In the end, it should be clear that descriptive markup is not just the best åpproach of the competing markup systems; it is the best imaginable approach.

James H. Coombs, Allen H. Renear, and Steven J. DeRose
"Markup systems and the future of scholarly text processing"
Communications of the ACM
(Volume 30, Number 11, November 1987)

TUGBOAT

Communications of the TEX Users Group

EDITOR BARBARA BEETON

VOLUME 9, NUMBER 1 • APRIL, 1988
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 1988.

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. 9, No. 2, is May 16, 1988; 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 electronic form are encouraged, via electronic mail, on magnetic tape or diskette, or transferred directly to the AMS computer; contributions in the form of camera copy are also accepted. 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

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

Frederick H. Bartlett
The Bartlett Press, Inc.
564 Amsterdam Avenue, 3B
New York, New York 10024
212-787-0395

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

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
bnb@Seed.AMS.com,
bnb@xx.lcs.MIT.Edu,
Beeton@Score.Stanford.Edu

Peter Breitenlohner
Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)
Foehringer Ring 6
D-8000 München 40
Federal Republic Germany
(089) 32 308-412
Bitnet: PEB@DM0MPI11

Malcolm Brown
AIR/Systems Development
Sweet Hall 3rd floor
Stanford University
Stanford, CA 94301
415-723-1248
MBB@jessica.Stanford.Edu

Marcus Brown
Dept of Computer Science
Texas A & M University
College Station, TX 77843-3112

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

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

Malcolm Clark
Imperial College Computer Centre
Exhibition Road
London SW7 2BP, England
Janet: texline@uk.ac.ic.cc.vaxa

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
Crawford-J@Ohio-State
Bitnet: TS0135@OHSTVMA

Jackie Damrau
Dept of Math & Statistics
Univ of New Mexico
Albuquerque, NM 87131
505-277-4623
damrau@dbitch.unm.edu
Bitnet: damrau@bootes

Shane Dunne
Department of Computer Science
University of Western Ontario
London, Ontario N6A 5B7, Canada

Allen R. Dyer
2922 Wyman Parkway
Baltimore, MD 21211
301-243-0008 or 243-7283

Maureen Eppstein
Administrative Publications
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-725-1717
as.mve@Forsythe.Stanford.Edu

Jim Fox
Academic Computing Center HG-45
University of Washington
3737 Brooklyn Ave NE
Seattle, WA 98105
206-543-4320
fox@uwavm.acs.washington.edu
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
furuta@mimsy.umd.edu

Alonzo M. Gariepy
OAIP, 18th floor, Mowat Block
900 Bay Street
Toronto, Ontario M7A 1L2 Canada
416-965-8765

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

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

William Hawes
Box 308
Maynard, MA 01754

Doug Henderson
Division of Library Automation
University of California, Berkeley
186 University Hall
Berkeley, CA 94720
415-642-9485
Bitnet: dlatex@ucbcmsa

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

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

Don Hosek
Platt Campus Center
Harvey Mudd College
Claremont, CA 91711
Bitnet: dhosek@hmcvax

Doris T. Hsia
2630 Sierra Vista Ct
San Jose, CA 95116
415-723-8117

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

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

Richard J. Kinch
Kinch Computer Co.
501 S. Meadow St.
Ithaca, NY 14850
607-273-0222

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

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

Silvio Levy
Princeton University
Fine Hall, Washington Road
Princeton, NJ 08544
levy@princeton.edu

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

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

Marie McPartland-Conn
Henco Software
100 Fifth Avenue
Waltham, MA 02054
617-466-4220

Robert Messer
Department of Mathematics
Albion College
Albion, MI 49224
Bitnet: RAM@ALBION

Tim Morgan
Department of Information and
Computer Science
University of California
Irvine, CA 92717
714-856-7553
morgan@ics.uci.edu,
Bitnet: morgan@uci

David Ness
TV Guide
Radnor, PA 19088
215-293-8860

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

Hubert Partl
EDP Center of the
Technical University Vienna
Wiedner Hauptstraße 8–10
A-1040 Wien, Austria
Bitnet: z3000pa@awituw01

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
CSnet: platt@uofm.cc.cdn
Bitnet: platt@uofmcc

Thomas J. Reid
Computing Services Center
Texas A&M University
College Station, TX 77843
409-845-8459

Tom Rokicki
Box 2081
Stanford, CA 94309
415-326-5312
Rokicki@Polya.Stanford.edu

Barry Smith
Kellerman & Smith
534 SW Third Ave
Portland, OR 97204
503-222-4234; TLX 9102404397
Usenet: tektronix!reed!barry

Shozo Taguchi
Deputy General Manager
Software Division
Fujitsu Limited
140 Miyamoto, Numazu-shi
Shizuoka-ken 410-03, JAPAN

Rilla Thedford
Intergraph Corporation, MS HQ1006
One Madison Industrial Park
Huntsville, AL 35807
205-772-2440

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

Jennifer L. Vollbrecht
Kinch Computer Co.
501 S. Meadow St.
Ithaca, NY 14850
607-273-0222

James W. Walker
Department of Mathematics
University of South Carolina
Columbia, SC 29208
803-777-3882
Bitnet: N410109@univscvm

Samuel B. Whidden
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
sbw@Seed.AMS.com

Ken Yap
Dept of Computer Science
University of Rochester
Rochester, NY 14627
Ken@cs.Rochester.edu
Usenet: ..!rochester!ken

Hermann Zapf
Seitersweg 35
D-6100 Darmstadt
Federal Republic Germany

General Delivery

From the President

Bart Childs

Our TEX world has been progressing in impressive fashion. In spite of the fact that large number of errors have been found and corrected in the past year, TEX is a mature and stable program. TEX 2.9 is now widely distributed. When you consider the size of TEX it is amazing that Don was able to create it at the level he did. Of course it is difficult to compare it with other software items because each is different. A high level language (HLL) does not have as many commands or elements. TEX has about 200 primitives and plain TEX adds about 700 macros to that. Most HLLs don't come close and from vendor to vendor the compiler is usually quite different. Barbara Beeton posed an interesting question, "if another error is found, what version of TEX will it be?"

I have finished (?) a port to the Cray under CTSS. True to form, some errors in the Pascal compiler were found. The reason I question that I have finished is that I can't run the `trip` test. One of the errors is that I can't yet get input from the terminal. The trip test requires it. I plan to submit an article on the port to the next issue and include a few timings.

The finance committee has met twice since the annual meeting. We are in good condition and have added two permanent professionals, Clifford Alper and Alan Wittbecker, to our staff in Providence. Ray worked hard for that and we appreciate the efforts of other TEXers that came up with the leads on these two.

I am preparing a review on the "TEX: the Program" course that Don taught a year ago at Stanford. This course was videotaped, and the tapes are now available; see page 87 for details. We are also planning to have a small committee prepare a detailed syllabus on the courses that TUG offers. This will be available to all. We wish to ensure quality and continuity in TEX instruction by TUG and internal courses as well.

In the process of incorporation, Ray took the opportunity to make a slight change in the way we had been operating. With our recent successes, it became obvious that the positions of President and Treasurer require more interaction with him. It would increase the continuity with TUG if these positions were not elected in the same year. At this summer's meeting in Montreal we will be asked to confirm this schedule.

We decided to send Alan Hoenig to the TEX meeting in Exeter this summer as a representative of TUG.

I can offer nothing but praise for the other officers you have elected. Rilla Thedford is building the volunteer network and Alan is timely with his correspondence. Thank you for providing such a good group. I think they simply reflect the organization.

Again, the TUGboat continues to improve. I found Doug Henderson's METAFONT column particularly useful and enlightening. You contributors keep up the good work.

Happy TEXing.

TUGboat Docks at Fire Station

Alan Wittbecker
TEX Users Group

Alas, TUG outgrew its space at the AMS. As the result of a search for larger quarters, TEX Users Group will be moving on **May 1st** to the old (1866) Niagara fire station at 653 North Main Street in Providence. TEX Users Group will occupy the top floor of the station, which has 1,200 square feet of office floor space and 800 square feet of storage space (the lower floor is occupied by Computopia, a retail computer store).

The station was renovated by Morris Nathanson Design and includes original firehouse memorabilia. Of great importance to us is the fact that it is only one block away from the AMS, so contacts and exchanges will continue to be close. TUG will be announcing a new phone number as soon as arrangements are complete.

All correspondence should continue to be addressed to:

TEX Users Group
P. O. Box 9506
Providence, Rhode Island 02940–9506

All payments should continue to be addressed to:

TEX Users Group
P. O. Box 594
Providence, Rhode Island 02901

The new address for delivery services is:

TEX Users Group
653 North Main Street
Providence, Rhode Island 02903

Donald E. Knuth Scholarship

Marie McPartland-Conn and Doris Hsia were honored at the 1987 Annual Meeting, University of Washington, Seattle, as the 1987 Scholarship Winners. They have volunteered to serve on the 1988 selection committee.

We are pleased to announce the Third Annual "Donald E. Knuth Scholarship" competition. Once again this year **two** Scholarships will be awarded. The awards consists of an all-expense-paid trip to TUG's 1988 Annual Meeting and the Short Course offered immediately following the meeting. Funding for one of the scholarships will be provided by ArborText, Inc., Ann Arbor, Michigan, and for the other, by the TEX Users Group. The competition is open to all 1988 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 13, 1988, 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 McGill University, Montréal, Canada, August 22–24, 1988.

Selection of the scholarship recipient will be based on the TEX sample. Judging will take place May 14 – June 13, and the winner will be notified by mail after June 14.

The Scholarship Committee consists of Marie McPartland-Conn, chairperson; Doris Hsia, Stanford University; and Alan Wittbecker, TEX Users Group. All applications should be submitted to the Committee at the following address:

Marie McPartland-Conn
Henco Software
100 Fifth Ave.
Waltham, MA 02054

Notes from the Editor

Barbara Beeton

A few recent inquiries from readers have pointed out that (1) some people out there really *do* read TUGboat, and (2) some features your editor thought obvious really aren't obvious at all. So, here are some random notes on what to look for, and where.

Addresses of all authors, officers, steering committee members and others mentioned in significant capacities in an issue are given in the "best" form available starting on the next *recto* page after the title page. Starting with this issue, we're sending proof of the address list as well as proof of each article to the author for verification before it's published.

Cover 3 — that's the inside back cover — is an extra-charge option for advertisers, but if TUG has something particularly important to announce, cover 3 will be reserved for that. Cover 3 of issue 8#3 contained the first announcement of the 1988 Annual Meeting, to be held at McGill University in Montréal from August 22–24. The meeting is also listed in the Calendar, and both the announcement (Call for Papers) and the Calendar are listed in the contents on the back cover. So the several inquiries, "When and where is this year's meeting?" tell me that something's just not getting through; we'll try harder to make this information more prominent, and suggestions for how to do it are welcome.

