1) #1; 1981 (v. 2) #1, 2, 3; 1982 (v. 3) #1, 2; 1983 (v. 4) #1, 2; 1984 (v. 5) #1, 2; 1985 (v. 6) #1, 2, 3; 1986 (v. 7) #1, 2, 3	

Air mail postage is included in the rates for all subscriptions and memberships outside North America.
Quantity discounts available on request.

TOTAL ENCLOSED: _____
(*Prepayment in U.S. dollars required*)

* * * *

Membership List Information

Institution (if not part of address):

Title:
Phone:
Network address: [ ] Arpanet [ ] BITnet
[ ] CSnet [ ] uucp

Specific applications or reason for interest in TEX:

My installation can offer the following software or technical support to TUG:

Please list high-level TEX users at your site who would not mind being contacted for information; give name, address, and telephone.

Date:
Status of TEX: [ ] Under consideration
[ ] Being installed
[ ] Up and running since
Approximate number of users:
Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Hardware on which TEX is to be used:

Computer(s)	Operating system(s)	Output device(s)

Please answer the following questions regarding output devices used with TEX if this form has never been filled out for your site, or if you have new information. Use a separate form for each output device.

Name _____ Institution _____

A. Output device information
Device name
Model
1. Knowledgeable contact at your site
Name
Telephone
2. Device resolution (dots/inch)
3. Print speed (average feet/minute in graphics mode)
4. Physical size of device (height, width, depth)

5. Purchase price
6. Device type
[ ] photographic [ ] electrostatic
[ ] impact [ ] other (describe)

7. Paper feed [ ] tractor feed
[ ] friction, continuous form
[ ] friction, sheet feed [ ] other (describe)

8. Paper characteristics
a. Paper type required by device
[ ] plain [ ] electrostatic
[ ] photographic [ ] other (describe)

b. Special forms that can be used [ ] none
[ ] preprinted one-part [ ] multi-part
[ ] card stock [ ] other (describe)

c. Paper dimensions (width, length)
maximum
usable
9. Print mode
[ ] Character: ( ) Ascii ( ) Other
[ ] Graphics [ ] Both char/graphics
10. Reliability of device
[ ] Good [ ] Fair [ ] Poor
11. Maintenance required
[ ] Heavy [ ] Medium [ ] Light
12. Recommended usage level
[ ] Heavy [ ] Medium [ ] Light
13. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Service [ ] Reliable [ ] Unreliable

B. Computer to which this device is interfaced
1. Computer name
2. Model
3. Type of architecture*
4. Operating system

C. Output device driver software
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
[ ] Output device is run off-line
[ ] O/D contains user-programmable micro
[ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)

3. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Purchase price
4. Modifications
[ ] Specified by Stanford
[ ] Designed/built in-house
[ ] Other (explain)

5. Software for interface device
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

E. Fonts being used
[ ] Computer Modern
[ ] Fonts supplied by manufacturer
[ ] Other (explain)

1. From whom were fonts obtained?

2. Are you using Metafont? [ ] Yes [ ] No

F. What are the strong points of your output device?

G. What are its drawbacks and how have you dealt with them?

H. Comments – overview of output device

* If your computer is "software compatible" with another type (e.g. Amdahl with IBM 370), indicate the type here.

TEX Order Form

The current versions of the public domain TEX software, as produced by Stanford University, are available from Maria Code by special arrangement with the Computer Science Department.

Several versions of the distribution tape are available. The generic ASCII and EBCDIC tapes will require a Pascal compiler at your installation. Each tape contains the source of TEX, and WEB (a precompiler language in which TEX is written), and METAFONT. It also contains font descriptions for Computer Modern, macros for AMS-TEX, LATEX, SliTEX and HP TEX, some sample "change files", and many other odds and ends.

Ready-to-run versions of TEX are available for DEC VAX/VMS, IBM VM/CMS and DEC 20/TOPS-20 formats. They contain everything on the generic tape as well as the compiled programs. This means that you will not need a Pascal compiler unless you want to make source changes. Order these tapes if and only if you have one of these systems.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on the tape.

The price of the tapes now includes the cost of the tape reels. Either 1200′ or 2400′ reels will be used depending on the needed capacity. If you order a distribution tape and a font tape, they will most likely be put on a single 2400′ foot reel. All tapes are 1600 bpi.

Please take care to fill in the order form carefully. Note that postage (other than domestic book rate, which is free) is based on the total weight and postal class which you select. Sales tax is added for orders with a shipping address in California.

The order form contains a place to record the name and telephone number of the person who will actually use TEX. This should *not* be someone in the purchasing department.