Suggestions for anything relevant to the present or future contents of TUGboat are welcome, of course. Editorial information, including the next issue's deadline (also in the Calendar) and the names of Associate Editors for various topics, is on the back of the title page, facing the address list. Addresses for the TUG office, including the street address (for express shipments that can't be delivered to a post office box) are given with the general TUG information on the inside front cover, and addresses of the various editors are included in the aforementioned address list.

The rapid growth of TUG's membership and continuing demand for back issues of TUGboat has made reprinting necessary. Some issues and full volumes have been reproduced from printed issues rather than from the original camera copy. This means that the statement "... can be taken as representative of the output..." is not necessarily true. Volumes 7 and 8 (all issues bound together) and some copies of issues 7#1 and 8#1 that were

not part of the initial subscription mailing are the only items affected.

Some efforts at generating standards or guidelines for certain topics of importance to TEX users deserve a reminder. Robert McGaffey's committee to develop output driver standards (TUGboat 8#2, p. 161) can still use some more input; if you've just lost an argument with your output driver, which refuses to digest your new font of 256 characters, or tells you that it can only accept more than 4 fonts on page 27 when the moon is gibbous, send your suggestions to Robert. (See the address list ...) Christina Thiele and her committee are likewise attempting to bring order out of chaos by developing guidelines for writing scrutable macro packages (TUGboat 8#3, p. 307). It would be stretching the truth to say that all the macros I've written are well-structured and exhaustively documented; nonetheless, there are times when documentation and careful visual structuring are essential simply to preserve one's sanity under later deadline pressure. Send your best (and worst — what *not* to do is also instructive) ideas.

The TUG office has a new TEXnician, Alan Wittbecker, previously with Nieman Ryan Publishing in the Northwest. He will be assisting in TUGboat production, as will a new volunteer, Tom Reid of Texas A&M University. They will be helping to retag the TEX source files submitted for publication to use standard TUGboat macros where possible, or to make sure that newly-defined macros in one item don't subvert existing standard macros or persist to defeat the formatting of subsequent articles when a whole issue is strung together. It is my hope that, beginning with this issue, items can be returned to the authors after publication so that they can see what has been done to incorporate their work into a larger document. For authors whose submissions arrived on floppy disk, we intend to include the TUGboat macros in the return package, to permit the articles to be rerun in TUGboat format by the author, and, not inconsequentially, to provide an incentive for authors to prepare future articles directly in TUGboat format.

I would like particularly to thank one volunteer whose contribution was not properly acknowledged in the last issue. Doug Henderson, TUG's METAFONT coordinator and implementor of PCMF, prepared the APL font (TUGboat 8#3, p. 275) and Silvio Levy's text Greek font (p. 20, this issue) for both laser printer and typesetter, a task for which no staff time was available at the Math Society. He is also serving as a repository of information on METAFONT parameter settings for various output devices and of `.mf` source files for non-CM fonts and CM fonts in nonstandard sizes.

Two other volunteers whose long and devoted service deserves special recognition are Ken Yap, keeper of the LATEX style collection, and Don Hosek, keeper of output device driver information. Both have worked hard to assure the accuracy of the information in their columns, suffered the indignity of having errors introduced during editing, and adjusted to the dictates of this Editor even if their local systems didn't make the task easy. Ken is looking forward to receiving his degree next year, and will be leaving Rochester; before he leaves, he would like to find a new home for the LATEX collection, with someone who has a strong interest in LATEX, an account on an Internet node that supports anonymous FTP, the support of the local system management in providing disk space and other encouragement, and time to do the job right. Particulars can be obtained from Ken by sending him an inquiry by electronic mail at `Ken@cs.Rochester.edu`; please send a copy to `bnb@Seed.AMS.com`.

And speaking of electronic mail, the TUG office now has Internet access through the Math Society's connection. Inquiries regarding membership records, subscriptions or orders can be sent to the TUG office manager, Karen Butler, `klb@Seed.AMS.com`. Messages to management should go to Ray Goucher, `reg@Seed.AMS.com`. Address TEXnical inquiries to Alan Wittbecker, `aew@Seed.AMS.com`.

Finally, some comments on TEX outside of TUG. Chuck Bigelow's article, "Notes on typeface protection" (TUGboat 7#3, p. 146), has now been reprinted several times; the most captivating is the French translation by Jacques André in *TSI — Technique et Science Informatiques* under the title "Du piratage de fontes" — it makes the topic sound dreadfully sinister. Reviews of the IBM PC and Macintosh implementations of TEX appeared in the January 11 issue of *The Scientist*, accompanied by benchmark data prepared by Malcolm Brown, moderator of TEXhax. And, on the first page of the *Boston Globe* want ads for January 31, Stratus Computer listed a position for a publications specialist to work with "Tex [sic] (a typesetting language)". If you spot a reference to TEX or METAFONT or TUG in any publication that may not make it to Rhode Island, send us a copy or citation — we like to keep informed about what the rest of the world is thinking.

Document Production: Visual or Logical?

Leslie Lamport

The Choice

Document production systems convert the user's input — his keystrokes and mouse clicks — into a printed document. There are many different ways of classifying these systems. I will discuss a classification based on the extent to which the user regards his document as a visual structure rather than a logical one. A system in which the user specifies a visual description of the output will be called a *visual* system, and one in which he specifies the logical structure of his document will be called a *logical* system. Visual systems may be more convenient for short, simple documents like love letters or laundry lists. However, I will argue that logical systems are better for more complex documents like books and technical articles.

In a purely visual system, one would simply paint a collection of pixels on the screen. The word *cat* would be no different from a picture of a cat — the user could change the shape of the *t* as easily as he could change the shape of the tail in a picture of a cat. Finding all instances of *cat* and replacing them by *dog* would be as hard as finding all cats in a picture and replacing them with dogs.

In a purely logical system, one would enter only the logical structure of a document, describing such things as words, paragraphs, theorems, sections, and cross-references. The system would translate this logical structure into a collection of dots on sheets of papers, with the user giving only general instructions — for example, specifying two-column output formatted for a conference proceedings.

There are no purely visual systems used for document production. All systems keep some logical representation of the document that they use to generate the pixels. The most primitive ones keep only the letters that generate the characters. In such a system one can easily find all instances of *cat*, but a search for all instances of *domestic* would miss the ones in which the word is hyphenated across lines. More sophisticated systems keep more of the logical structure, thereby acting more like logical systems. It is my thesis that such systems are good for serious document production only to the extent that they act like logical systems.

Reprinted from *Notices of the American Mathematical Society* (1987), Mathematical Text Processing, "Document Production: Visual or Logical?", Volume 34, pages 621–624, by permission of the American Mathematical Society.

I know of no purely logical system that is currently available. Systems like *Scribe* and LaTeX permit the user to describe the visual appearance as well as the logical structure of the document — for example, by inserting a command to add a quarter-inch vertical space. The need to provide the user with such commands is a symptom of the deficiencies of these systems.

Current logical systems require the user to describe his document as a text string, filled with obscure-looking commands. This is a cumbersome way to represent the logical structure of a document; it is a sign of the primitive nature of these systems, not an inherent feature of logical document production. Systems can be built to allow more convenient editing of the document's logical structure. I'm not interested in the question of whether the inconvenience of describing the document with an ASCII text file is bad enough to make visual systems preferable. Choosing between two evils is never pleasant. I will confine myself to arguing the inherent superiority of logical systems to visual ones.

Computers Work Logically, Not Visually

In a recent paper, I used the notation f^e_x to denote the result of substituting e for x in f. With a visual system, I would have entered this notation by simply putting the e above and to the right of the f and the x below and to the left. Using LaTeX, I might have typed the formula as `f^{e}_{x}`, the `^` command indicating a superscript and `_` indicating a subscript. This input would have been a partially logical and partially visual description — logical because the subscript and superscript are denoted logically by commands rather than visually by placement, but visual because it describes the representation (super- and subscript) rather than the logical concept of substitution. I therefore chose to define a command `\subfor` of three arguments, and typed the formula f^e_x as `\subfor{f}{e}{x}`. The input then unambiguously describes the logical structure that it represents. For example, the input would distinguish the result of substituting 3 for i in s, represented as `\subfor{s}{3}{i}`, from the cube of the i^{th} element of a sequence s, represented as `s^{3}_{i}`, even though both are printed as s^3_i.

After I had completed the first draft of my paper, someone told me that I had used the wrong notation; the result of substituting e for x in f should be denoted by f^x_e rather than f^e_x. I had to reformat my paper to conform to the correct notation. Had I used a visual system, in which only the visual representation is maintained, I would

have had to examine every page visually to find all instances of the notation and changed each one individually. Had I used the half-logical method, entering `f^{e}_{x}`, I could have written a program to find all text strings of the form `...^{...}_{...}` and allow me to choose whether to transform it. (Human intervention would still be required to prevent changing the cube of s_i to s_3^i.) Having chosen a logical representation, I merely had to change the definition of the `\subfor` command — a simple change at a single point in the text.

The ease of making the change when the notation was represented logically rather than visually was no fluke; it was a consequence of the fundamental fact that computers are good at processing logical information, but bad at processing visual information. Recognizing that f_x^e is a single logical entity with three components is a difficult problem of artificial intelligence if one is given only the visual representation, but it is a trivial programming exercise if f_x^e is represented as `\subfor{f}{e}{x}`.

Writing or Formatting?

The purpose of writing is to convey ideas to the reader. The worst aspect of visual systems is that they subvert the process of communicating ideas by encouraging the writer to concentrate on form rather than content. Ideas are conveyed by the logical structure of the text; the function of the visual format is to display this structure. The author should be concerned with the structure, not any particular visual representation.

Visual systems encourage the user to substitute formatting for good writing. A simple example is the use of vertical space. If there's an awkward transition from one paragraph to the next, the user of a visual system can simply add some vertical space between the paragraphs. But, what does this space accomplish? The awkward transition is still there; the reader is still jarred by it. The extra space simply declares that there is an awkward transition and the author is either too lazy or too bad a writer to fix it.

An awkward transition is a symptom of a poorly structured document; it can be fixed only by restructuring the document. A logical system forces the writer to think in terms of the document's logical structure; it doesn't give him the illusion that he is accomplishing anything with cosmetic formatting changes.

Phosphors or Ink?

Proponents of many visual systems boast that they let the user work with an exact replica of the printed page. In fact, a serious drawback of many visual systems is that they force the user to work with an exact replica of the printed page. When the author is editing his document, he becomes a reader. Like any reader, he wants to be presented with the document in a format that is easy to read. A format that is adapted to the printed page is a poor one for a screen. Phosphors are different from ink, and a screen is not a piece of paper; it is not easy to read a picture of a printed page on a screen.