Make checks payable to *Maria Code*. Export orders must send checks which are drawn on a US bank. International money orders are fine. Purchase orders are accepted if your company has a policy of prompt payment (30 days maximum).

Your order will be filled with the current versions of software and manuals at the time it is received. Since some versions are "pre-announced", please indicate if you want to wait for a specific version.

Telephone calls are discouraged, but if you must call, please do so between 9:30 a.m. and 2:30 p.m. West Coast time. The number for Maria Code is (408) 735-8006. Do not call for advice or technical assistance since no one is there who can help you. You may try Stanford or some other of the helpful people whose names appear in TUGboat.

TEX Order Form

Distribution tapes:

ASCII generic format ____
EBCDIC generic format ____
VAX/VMS Backup format ____
DEC 20/TOPS-20 Dumper format ____
IBM VM/CMS format ____

Tape prices: $92 for first tape, $72 for each additional tape

Font tapes (GF files):

200/240 dpi CM fonts ____
300 dpi CM fonts ____

Allow 2 lbs shipping weight for each tape ordered.

Documents:	Price $	Weight	Quantity
TEXbook (vol. A) softcover	25.00	2	____
TEX: The Program (vol. B)	37.00	4	____
METAFONTbook (vol. C) softcover	22.00	2	____
METAFONT the Program (vol. D)	37.00	4	____
Computer Modern Typefaces (vol. E)	37.00	4	____
WEB language *	12.00	1	____
TEXware *	10.00	1	____
BibTEX *	10.00	1	____
Torture Test for TEX *	8.00	1	____
Torture Test for METAFONT *	8.00	1	____
LATEX – document preparation system	25.00	2	____

* published by Stanford University

Payment calculation:

Number of tapes ordered _____ Total price for tapes _____
Number of documents ordered _____ Total price for documents _____
Add the 2 lines above _____
Orders from within California: Add **sales tax** for your location _____

Shipping charges: (for domestic book rate, skip this section)

Total weight of tapes and books _____ lbs.
Check type of shipping and note rate:

____ domestic priority mail: rate $ 1.00/lb.
____ air mail to Canada and Mexico: rate $ 1.50/lb.
____ export surface mail (all countries): rate $ 1.00/lb.
____ air mail to Europe, South America: rate $ 4.00/lb.
____ air mail to Far East, Africa, Israel: rate $ 6.00/lb.

Multiply total weight by shipping rate. Enter **shipping charges:** _____

Total charges: (add charges for materials, tax and shipping) _____

Methods of payment: Check drawn on a US bank. Make payable to Maria Code.
International money order.
Purchase order (maximum 30 days allowed for payment)

Send order to: Maria Code, Data Processing Services,
1371 Sydney Drive, Sunnyvale, CA 94087

Name and address for shipment: _____

_____

_____

Contact person (if different from above): _____

Telephone: _____

9/86

MACROTEX:

A TEX toolkit. An extensive set of macros that function in a Plain TEX environment. The functionality of a full macro package without restricting your complete access to TEX. Compatible with preexisting macros or macros yet to be designed.

Index Production
Table of Contents Production
Modular Page Numbering
(i.e., Appendix-1)
Cross Referencing,
(including modular page numbers, Figure, Table, and Equation numbers)
Letter, Report, Book formats
Glossary Production
Margin Notes
Text Wrapping around Partial Page Figures.
Partial Verbatim, Boxed Verbatim, and Nofill environments.
Screen Simulation
Elaborate Listing Environment
Table Macros: including auto text wrapping in column, \obeylines and listing macros working inside tables; Tables that break across pages with heading continuing with each new page.
Auto ruled or boxed tables.
Tables that automatically align columns of numbers on the decimal point.
Mailing Label Production
(from a list of addresses with varying numbers of lines for each address)
Diagonal lines
Diamond shapes
And Many Others...

```
    A screen outline forms

around
   these words            and symbols: &%$#~\
              in a verbatim environment.
```

```
   Or we      can use
       a
         nofill macro   that

   maintains spaces      and blank lines while allowing
   Font Changes and  other TEX
        control sequences to continue functioning.
```

SAMPLE INDEX

B

Backup Category, 7
Backup Date Inquiry, 6–9
Backup Disk Files to Tape, 6–**8**, 9–10
  backup categories, 10
    audit control logs, 10
Backup System Status Messages, 14

Documentation includes directions for MACROTEX use as well as the complete MACROTEX code annotated with suggestions for customization. Diskette included for PC or Macintosh users.

Price: $50 for individuals, site license available for corporate and academic users. Please write or call for complete descriptive brochure and order form.

TEXnology, Inc.