A computer screen differs from a printed page in many ways, including resolution, width, and the availability of different colors. Each of these differences implies differences in the way information should be displayed. In addition to the differences in the two media, the presence of a computer behind the screen also has striking implications. Consider the problem of pagination. One of the worst features of books is the splitting of text across pages. It would be easier to read a document straight through, from front to back, if it were printed as a continuous scroll. We use books rather than scrolls because they are easier to produce and because documents are not always read in such a linear fashion. The computer offers the best of both worlds. We can scroll through text, avoiding distracting page breaks, and still move easily to another part of the document. It is senseless to use a computer to simulate a book, complete with page breaks.

A typical writer of technical material spends two to eight hours per page writing. He spends much of that time looking at the representation of the document on his screen. A visual system that forces the writer to view on his screen a version formatted for paper makes his task harder.

Who Should do the Formatting?

Logical systems attempt to remove formatting concerns from the author. The author specifies only the general form of the output — technical report, journal article, etc. — while the system makes the actual formatting decisions — amount of paragraph indentation, amount of space above a displayed equation, etc. Visual systems give free rein to the author's artistic tendencies, allowing him to format everything as he wishes. This would be fine if documents were meant to be displayed on walls and admired for their aesthetic qualities, but they're not.

The purpose of writing is to convey ideas to the reader. The purpose of formatting is to make the document easier to read, not to look pretty. Document design is a skill acquired through training and experience. A logical system can apply the skill of a trained designer to the formatting of a document. A visual system forces the author to do his own document design, often with disastrous results. Most authors are not competent designers and make typographic errors — formatting decisions that make the document harder to read.

A LaTeX user once complained because he wanted to format an equation to look something like this:

$$\forall i : \qquad f(x_i) > g(y_i) \tag{7}$$

Formatting the equation in this way would have been easy with a visual system; he would just have put everything where he wanted it. However, LaTeX provides no easy way to do this. The user just enters the equation and LaTeX formats it the way it wants. (It also assigns the equation number.) If the user declares the $\forall i$ to be part of the equation, the result looks like this:

$$\forall i : f(x_i) > g(y_i) \tag{8}$$

If he declares the $\forall i$ to come before the equation, then LaTeX makes it part of the text preceding the displayed equation.

This particular user found the formatting of (7) more aesthetically pleasing than that of (8), and I agreed with him. However, (7) is a typographical mistake. Equations are numbered so they can be referred to in the text. When the reader encounters a reference to (7), it is not immediately clear from the formatting whether it refers to the entire equation $\forall i : f(x_i) > g(y_i)$ or just to the inequality $f(x_i) > g(y_i)$. It is clear from the formatting that (8) refers to the whole equation and that, if the $\forall i$ were part of the preceding text, then the equation number would refer only to the inequality. The formatting of (7) introduces an ambiguity, making the document harder to read.

The purpose of document design is to display the logical structure of the document through its formatting, thereby making it easier to read. A user with no training in design is easily seduced by a visual system into formatting the document to be aesthetically pleasing, often making it harder to read.

A visual system can makes things hard even for a trained designer. An important principle of document design is uniformity — the same logical element should be formatted the same way throughout the document. It is difficult to achieve uniformity if the user must specify the formatting of each instance of the element. For example, all displayed quotations should be indented the same amount, but this is not likely to happen if the user must specify the amount of indentation whenever he types a quotation.

Must the User Ever Format?

There are two reasons why the author may have to specify formatting in a logical system. First, no logical system can provide a complete assortment of predefined logical structures. For example, a general-purpose system is unlikely to provide facilities for formatting recipes. The writer of a cookbook must tell the system how to format recipes — hopefully, after consulting a professional designer. A logical system should permit the user to define his own logical structures and to specify how they are to be formatted. Several different formats might have to be specified — for example, one for a single-column page, one for a double-column page, and one for the computer screen. In a logical system he does this once; in a visual system he must format each recipe individually.

The second reason for specifying formatting is to overcome an inherent problem with computers. Embodying design principles into programs is difficult, and a designer will always be able to do a better job of formatting an individual document than will a computer program that he devises. Achieving the highest possible quality requires the ability to make changes to the system's output. This will be a matter of fine tuning, changing such things as page breaks and figure placement. This is a visual process, and one would like a visual system for doing it — one that allows the user to manipulate screen images of the final output.

If such visual editing is ultimately desirable, why not use a visual system in the first place? The answer is that the flea should not wag the dog. The changes will generally be of such a minor nature that they are not worth bothering with in a preliminary version intended for a small audience, nor for any document that is not widely distributed. They will be done only when producing the final copy for the publisher.

Even using LaTeX, which does not make the final formatting very easy, I usually spend less than two minutes per page doing the final formatting to produce camera-ready output. This is insignificant compared with the two to eight hours per page I spend writing. There is much more to be gained by making writing easier than by simplifying the final formatting task.

Software

Still another aspect of multiple change files: The PATCH processor

Peter Breitenlohner

Recently there have been quite a few TUGboat articles about extensions of the WEB system, either extensions to other languages like C or Modula-2, or extensions which allow multiple change files (W. Appelt and K. Horn in TUGboat 7(1986)20, K. Guntermann and W. Rülling in 7(1986)134, E.W. Sewell in 8(1987)117). Surprisingly enough (at least for me) another extension which allows one to *include* or *insert* a WEB file together with its change file(s) into another one has never been discussed. I would therefore like to present a program which does exactly this.

Before describing this program let me recall my motivation to write it. My experience with the WEB system dates back to the time when I had just installed METAFONT, had proudly produced the first GF file and was then told by GFtype that all kinds of backpointers were wrong. This was repaired easily enough — the two programs had different opinions about the record length of GF files — but it has brought the following fact to my attention. The WEB source files for TEX, METAFONT and their friends contain large sections of code which occur in many files in more or less identical form and the changes applied to them had better be consistent. The chapters 'The character set' or 'Packed file format' are typical examples and the updates to GFtoPK, PKtoPX, PKtype and PXtoPK published in TUGboat 7(1986)140 are a characteristic symptom.

If one could make such sections of code completely (not just almost) identical one could keep one copy of this code in a separate file and *include* it whenever needed. This would clearly save a rather large amount of disk space. Much more important this would guarantee that the same changes are applied whenever this section of code is used and would greatly facilitate the task of creating a new set of change files for a new computer, compiler or operating system.

These considerations motivated me to write a program which I have named PATCH because it takes various patches and combines them into one program file. Each patch consists of one WEB file and $n \geq 0$ change files *change_j* $(0 < j \leq n)$ which are applied one after the other. Here PATCH operates exactly like TIE as described by Guntermann and Rülling: first *change_1* is applied to the WEB file as usual, then *change_2* is applied to the result, and so on. In addition there is one short 'patch file' which specifies the names of the WEB and change files. An important milestone is reached when the result of this merging of files yields a record starting with '@i', a control code which is normally undefined. Such a record must contain the file name of a new patch file and the resulting new patch is inserted instead of this record. This new (secondary) patch may, of course, again yield records starting with '@i' and thus invoke further secondary patches and so on.

The PATCH processor has actually three modes of operation: In *merge_mode* PATCH produces a WEB file and *one* change file, and can thus serve as preprocessor for TANGLE and WEAVE. In this mode all change files *change_j* are combined into one change file and all secondary patches are inserted in a suitable way. In *insert_mode* PATCH produces just a WEB file containing all the changes and insertions. This WEB file can also serve as input to TANGLE or WEAVE but in this case the information about changed modules would not be available to WEAVE. Finally, in *update_mode*, all changes are applied to the primary patch. Requests for secondary patches are, however, not honored but copied to the resulting WEB file. This mode of operation is intended to incorporate modifications into a WEB file once they have been fully tested and are frozen.

A next step was to combine PATCH with TANGLE and WEAVE to new programs TPATCH and WPATCH in order to avoid the necessity of a separate preprocessing step. Using PATCH, this turned out to be surprisingly easy due to the clear structure of TANGLE and WEAVE. All the code for the actual processing (modules 37–178 of TANGLE and modules 36–257 of WEAVE) required practically no changes except that the parts which merge the WEB and change file (modules 124–138 resp. the almost identical modules 71–85) had to be replaced by the corresponding code from PATCH. All three processors PATCH, TPATCH and WPATCH have now been used for several months and are thoroughly tested.

Coming back to the original motivation for PATCH one can now try to create patches for things like 'Reading GF files', 'Reading PK files' and 'Reading PXL files', or maybe two versions of them for sequential and random access to the files. At the moment I am in fact doing just this. Such patches should be tools which can be inserted into a program whenever needed. They can be extremely useful for all kinds of DVI drivers which could use any one of them or even two of them alternatively.

Another application is related to the fact that the files `tex.web` and `mf.web` are extremely large; they are in fact too large to be even inspected by our text editor. In order to circumvent this problem one can split the file `tex.web` — D.E. Knuth might forgive this — into smaller files containing the limbo material (`tex00.web`) or the code for one chapter of 'TEX: The Program' (`tex01.web` through `tex55.web`) and split the change files for TEX and INITEX accordingly. The primary patch for TEX will then consist of a short WEB file (say `texskel.web`, the skeleton) containing 56 '`@i`' commands to invoke these patches. This primary patch requires no change file as all changes are applied to the secondary patches. The skeleton file for INITEX will be almost identical. Only one or two secondary patches will be different, their patch files will have to specify an additional change file in order to create INITEX instead of TEX, but there is no necessity to maintain, for each kind of installation, two almost identical change files, one for TEX and one for INITEX. Furthermore one could create a L-R TEX (`TEX-XET`) and/or a multilingual TEX by just adding one additional change file to a few of the secondary patches. These additional change files could probably be installation-independent.

Turkish Hyphenations for TEX

Pierre A. MacKay

Turkish belongs to the class of agglutinative languages, which means that it expresses syntactic relations between words through discrete suffixes, each of which conveys a single idea such as plurality or case in nouns, and plurality, person, tense, voice or any of the other possibilities in verbs. Since each suffix is a distinct syllable (occasionally more than one syllable), Turkish sentences are likely to contain a high proportion of long multi-syllable words, and to need an efficient system of hyphenation for typesetting. Owing to the long association of almost every Turkic-language region with Islam, certain conventions of the language have been deeply influenced by Arabic orthographic habits, and among these is the syllabification scheme on which a system of hyphenation is built.