Amy Hendrickson, Consultant

57 Longwood Avenue, Brookline, MA 02146 *(617) 738-8029*

Custom Macro Writing • Instruction • Book Production

DETERMINED.

TEX Support for PC's

Drivers for PC-TEX, MicroTEX, and PC's used as terminals. All drivers support pxl and pk files.

dviscrn – a screen driver for most DOS machines including Tandy 2000 and TI machines. Uses any available TEX fonts at full or 'half' resolution — $100.

dvidmp – a driver for TI-855, Epson LQ, Apple Imagewriter, Tandy DMP-2100, HP Thinkjet, etc. — $100.

dvi? – for QMS -KISS -Smartwriter -Lasergrafix, TT's Jlaser, and soon others including Postscript machines — $200.

We also have TEXnC for many unix systems including IRIS, Convex, AT&T, Pyramid etc. Coming soon: TEX for Xenix machines.

AmigaTEX

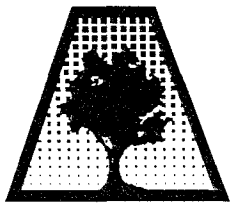
TEX for the Amiga is now available. This version of TEX was translated into C by Tom Rokicki and is well suited to Unix style operating systems. It runs on a 512K, 2–floppy drive system; add memory and it is possible to run an editor, the previewer, and TEX in multitasking mode.

Includes TEX, LaTEX, IniTEX, and a previewer with fonts — $300.

QMS–Kiss and –Smartwriter driver with fonts — $200.

Coming soon: Drivers for other printers, dot matrix and laser. Also, AtariTEX for the Atari–1040ST.

Norman Naugle
n^2 Computer Consultants
P.O. Box 2736
College Station, TX 77841



WE'RE

We're ArborText and that means new and innovative products for you. Products like the new ASCII terminal Previewer, Font design package, and Compugraphic driver. *The Publisher*™ is one that we're really excited about. It's a structured documentation system based on interactive TEX that offers

- ☐ Ease of use.
- ☐ WYSIWYG editing.
- ☐ WYSIWYG mathematic and table formatting.
- ☐ WYSIWYG Previewer with PostScript screen fonts.
- ☐ Industry standards: TEX, SGML, and PostScript.
- ☐ Open architecture programmable system.
- ☐ Upward compatibility with all true TEX systems.
- ☐ Interactive or batch page processing
- ☐ Valid, coded .tex and SGML files.
- ☐ Standard DVI output.

Introductory Offer

Because *The Publisher* is our most exciting new product ever, we are offering a special get-acquainted price. If you are one of our thousands of current customers using PC's, workstations, and mainframes, you qualify for a fifty percent discount off the list price for the beta release of *The Publisher* for a limited time only. The beta version of *The Publisher* runs only on Sun III workstations. You may also qualify for the discount by ordering other ArborText products at the same time you order *The Publisher*. All customers ordering the beta release will be upgraded without charge to the official first release product. The single copy list price of *The Publisher* is $2,000; $1,500 with an academic discount.

Font Tools

The Font Tools package contains Metafont and PKedit—a powerful pair of programs for designing your own screen and laser printer fonts. Metafont is a font creation program written by Donald Knuth for use with TEX. PKedit is a character editing program that can be used to fine tune scanned graphics and low resolution Metafont characters for inclusion in TEX

ARBORTEXT.

files and to design new fonts from existing .pk files.

Adobe Screen Fonts

Preview can now display a full range of tuned screen fonts for the Times Roman, Helvetica, Courier, and Symbol faces. These fonts represent the base set of fonts that are delivered with every PostScript printer. The ArborText Preview program runs on IBM PC's and compatibles, Sun Workstations, Apollo Domain Computational Nodes, and Digital VAXstation II's.

Terminal Previewer

DVICRT is an inexpensive solution to Previewing on any ASCII terminal or line printer. DVICRT displays formatted TEX pages with actual line breaks and page breaks.

DVICG

ArborText has developed a driver for the Compugraphic 8000, 8400, and 8600 phototypesetters. DVICG allows you to use the resident Compugraphic fonts alone or combine them with the TEX Computer Modern fonts.

New Low Prices

We are pleased to announce that effective March 1, 1987 we have once again cut most of our prices in half. In addition, we have instituted a standard, academic discount of 25 percent. Phone us for complete pricing information and product listings or send in the form below.

Yes, I am interested in receiving information about:
☐ TEX software. ☐ Prices. ☐ The Publisher.

Name
Title
Company/School
Address
City State Zip
Business Phone ( )
Computer Hardware
Printer(s)

ARBORTEXT INC.
416 Fourth Street
P.O. Box 7993
Ann Arbor, Michigan 48107
(313) 996-3566