According to the syllabification pattern of Arabic, a syllable is assumed always to consist of an initial consonant (even when that consonant is no longer written) and to terminate in a vowel `-cv-` or in the next unvowelled consonant `-cvc-`. This pattern is followed so absolutely that it is permitted to break up native Turkish suffixes. The plural suffix *-ler-* will be hyphenated as *-le-rine* in an environment where the `-cv-cv-cv` pattern predominates. A syllabic division of *çektirilebilecek* provides six places for hyphenation *çek-ti-ri-le-bi-le-cek*, while a morphological division of the word would produce only five *çek-tir-il-e-bil-ecek.**

There are almost no exceptions to this pattern. Words which appear to begin with a vowel, like *et-mek*, can also be described as beginning with the now suppressed half-consonant *hamza.* Widely sanctioned orthographic irregularities like *brak-mak* can be found in stricter orthography as *bı-rak-mak.* The only universally practiced violation of the rule is associated with the word *Türk*, in which the *-rk-* combination is inseparable, and contributes to several of the very few three-consonant clusters regularly used in the language—*Türkçe*, *Türkler.* One other significant consonant cluster occurs in the suffix *[i]m-trak.*

The Ottoman Texts Project at the University of Washington has undertaken the development of a set of editing and typesetting tools for the production of texts in modern Latin-letter Turkish, using the full range of diacriticals needed for scholarly editions of historic Arabic-script manuscripts. Because we wish to work in cooperation with scholars in Turkey, who are most likely to have access to unmodified versions of TEX, we have chosen a font-based adaptation of the TEX environment, which will require no alterations in the program. The work on fonts is largely complete, and one of the last major efforts necessary is the creation of a Turkish hyphenation table.

The obvious way to create such a table in the TEX environment, is to run a list of correctly hyphenated words through `Patgen`, but it is not always easy to find such a list. English and German dictionaries quite commonly provide hyphenation patterns, but the dictionaries of the Romance languages rarely do, and in Turkish, the hyphenation pattern is so obvious that the production of such a list is viewed as an unimaginable waste of time. Rather than try to scan a Turkish word-list and supply hyphens, we have taken advantage of the strict formalism of the patterns and generated the Turkish hyphenation file by program.

---

* The word is a future participle, and describes something as being capable of being extracted at some time in the future — like a tooth.

Turkish orthography uses a very large number of accented characters. The Latin-letter character set which has been in use since the orthographic reform of 1928 is extended, even in Modern Turkish, by means of a considerable number of diacriticals and accents. A diligent search through the modern dictionary will produce several five- and six-letter words in which every character is accented, and an intensive search might come up with words as much as nine letters long with every character accented. In critical editions of Ottoman texts, the number of accents more than doubles. Modern Turkish knows only the accented and unaccented pair of letters '**s**' and '**ş**', but Ottoman Turkish has '**s**', '**ş**', '**ṣ**' and '**s̲**', which represent four completely distinct characters in the Arabic alphabet. The letter '**h**' shows almost as much variety, and so do several others. Our Ottoman Turkish font has twenty-seven accent and letter composites, in addition to the basic twenty-six simple Latin letters. Moreover, all composites can exist in upper case forms as well as in lower case. To accommodate these composite characters in the normal ASCII character set, we use an input coding convention in which accented letters are treated as a class of ligatures, and three characters from the ASCII symbol set are borrowed for use as postpositive pseudo-letters, to trigger the selection of accented letters in the Turkish fonts. The three symbols are the exclamation point '!', the equals sign '=', and the colon ':'.

The choice of these symbols is based on a proposal made more than ten years ago at the Orientalist Congress held in Paris, in 1974. Owing to the extraordinary richness of the Ottoman Turkish character set, it has been necessary to extend the old proposal, but it still retains the original principles, which are closely associated with the coding scheme used by the Onomasticon Arabicum project, which is coordinated at the Centre National de la Recherche Scientifique in Paris. (The Onomasticon Arabicum uses a post-positive dot and a post-positive hyphen to indicate diacriticals, which is acceptable in a data-base of names, but not in continuous prose text.) The current set of conventions, using (! = :), produces an input file which can, if necessary, be edited on a ordinary terminal lacking any special Turkish character features, and which a Turkish speaker can become accustomed to without too much difficulty. When coupled with a well-designed macro file and a rewritten hyphenation table, it provides the possibility of naturalizing a TEX environment into Turkish without any large investment in special purpose hardware and rewritten versions of non-standard (non-)TEX.

The exclamation point is used for all the "emphatic" letters of the Arabic alphabet (the alphabet in which Turkish was written until 1928). These are the letters *Ḍad* (usually pronounced as '**z**' in Turkish, and hence paired with a non-Arabic letter known as *Żad*), *Ṣad*, *Ḥa'*, *Ṭa'* and *Ẓa'*. The equals sign is used for all the consonants which are represented in Latin-letter transcriptions by a letter with a bar under, such as '**d̲**' (*dhal*), more commonly written in Turkish as '**z̲**', and also for vowels with a macron or, following the Turkish convention, a 'hat' accent, and similar forms, chosen like the cupped '**ğ**', because the equals sign is visually closer than the colon is. (Moreover, the colon is needed for a different variety of the letter '**g**'.) The colon is a catch-all for everything else, but works out rather well visually, as it happens. The three post-positives are not accents, but regular characters, which use the TEX convention of ligatures to invoke accented characters from the font, just as the second '**f**' in the normal TEX '**ff**' ligature pair does. If a standard Latin-letter character does not have an associated ligature table in the font, a following diacritical postpositive will be unaffected. Thus, the letter '**o**', when followed by a colon will produce '**ö**', but the letter '**e**' when followed by a colon will produce '**e:**'. The equals sign retains its normal function in math mode because the math font `TFM` files do not call it into ligature pairings, and the colon and exclamation point can be invoked by the command sequences `\:` and `\bang` when the simple character will not work.

Since the hyphenation evaluation loop in TEX dismantles all ligatures before it looks for acceptable hyphenation positions, it will have to accept the post-positive symbols (! = :) as part of the alphabet, so each of these symbols receives its own value as an `\lccode`. The full Turkish-TEX alphabet is:

a â e ı i î o ö ô u ü

' ' b c d f g h j k l m n p r s t v y z

ḍ ḥ ḳ ṣ ṭ ẓ

d̲ ğ h̲ ñ s̲ t̲ z̲

ç ġ ş ż

In the hyphenation loop of TEX, these characters resolve into the set:

```
! = : @ # a b c d e f g h i
j k l m n o p r s t u v y z
```

and it is this latter set only which will appear in the hyphenation patterns. The dotted `i` in the above list really stands for the Turkish undotted

'ı'. The input code convention for Turkish uses `i:` for the Turkish 'i'. The `@` sign stands for the Arabic letter *hamza* and the `#` stands for *'ayn*. To avoid conflict with `plain.tex` uses of these two characters, they appear explicitly only in the hyphenation pattern file. Turkish text input uses `\'` to generate `\char'43` (*'ayn*) and `\'` to generate `\char'100` (*hamza*).

We begin constructing the table by considering the pseudo-letters (`! = :`). Since these are used exclusively in ligature pairs, no hyphenation is ever permissible between them and the preceding letter. Odd values permit, and even values in the hyphenation code prohibit hyphenation, so we give the highest possible even value (8) to the region preceding each pseudo-letter. The pseudo-letters can follow both vowels and consonants, so hyphenation will often, but not always, be possible after them. We give that region the lowest possible odd value (1) to show that hyphens are permitted here.

```
8!1 8=1 8:1
```

In strict orthography, a vowel cannot be separated from the preceding consonant, and the few apparent instances of hyphenation between two adjacent vowels (suppressed consonant) can be treated later. In all normal instances a vowel cannot accept a hyphen in the preceding region and will probably accept one in the following region, so the vowels are set thus.

```
2a1 2e1 2i1 2o1 2u1
```

A consonant may begin a `-cv-` sequence or end a `-cvc-` sequence, so we give it a 1 on either side:

```
1b1 ... 1z1
```

This simple lot of patterns will provide for all normal `-cv-` instances such as

```
1h1
  8=1
    2a1
1h8=2a1
```

which will result in the sequence `-h=a-`, with hyphens fore and aft.

The next group of patterns controls hyphenation at the end of words. TEX will usually not break off two-letter fragments in its hyphenation loop, but owing to the nature of the input coding we have chosen, it may see a three- or four-letter sequence where a two-letter result is intended. We do not want to find *lü*, *çü* and *si* isolated at the beginning of a line, nor do we really want the *cek* of *-ecek* broken off if it is at the end of a word. To prevent hyphenations of this sort, the program generates all possible patterns of the type:

```
2ba=. ... 2z:u:.
```

using the conventional `.` for end-of-word. The resultant list includes sequences that are phonetically impossible in Turkish but these take up so little additional space in the file that they can be left there. The pattern `2e2cek` is added as a special case.

The break after `-cvc-` syllables is almost taken care of:

```
1h1
  8=1
    1h1
1h8=1h1
```

but it makes the thoroughly undesirable `-cv-ccv-` sequence as acceptable as the correct `-cvc-cv-` sequence. To prevent this error, all possible Turkish two-consonant sequences (e.g. `h=h=` → '**hh**') are covered by patterns such as `2h=h=`, in which the value 2 will override the 1 after the preceding vowel.

The few undesirable hyphenations at the beginning of words which appear to start with a vowel are prevented by generating the patterns `.a=2` through `.u:2` and similarly, the few instances where an apparent `-cv-v-` hyphenation stands for `-cv-[c]v-` can be allowed by adding the full range of patterns `a3a2` through `u:3u:2` which includes a large number of impossible pairings.

The last patterns to be added are `m1t4ra4k` and `t2u8:2r4k1`. At the price of slightly excessive strictness (the prohibition against the `r-k` division is only valid when the word begins with an uppercase `T`) we can ensure that *Türk* always stays in one piece.

Files of this sort, when generated by program, tend to be larger than hand-worked files, but if it seems that all the redundancies mentioned above might be seriously wasteful of space, consider the following statistics:

	Entries	Trie size	Ops
English	4460	5492	181
Turkish	1840	616	16

The format file that makes use of this set of patterns will no longer serve very well for English language TEX. The font-based solution to foreign-language typesetting is definitely monolingual, since only one `hyphen.tex` file can be read in at a time. A multilingual system, good for both English and Turkish, would require modifications of the program code. This simple solution, however, will be quite satisfactory in a purely Turkish environment, and can be made even more successful by taking the `tex.pool` file and translating it all into Turkish.

FONT FORUM

Georgia K.M. Tobin

The ABC's of Special Effects

Special effects such as reverse video, pattern fills and pseudo shadowing can be fairly easily achieved in a METAFONT font by making use of **picture** variable manipulation. Since the way I did this incorporated the special effect at generation time and thus used the same code which generates my standard fonts, these specialty fonts retain much of the flexibility inherent in any meta-font, i.e. they can be generated for different point sizes and resolutions.

My basic plan of attack was to define a subroutine called *pattern* which in turn redefined the subroutine *endchar*. (Nota Bene: It would not do, under any circumstances, to simply overwrite the definition of *endchar* in plain.mf. METAFONT is quite content to redefine any subroutine whose name appears subsequent to its initial occurrence.) In this way, I can simply input the appropriate pattern code and invoke *pattern* at run time; each letter which is subsequently cranked through by METAFONT will be done in the special effect specified.

The simplest effect is reverse video. A step-by-step consideration of the creation of such a font should make the preceding generalities much clearer. I created a file called *pattern_reversevideo.mf* which contains the following definition of *pattern*:

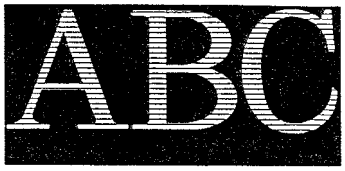
```
def pattern=
def endchar=
cullit;
picture NormalChar;
    NormalChar=currentpicture;
clearit;
fill (0,-desc-2vo)--(w+ho,-desc-2vo)--
    (w+ho,cap+2vo)--(0,cap+2vo)--cycle;
picture BlackBox;
    BlackBox:=currentpicture;
picture ReverseVideo;
    ReverseVideo=BlackBox-NormalChar;
currentpicture:=ReverseVideo;
% The rest is from standard endchar
scantokens extra_endchar;
chardx:=w;
shipit;
if displaying>0:  showit; fi
endgroup;
enddef;
enddef;
```

After loading all the normal font- and style-specific stuff, but *before* inputting any character descriptions, I input this file and then call *pattern*; the effect of this is to ensure that the **endchar** routine defined therein is run at the conclusion of each character.

So, prior to executing this **endchar**, METAFONT has just drawn a character in the usual manner. We recall that, depending upon the way in which the drawing was done, a blackened pixel may have any value greater than or equal to 1, and a white pixel any value less than or equal to zero. Since this can cause complications later as we add and subtract **pictures**, the first thing to do is a bit of housekeeping: all black bits are set equal to 1, and all white bits are set equal to zero. In other words, we capture the image of the character we have drawn in a **picture** variable *NormalChar* which is composed solely of 0's and 1's. Then, **currentpicture** is zeroed out, in preparation for a new drawing. We fill the whole letter grid completely – that is, set all bits to 1 – and set a **picture** variable *BlackBox* equal to the thus blackened grid. We then create yet another **picture** variable *ReverseVideo* and set it equal to *BlackBox* less *NormalChar*; i.e., the grid of all ones minus the grid with ones only at pixels that are part of the character. The result is:

An obvious and easy variation on this theme is a

greyed reverse video font. One way of achieving this is with a striped overlay:



We can produce this by creating a file which is called `pattern_reversegreyed` which replaces the line `currentpicture:=ReverseVideo;` in the preceding code with these lines:

```
clearit;
pickup MinPen;
for f=-desc-2vo step HugeStep
  until cap+2vo:
    draw (0,f)--(w,f);
endfor;
picture StripeOverlay;
    StripeOverlay=currentpicture;
currentpicture:=
    StripeOverlay+ReverseVideo;
```

which simply defines and draws a **picture** *StripeOverlay* which is added to the **picture** variable for the reverse video character as drawn in the same way as above. If we produce a new pattern file wherein we change that last line of code above to:

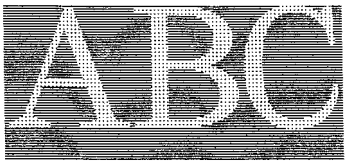
```
currentpicture:=
    StripeOverlay-ReverseVideo;
```

we get:

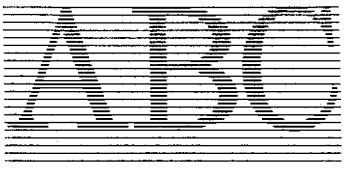
Extending the basic idea here just a bit, it seems that we can put a character filled with one pattern on a background filled with another. The following is the salient portion of the code which defines *pattern_stripendot*, a character filled with dots on a striped background. (We may assume that **pictures** *NormalChar* and *ReverseVideo* have been defined as in the preceding examples.)

```
% pattern one:  the background pattern
pickup MinPen;
for f=-desc-vo step MedStep
  until cap+vo:
    draw  (0,f)--(w+ho,f);
 endfor;
currentpicture:=
    currentpicture-NormalChar;
cullit;
picture StripedGround;
  StripedGround:=currentpicture;
clearit;
% pattern two:  the character fill
pickup PinPointPen;
for g=0 step BigStep until w:
  for f=-desc-vo step BigStep
    until cap+vo:
      draw  (g,f);
 endfor;
endfor;
cullit;
currentpicture:=
    currentpicture-ReverseVideo;
cullit;
picture DottedChar;
    DottedChar:=currentpicture;
clearit;
currentpicture:=
    DottedChar+StripedGround;
```

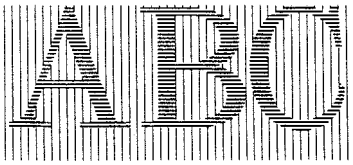
and yields:



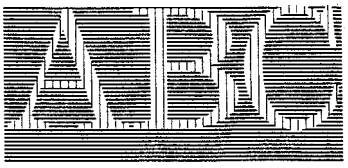
Clearly, by modifying what goes in the slots for pattern one and pattern two, we can produce



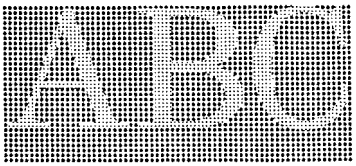
or



or



or even



When I was outlining this article, I had intended to say something along the lines of 'you are limited only by your cleverness in coding patterns' at this juncture; but as I produced the patterned fonts to use, I discovered that you are also limited by METAFONT's capacity. In particular, the vertically oriented patterns fly in the face of the underlying idea of GF files, and the GF files get very big very fast.

* * * * * * * * *

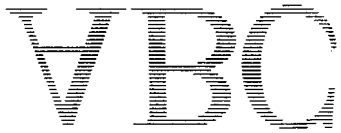
In addition to doing arithmetic with **picture** variables to produce pattern filled characters, we can also manipulate **pictures** with rotations and shifts. We can produce a mirror image font

with a shifted reflection of *NormalChar*:

```
picture MirrorImage;
    MirrorImage:=NormalChar
    reflectedabout ((0,0),(0,h))
    shifted (w,0);
```

A reverse video mirror image is accomplished by producing *ReverseVideo* in the same way as shown in the first example, and then doing a **reflectedabout** followed by **shifted** as before:

And, not to belabor the point, the same approach can produce any random pattern **reflectedabout** any line (that does not result in a transformation that METAFONT deems "too hard") and **shifted** any amount.



* * * * * * * * *

Finally, I combined the two approaches – arithmetic on **picture** variables and manipulation of them via **shifts** – to get a 'pseudo-shadow' effect. First, I **shifted** the character image to the right and down and subtracted the non-shifted character image from that shadow (giving the **picture** *ShadowOnly*). Then I laid a pattern-filled character on top. More precisely, I said:

```
f:=-desc-2vo;
pickup MinPen;
for f=-desc-2vo step HugeStep
  until cap+2vo:
    draw  (0+ho,f)--(w,f);
endfor;
currentpicture:=
    currentpicture-ReverseVideo;
cullit;
picture StripedChar;
    StripedChar:=currentpicture;
clearit;
currentpicture:=NormalChar
  shifted (.5ucHairP,-.5ucHairP);
picture Shadow;
    Shadow:=currentpicture;
clearit;
currentpicture:=Shadow-NormalChar;
cullit;
picture ShadowOnly;
    ShadowOnly:=currentpicture;
clearit;
currentpicture:=
    ShadowOnly+StripedChar;
```

to get:

It should go without saying that any pattern may be used for the shifted shadow image, and any pattern may be used for the non-shifted image; likewise, the shifting may be in any direction, although the amount will doubtless be small in any case.

You'll want to use discretion in combining patterns and shifts: such combinations quickly lead to effects that are not so much 'special' as rather woozy, as

in this dotted character with a horizontally striped shadow

or as in this font I call *HangOver*:

As a reaction to such excesses, I like the rather ethereal look of a shifted black shadow with *no* pattern in the character:

* * * * * * * * *

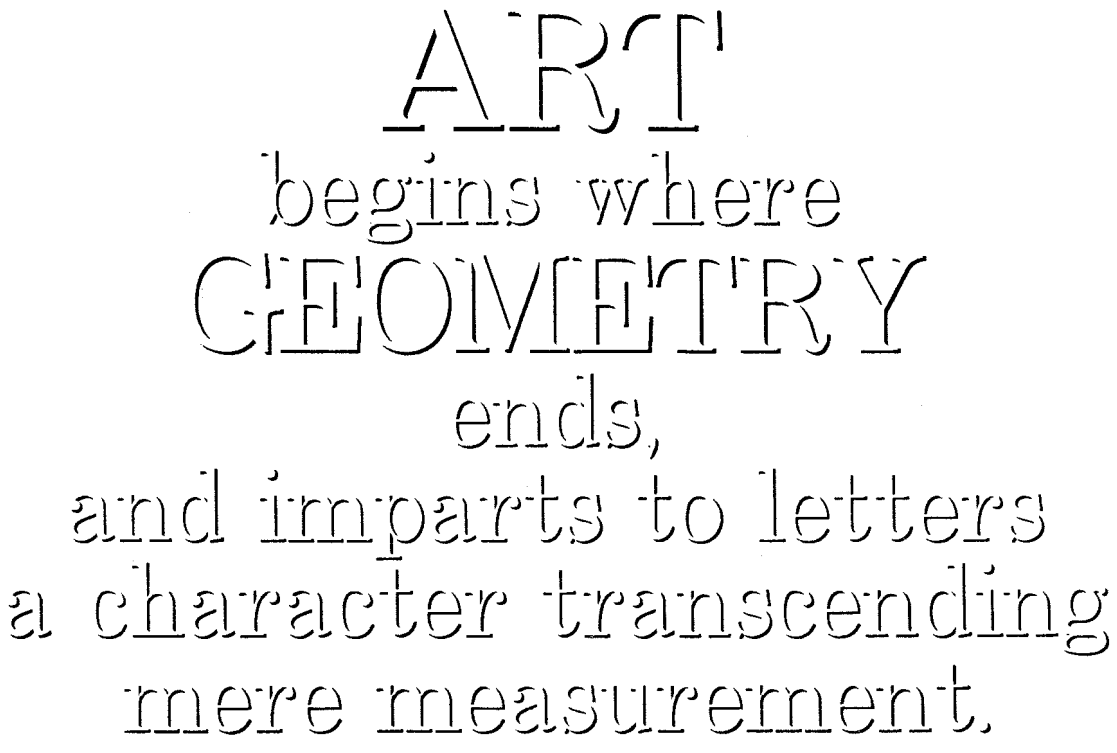
The code for each pattern given here can be applied to *any* font: I can do italic or bold Schoolbook as well as the text Schoolbook shown in the samples with no changes. A completely different face – my decorative Uncial face

or a sans serif I'm working on

or my experimental Hebrew

– will need only minimal changes, viz. the names of pens and character parts; the same is true of CMR fonts.

I hope that these samples will serve as a springboard for my readers to generate special effect fonts of their own.



This document originally printed at 300 dpi with Metafoundry fonts.

Blackboard Bold

Robert Messer
Albion College

Mathematicians require distinctive notation to denote the natural numbers, the integers, the rational, real, and complex number systems. The standard convention calls for bold face characters to denote the underlying sets. The blackboard technique of indicating bold face by doubling one of the strokes of the character has in turn influenced the printed form of these symbols. Upper case letters in the blackboard bold style are available in $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX (see page 126 of *TUGboat*, Vol. 6, No. 3), but they are not among the Computer Modern fonts.

Donald Knuth produced the entry $\mathbb{R}$ in the index of *The TEXbook* with `$\rm I\!R$`, where `\!` is the `plain.tex` macro for a negative thinspace (an mskip of −3 mu, equivalent to approximately −.17 em). This trick yields reasonable blackboard bold characters for letters with a vertical stroke on the left side. For other letters, judiciously chosen vertical rules work nearly as well.

Dimensions for kerning and vertical rules have been empirically determined for the set of blackboard bold uppercase letters illustrated and defined below. These are based on the Computer Modern Roman 10 point font (`cmr10`) at `\magstephalf`. Adjustments are necessary for other fonts and at large magnifications.

These constructions are not a perfect substitute for a blackboard bold font. At a resolution of 300 dots per inch, discretization errors result in noticeable differences in the relative positions of the various parts of the characters. Fortunately, occurrences of these symbols are seldom near enough together to invite close comparison.

$\mathbb{A}\mathbb{A}\mathbb{A}$ $\mathbb{B}\mathbb{B}\mathbb{B}$ $\mathbb{C}\mathbb{C}\mathbb{C}$ $\mathbb{D}\mathbb{D}\mathbb{D}$ $\mathbb{E}\mathbb{E}\mathbb{E}$

$\mathbb{F}\mathbb{F}\mathbb{F}$ $\mathbb{G}\mathbb{G}\mathbb{G}$ $\mathbb{H}\mathbb{H}\mathbb{H}$ $\mathbb{I}\mathbb{I}\mathbb{I}$ $\mathbb{J}\mathbb{J}\mathbb{J}$

$\mathbb{K}\mathbb{K}\mathbb{K}$ $\mathbb{L}\mathbb{L}\mathbb{L}$ $\mathbb{M}\mathbb{M}\mathbb{M}$ $\mathbb{N}\mathbb{N}\mathbb{N}$ $\mathbb{O}\mathbb{O}\mathbb{O}$

$\mathbb{P}\mathbb{P}\mathbb{P}$ $\mathbb{Q}\mathbb{Q}\mathbb{Q}$ $\mathbb{R}\mathbb{R}\mathbb{R}$ $\mathbb{S}\mathbb{S}\mathbb{S}$ $\mathbb{T}\mathbb{T}\mathbb{T}$

$\mathbb{U}\mathbb{U}\mathbb{U}$ $\mathbb{V}\mathbb{V}\mathbb{V}$ $\mathbb{W}\mathbb{W}\mathbb{W}$ $\mathbb{X}\mathbb{X}\mathbb{X}$ $\mathbb{Y}\mathbb{Y}\mathbb{Y}$

$\mathbb{Z}\mathbb{Z}\mathbb{Z}$

```
% Poor person's blackboard bold
% January 6, 1988

% Robert Messer
% Department of Mathematics
% Albion College
% Albion, MI 49224
% Bitnet: RAM@ALBION

\def\A{{\rm\kern.22em
   \vrule width.02em
       height0.5ex depth 0ex
   \kern-.24em A}}
\def\B{{\rm I\kern-.25em B}}
\def\C{{\rm\kern.24em
   \vrule width.02em
       height1.4ex depth-.05ex
   \kern-.26em C}}
\def\D{{\rm I\kern-.25em D}}
\def\E{{\rm I\kern-.25em E}}
\def\F{{\rm I\kern-.25em F}}
\def\G{{\rm\kern.24em
   \vrule width.02em
       height1.4ex depth-.05ex
   \kern-.26em G}}
\def\H{{\rm I\kern-.25em H}}
\def\I{{\rm I\kern-.25em I}}
\def\J{{\rm\kern.19em
   \vrule width.02em
       height1.47ex depth 0ex
   \kern-.21em J}}
\def\K{{\rm I\kern-.25em K}}
\def\L{{\rm I\kern-.25em L}}
\def\M{{\rm I\kern-.23em M}}
\def\N{{\rm I\kern-.23em N}}
\def\O{{\rm\kern.24em
   \vrule width.02em
       height1.4ex depth-.05ex
   \kern-.26em O}}
\def\P{{\rm I\kern-.25em P}}
\def\Q{{\rm\kern.24em
   \vrule width.02em
       height1.4ex depth-.05ex
   \kern-.26em Q}}
\def\R{{\rm I\kern-.25em R}}
\def\S{{\rm\kern.18em
   \vrule width.02em
       height1.47ex depth-.9ex
   \kern.12em
   \vrule width.02em
       height0.7ex depth 0ex
   \kern-.34em S}}
```

```
\def\T{{\rm\kern.45em
   \vrule width.02em
       height1.47ex depth 0ex
   \kern-.47em T}}
\def\U{{\rm\kern.30em
   \vrule width.02em
       height1.47ex depth-.05ex
   \kern-.32em U}}
\def\V{{\rm\kern.27em
   \vrule width.02em
       height1.47ex depth-.8ex
   \kern-.29em V}}
\def\W{{\rm\kern.25em
   \vrule width.02em
       height1.47ex depth-.9ex
   \kern.34em
   \vrule width.02em
       height1.47ex depth-.9ex
   \kern-.63em W}}
\def\X{{\rm\kern.30em
   \vrule width.02em
       height1.47ex depth-1ex
   \kern.12em
   \vrule width.02em
       height0.4ex depth 0ex
   \kern-.46em X}}
\def\Y{{\rm\kern.25em
   \vrule width.02em
       height1.0ex depth 0ex
   \kern-.27em Y}}
\def\Z{{\rm\kern.26em
   \vrule width.02em
       height0.5ex depth 0ex
   \kern.04em
   \vrule width.02em
       height1.47ex depth-1ex
   \kern-.34em Z}}

% Some dimensions will require
% adjustments at large magnifications.

% Test print
\nopagenumbers
\tabskip=0pt plus1fil
\halign to18.75pc{&\setbox0=\hbox{#}%
  \hfil\copy0\copy0\box0\hfil\cr\cr
\A&\B&\C&\D&\E\cr\cr
\F&\G&\H&\I&\J\cr\cr
\K&\L&\M&\N&\O\cr\cr
\P&\Q&\R&\S&\T\cr\cr
\U&\V&\W&\X&\Y\cr\cr
  &  &\Z\cr\cr}
```

Using Greek Fonts with TEX

Silvio Levy
Princeton University

In this document I hope to show that typesetting Greek in TEX using the gr family of fonts can be as easy as typesetting English text, and leads to equally good results. This is meant to be a tutorial, not an exhaustive discussion; some TEXnical remarks that should be useful after the reader has acquired some familiarity with the fonts are printed in fine print.

The Alphabet

In order to typeset Greek text, you need to go into "Greek mode." This is achieved by typing `\begingreek` anywhere in your document; Greek mode will remain in effect until you type a matching `\endgreek`. While in Greek mode, the letters 'a' to 'z' and 'A' to 'Z' come out as Greek letters, according to the following code:

α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	φ	χ	ψ	ω
a	b	g	d	e	z	h	j	i	k	l	m	n	x	o	p	r	s	t	u	f	q	y	w

There is no digamma yet. The same character 's' will print as 'σ' or 'ς', depending on its position in a word.

The system does this by accessing a ligature of 's' with any other letter that follows it. If, for some reason, you want to print an initial/medial sigma by itself (as in the table above), or at the end of a word, you should type 'c'.

Try to typeset some simple text now. Create a file containing the following lines:

```
\input greekmacros % where \begingreek and
%                  % other commands are defined
This is English text.
\begingreek
This is Greek text.
\endgreek
```

When you TEX this file, you get the following gibberish:

This is English text. Τηις ις Γρεεκ τεξτ.

If you say `\greekdelims` near the top of your file, the character $ can be used in place of both `\begingreek` and `\endgreek`. The control sequence `\math` takes on the former meaning of $.

Editor's note: Thanks to Doug Henderson for METAFONTing the typesetter fonts from Silvio Levy's .MF source files. Doug was also responsible for the typesetter renditions of the APL fonts in TUGboat 8#3.

Accents and Breathings

To get an acute, grave or circumflex accent over a vowel, type ', ` or ~, respectively, before the vowel. To get a rough or smooth breathing, type < or > before the vowel (or rho) and any accent that it may have. To get an iota subscript, type | *after* the vowel. A diaeresis is represented by ", and if accompanied by an accent it can come before or after the accent.

For example, `>en >arq~h| >~hn <o l'ogos` gives ἐν ἀρχῇ ἦν ὁ λόγος. Neat, ain't it?

Accents and breathings, too, are typeset by means of ligatures: a vowel with a breathing, an accent and iota subscript, for example, is realized as a four-character ligature. The only exception is when a breathing is followed by a grave accent, in which case the breathing + accent combination is typeset as a TEX `\accent` over the vowel. This means that words containing such combinations cannot be hyphenated in (standard) TEX; but this is not a problem because, with the exception of very rare cases of crasis, all such words are monosyllables.

Punctuation

Here's the table of correspondences for punctuation:

```
. , · : ! ; ' « »
. , ; : ! ? '' (( ))
```

The last three entries represent the apostrophe and quotations marks. The other available non-letters are the ten digits, parentheses, brackets, hyphen, em- and en-dashes, slash, percent sign, asterisk, plus and equal signs. All of these are accessible in the same way as under plain TEX. In a future release there will be tick marks for numbers ($\alpha' = 1$, $,\alpha = 1000$).

Hyphenation

A hyphenation table for both modern and ancient Greek is currently being debugged. For now one can use plain TEX with its (English) hyphenation table, which gives the right results about 90% of the time (amazing, isn't it?). Be sure to proofread your text carefully, unless you've turned hyphenation off.

In standard TEX, only one hyphenation dictionary can be used in a job. Thus, even with a Greek hyphenation table, a file that combines Greek and English text is likely to be incorrectly hyphenated. Michael Ferguson's extension to TEX handles multiple hyphenation tables, and hyphenates words containing `\accents` (cf. the previous fine print paragraph).

Interaction with other macros

While in Greek mode you can do just about everything that you can outside: go into math mode, create boxes, alignments, and so on. The file `greekmacros.tex` sets things up so that in Greek mode the control sequences `\tt` and `\bf` switch to a typewriter and a bold Greek font, respectively: thus `{\tt s''>agap~w}` gives σ'ἀγαπῶ. (Try it.) On the other hand, there are no "italic" or slanted Greek fonts, so `\it` and `\sl` will give you the same fonts as outside Greek mode. The various constructions under AMS-TEX and LATEX for increasing or decreasing point sizes don't work yet; they will in a future release.

The characters that form diacritics (<, >, ', `, ~, " and |) are treated differently depending on whether or not you're in Greek mode. More exactly, under plain TEX these characters (with the exception of ~) have a `\catcode` of 12: they print as themselves, and they cannot appear in control words. But in Greek mode ', `, ~, " and | are "letters", that is, they have a `\catcode` of 11, while < and > are active, with a `\catcode` of 13. This may be important even for beginners because it means that ', for example, can be taken as part of a control word. Thus the sequence

```
\begingreek
\line{wm'ega\hfil'alfa}
\endgreek
```

will cause an error message about an undefined control sequence `\hfil'alfa`, instead of printing

ωμέγα ἄλφα

as you might expect. (I hope classicists will forgive this use of the modern Greek one-accent system.) The solution, of course, is to remember to add a blank after the `\hfil`.

A more subtle problem arises when you use Greek text in macro arguments, if the arguments are scanned while you're outside Greek mode. This is because TEX assigns `\catcodes` to tokens as it first reads them, so when the argument is plugged into the body of the macro the characters above have the wrong `\catcode`. If the legendary Jonathan Horatio Quick were to write

```
\def\hellenize#1{\begingreek #1\endgreek}
                \hellenize{d'uo >'h tre~is,}
```

he would be unpleasantly surprised by the following output:

δύο ᾽η τρε ις,

which can be explained as follows: the ~, which should be a letter, is seen as an active character, and expands to a blank as in plain TEX; while the breathing, which should be active, is not, and in particular it doesn't do the right thing when next to the grave accent. Solutions to this problem require a bit of wizardry, and will not be discussed here; see, for example, Reinhard Wonneberger's article in the October, 1986 issue of *TUGboat*, especially pages 179–180.

Test of grreg10 on March 1, 1988 at 1352

Τὴ στιγμὴ τούτη νιώθω πόσο βαρύ 'ναι τὸ μυστήριο τῆς ξομολόγησης. Ὡς τώρα, κανεὶς δὲν ξέρει πῶς πέρασα τὰ δυὸ χρόνια μου στὸ Ἅγιον Ὄρος. Οἱ φίλοι μου θαρροῦν πὼς πῆγα νὰ δῶ βυζαντινὰ κονίσματα ἢ ἀπὸ μυστικοπάθεια νὰ ζήσω μιὰ περασμένη ἐποχή. Καὶ τώρα, νά, ντρέπομαι νὰ μιλήσω.
Πῶς νὰ τὸ πῶ; Θυμοῦμαι ἕνα ἀνοιξιάτικο δειλινό, ποὺ κατέβαινα τὸν Ταΰγετο, μιὰ ξαφνικὴ θύελλα μὲ κύκλωσε κοντὰ στούς Πενταυλούς. Τόσο φοβερὸς ἀνεμοσίφουνας, ποὺ ἔπεσα καταγῆς γιὰ νὰ μὴν γκρεμιστῶ. Οἱ ἀστραπὲς μ' ἔξωσαν ὁλοῦθε κι ἔκλεισα τὰ μάτια μὴν τυφλωθῶ, καὶ κατάχαμα, πίστομα, περίμενα. Ὅλο τὸ πανύψηλο βουνὸ ἔτρεμε, καὶ δυὸ ἔλατα δίπλα μου τσακίστηκαν ἀπ' τὴ μέση καὶ βρόντηξαν χάμου. Ἐνιωθα τὸ θειάφι τοῦ κεραυνοῦ στὸν ἀέρα, καὶ ξαφνικὰ ξέσπασε ἡ μπόρα, ἔπεσεν ὁ ἄνεμος, καὶ χοντρές, θερμὲς στάλες βροχὴ χτύπησαν τὰ δεντρὰ καὶ τὸ χῶμα. Τὸ θυμάρι, ἡ θρούμπα, τὸ φασκόμηλο, τὸ φλισκούνι, χτυπημένα ἀπ' τὸ νερό, τίναξαν τὶς μυρωδιές τους κι ὅλη ἡ γῆς μύρισε.

(From Kazantzakis' "Symposium")

Ἀλλ' ἀκούσονται, ἐάνπερ εὖ δοκῇς λέγειν. τόδε δέ σου ἐνενόησα ἅμα λέγοντος, καὶ πρὸς ἐμαυτὸν σκοπῶ· εἰ ὅτι μάλιστά με Εὐθύφρων διδάξειεν, ὡς οἱ θεοὶ ἅπαντες τὸν τοιοῦτον θάνατον ἡγοῦνται ἄδικον εἶναι, τί μᾶλλον ἐγὼ μεμάθηκα παρ' Εὐθύφρονος, τί ποτ' ἐστὶν τὸ ὅσιόν τε καὶ τὸ ἀνόσιον; θεομισὲς μὲν γὰρ τοῦτο τὸ ἔργον, ὡς ἔοικεν, εἴη ἄν· ἀλλὰ γὰρ οὐ τούτῳ ἐφάνη ἄρτι ὡρισμένα τὸ ὅσιον καὶ μή· τὸ γὰρ θεομισὲς ὂν καὶ θεοφιλὲς ἐφάνη. ὥστε τούτου μὲν ἀφίημί σε, ὦ Εὐθύφρον· εἰ βούλει, πάντες αὐτὸ· ἡγείσθων θεοὶ ἄδικον καὶ πάντες μισούντων. ἀλλ' ἆρα τοῦτο νῦν ἐπανορθώμεθα ἐν τῷ λόγῳ, ὡς ὃ μὲν ἂν πάντες οἱ θεοὶ μισῶσιν, ἀνόσιόν ἐστιν, ὃ δ' ἂν φιλῶσιν, ὅσιον· ὃ δ' ἂν οἱ μὲν φιλῶσιν, οἱ δὲ μισῶσιν, οὐδέτερα ἢ ἀμφότερα; ἆρ' οὕτω βούλει ἡμῖν ὡρίσθαι νῦν περὶ τοῦ ὁσίου καὶ τοῦ ἀνοσίου;

(From Plato's "Euthyphro")

Test of grreg10 on February 6, 1988 at 1647

Τὴ στιγμὴ τούτη νιώθω πόσο βαρύ 'ναι τὸ μυστήριο τῆς ξομολόγησης. Ὡς τώρα, κανεὶς δὲν ξέρει πῶς πέρασα τὰ δυὸ χρόνια μου στὸ Ἅγιον Ὄρος. Οἱ φίλοι μου θαρροῦν πὼς πῆγα νὰ δῶ βυζαντινὰ κονίσματα ἢ ἀπὸ μυστικοπάθεια νὰ ζήσω μιὰ περασμένη ἐποχή. Καὶ τώρα, νά, ντρέπομαι νὰ μιλήσω.
Πῶς νὰ τὸ πῶ; Θυμοῦμαι ἕνα ἀνοιξιάτικο δειλινό, ποὺ κατέβαινα τὸν Ταΰγετο, μιὰ ξαφνικὴ θύελλα μὲ κύκλωσε κοντὰ στούς Πενταυλούς. Τόσο φοβερὸς ἀνεμοσίφουνας, ποὺ ἔπεσα καταγῆς γιὰ νὰ μὴν γκρεμιστῶ. Οἱ ἀστραπὲς μ' ἔξωσαν ὁλοῦθε κι ἔκλεισα τὰ μάτια μὴν τυφλωθῶ, καὶ κατάχαμα, πίστομα, περίμενα. Ὅλο τὸ πανύψηλο βουνὸ ἔτρεμε, καὶ δυὸ ἔλατα δίπλα μου τσακίστηκαν ἀπ' τὴ μέση καὶ βρόντηξαν χάμου. Ἔνιωθα τὸ θειάφι τοῦ κεραυνοῦ στὸν ἀέρα, καὶ ξαφνικὰ ξέσπασε ἡ μπόρα, ἔπεσεν ὁ ἄνεμος, καὶ χοντρές, θερμὲς στάλες βροχὴ χτύπησαν τὰ δεντρὰ καὶ τὸ χῶμα. Τὸ θυμάρι, ἡ θρούμπα, τὸ φασκόμηλο, τὸ φλισκούνι, χτυπημένα ἀπ' τὸ νερό, τίναξαν τὶς μυρωδιές τους κι ὅλη ἡ γῆς μύρισε.

(From Kazantzakis' "Symposium")

Ἀλλ' ἀκούσονται, ἐάνπερ εὖ δοκῇς λέγειν. τόδε δέ σου ἐνενόησα ἅμα λέγοντος, καὶ πρὸς ἐμαυτὸν σκοπῶ· εἰ ὅτι μάλιστά με Εὐθύφρων διδάξειεν, ὡς οἱ θεοὶ ἅπαντες τὸν τοιοῦτον θάνατον ἡγοῦνται ἄδικον εἶναι, τί μᾶλλον ἐγὼ μεμάθηκα παρ' Εὐθύφρονος, τί ποτ' ἐστὶν τὸ ὅσιόν τε καὶ τὸ ἀνόσιον; θεομισὲς μὲν γὰρ τοῦτο τὸ ἔργον, ὡς ἔοικεν, εἴη ἄν· ἀλλὰ γὰρ οὐ τούτῳ ἐφάνη ἄρτι ὡρισμένα τὸ ὅσιον καὶ μή· τὸ γὰρ θεομισὲς ὂν καὶ θεοφιλὲς ἐφάνη. ὥστε τούτου μὲν ἀφίημί σε, ὦ Εὐθύφρον· εἰ βούλει, πάντες αὐτὸ ἡγείσθων θεοὶ ἄδικον καὶ πάντες μισούντων. ἀλλ' ἆρα τοῦτο νῦν ἐπανορθώμεθα ἐν τῷ λόγῳ, ὡς ὃ μὲν ἂν πάντες οἱ θεοὶ μισῶσιν, ἀνόσιόν ἐστιν, ὃ δ' ἂν φιλῶσιν, ὅσιον· ὃ δ' ἂν οἱ μὲν φιλῶσιν, οἱ δὲ μισῶσιν, οὐδέτερα ἢ ἀμφότερα; ἆρ' οὕτω βούλει ἡμῖν ὡρίσθαι νῦν περὶ τοῦ ὁσίου καὶ τοῦ ἀνοσίου;

(From Plato's "Euthyphro")

These text samples were set on a Xerox 4500; thanks to Tom Reid for the METAFONT processing.

	′0	′1	′2	′3	′4	′5	′6	′7	
′00x	–	σα	σβ	σ’	σδ	σε	σφ	σγ	"0x
′01x	ση	σι	σθ	σκ	σλ	σμ	σν	σο	
′02x	σπ	σχ	σρ	σσ	στ	συ		σω	"1x
′03x	σξ	σψ	σζ		‘	’	˘	–	
′04x	¨	!	¨	˛	˚	%		′	"2x
′05x	(	)	*	+	,	-	.	/	
′06x	0	1	2	3	4	5	6	7	"3x
′07x	8	9	:	·	¸	=	˝	;	
′10x	˚	A	B	˛	Δ	E	Φ	Γ	"4x
′11x	H	I	Θ	K	Λ	M	N	O	
′12x	Π	X	P	Σ	T	ϒ	˚	Ω	"5x
′13x	Ξ	Ψ	Z	[	˚	]	˝	¸	
′14x	‘	α	β	σ	δ	ε	φ	γ	"6x
′15x	η	ι	θ	κ	λ	μ	ν	ο	
′16x	π	χ	ρ	ς	τ	υ		ω	"7x
′17x	ξ	ψ	ζ	«	¸	»	˜	—	
′20x	ὰ	ά	ᾰ	σὰ	ᾲ	ᾴ	ᾶ	σᾳ	"8x
′21x	ά	ᾰ	ᾰ	σά	ᾴ	ᾶ	ᾶ	σᾴ	
′22x	ᾱ	ᾰ	ᾰ	σᾱ	ᾱ	ᾶ	ᾶ	σᾱ	"9x
′23x	ὴ	ή	ή	σὴ	ῆ	ῆ	ῆ	ση	
′24x	ή	ῆ	ῆ	σή	ή	ῆ	ῆ	σή	"Ax
′25x	η̄	ῆ	ῆ	ση̄	η̄	ῆ	ῆ	ση̄	
′26x	ὼ	ώ	ὼ	σὼ	ῲ	ῶ	ῶ	σῳ	"Bx
′27x	ώ	ῶ	ῶ	σώ	ῴ	ῶ	ῶ	σῴ	
′30x	ω̄	ῶ	ῶ	σω̄	ῳ̄	ῶ	ῶ	σῳ̄	"Cx
′31x	ὶ	ί	ϊ	σὶ	ὺ	ύ	ῦ	σὺ	
′32x	ί	ΐ	ΐ	σί	ύ	ῦ	ῦ	σύ	"Dx
′33x	ῑ	ΐ	ΐ	σῑ	ῡ	ῦ	ῦ	σῡ	
′34x	ὲ	έ	ἑ	σὲ	ὸ	ό	ὸ	σὸ	"Ex
′35x	έ	ἕ	ἕ	σέ	ό	ὅ	ὅ	σό	
′36x	ῑ	ῒ	ῒ	ῑ	ϋ	ὐ	ὐ	ῡ	"Fx
′37x	ᾳ	ῃ	ῳ	ῥ	ῥ	σ‘	σ’	σ¯	
	"8	"9	"A	"B	"C	"D	"E	"F	

Output Devices

TEX Output Devices

Don Hosek

The device tables on the following pages list all the TEX device drivers currently known to TUG. Some of the drivers indicated in the tables are considered proprietary. Most are not on the standard distribution tapes; those drivers which are on the distribution tapes are indicated in the listing of sources below. 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. When information on the drivers is available, it is included below.

Screen previewers for multi-user computers are listed in the section entitled "Screen Previewers". If a source has been listed previously under "Sources", then a reference is made to that section for names of contacts, etc.

Corrections, updates, and new information for the list are welcome; send them to Don Hosek, Bitnet DHOSEK@HMCVAX (postal address, 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

The programs listed under Adelaide have been submitted to the standard distributions for the appropriate computers. The PostScript driver permits inclusion of PostScript files in a TEX file. The driver is described in *TUGboat, Vol. 8, No. 1.*

AMS American Mathematical Society, Barbara Beeton, 401-272-9500 Arpanet: BNB@Seed.AMS.com

Arbor ArborText, Inc., Bruce Baker, 313-996-3566, Arpanet: bwb%arbortext@umix.cc.umich.edu

ArborText's software is proprietary and ranges in price from $150 to $3000. The drivers for PostScript printers, the HP LaserJet Plus, the QMS Lasergrafix, and Imagen printers are part of their DVILASER series. The drivers all support graphics and include other special features such as use of resident fonts or landscape printing when supported by the individual printers.

Printing on the Autologic APS-5 and μ-5 phototypesetters with DVIAPS includes support of Autologic standard library fonts and logo processing.

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

Addison-Wesley supports graphics on all Macintosh software, and on Imagen, PostScript, and QMS laser printers on the IBM PC.

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

Caltech California Institute of Technology, Chuck Lane, Bitnet: CEL@CITHEX

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

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

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

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

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

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

The LN03 driver is on the VAX/VMS distribution tape.

ENS Ecole Normale Superieure, Chantal Durand, Centre de Calcul, Ecole Normale Superieure, 45 rue d'Ulm, 75005 Paris, France

GA Tech GA Technologies

GMD1 Gesellschaft für Mathematik und Datenverarbeitung, Federal Republic of Germany, Ferdinand Hommes, Bitnet: GRZTEX@DBNGMD21, 0228-303221

GMD2 Gesellschaft für Mathematik und Datenverarbeitung, Federal Republic of Germany, Dr. Wolfgang Appelt, uucp: seismo!unido!gmdzi!zi.gmd.dbp.de!appelt

Heidelb'g University of Heidelberg, Federal Republic of Germany, Joachim Lammarsch, Bitnet: RZ92@DHDURDZ1

HMC Harvey Mudd College, Don Hosek, Bitnet: Dhosek@Ymir

HP Hewlett-Packard, Stuart Beatty, 303-226-3800

INFN INFN/CNAF, Bologna, Italy, Maria Luisa Luvisetto, 51-498286, Bitnet: MILTEX@IBOINFN

The CNAF device drivers are on the VAX/VMS distribution tape.

Interg'ph Intergraph, Mike Cunningham, 205-772-2000

JDJW JDJ Wordware, John D. Johnson, 415-965-3245, Arpanet: M.JOHN@Sierra.Stanford.Edu

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

The VAX/VMS Imagen driver supports graphics.

Kettler Kettler EDV Consulting, P. O. Box 1345, D-8172 Lenggries, Federal Republic Germany, +49 8042 8081

The LaserJet driver supports graphics inclusion in device dependent format. PK font files are used. This program is proprietary. Contact Kettler for further information.

LaserPrint LaserPrint, P. O. Box 35, D-6101 Fränkisch Crumbach, Federal Republic Germany, +49 6164 4044

The driver supports graphics inclusion in device dependent format. PK font files are used. This program is proprietary. Contact LaserPrint for further information.

LLL Lawrence Livermore Laboratory

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

Milan1 Università Degli Studi Milan, Italy, Dario Lucarella, 02/23.62.441

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

MIT Massachusetts Institute of Technology, Chris Lindblad, MIT AI Laboratory, 617-253-8828

The drivers for Symbolics Lisp machines use the Symbolics Generic Hardcopy interface as a back end, so it should work on any printer that has a driver written for it. The printers listed in the table indicate drivers the program has been tested on.

The UNIX drivers for PostScript and QMS printers both support landscape printing and graphics inclusion via specials.

MPAE Max-Planck-Institut für Aeronomie, H. Kopka, (49) 556-41451, Bitnet: MI040L@D606WD01

MR Math Reviews, Dan Latterner, 313-996-5266

NJIT New Jersey Institute of Technology

OCLC OCLC, Tom Hickey, 6565 Frantz Road, Dublin, OH 43017, 616-764-6075

OSU2 Ohio State University, Ms. Marty Marlatt, Department of Computer and Information Science, 2036 Neil Avenue, Columbus, OH 43210

The drivers are distributed on either ANSI or TOPS-20 DUMPER tapes, with hardcopy documentation. There is a $125 service charge (payable to Ohio State University) to cover postage, handling, photocopying, etc.

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

Graphic output is supported on Imagen, PostScript, and QMS printers.

Philips Philips Kommunikations Industrie AG, TEKADE Fernmeldeanlagen, Attn. Dr. J. Lenzer, Thurn-und-Taxis-Str., D-8500 Nürnberg, Federal Republic Germany, +49 911 5262019

PPC Princeton Plasma Physics Lab, Charles Karney, Arpanet: Karney%PPC.MFENET@NMFECC.ARPA

Versatec output from TEXspool is produced via the NETPLOT program. TEXspool also produces output for the FR80 camera. Color and graphics primitives are supported through specials.

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

RTI Research Triangle Institute, Randy Buckland, Arpanet: rcb@rti.rti.org

The program is available in the comp.sources.misc archives on Arpanet and Usenet.

Saar Universität des Saarlandes, Saarbrücken, Federal Republic of Germany, Prof. Dr. Reinhard Wilhelm, uucp: wilhelm@sbsvax.UUCP

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

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

Sci Ap Science Applications, San Diego, CA, 619-458-2616

SEP Systemhaus für Elektronisches Publizieren, Robert Schöninger, Arndtstrasse 12, 5000 Köln, Federal Republic of Germany

DVIP400 uses PXL files. Landscape printing is supported in all versions and graphics inclusion in all but the IBM PC version. Source is available on request. Cost varies from 300–1848DM.

Stanford Stanford University

The Imagen driver from Stanford is present on most distributions as the file DVIIMP.WEB. It provides limited graphics ability.

Sun Sun, Inc.

Sydney University of Sydney, Alec Dunn, (02) 692 2014, ACSnet: alecd@facet.ee.su.oz

Talaris Talaris, Rick Brown, 619-587-0787

All of the Talaris drivers support graphics.

T A&M1 Texas A&M, Bart Childs, 409-845-5470, CSnet: Childs@TAMU

Graphics is supported on the Data General drivers for the Printronix, Toshiba, and Versatec on the Data General MV. On the TI PC, graphics is supported on the Printronix and Texas Instruments 855 printers. There are also previewers available for both the Data General and the TI.

T A&M2 Texas A&M, Ken Marsh, 409-845-4940, Bitnet: KMarsh@TAMNIL

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

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

	Amdahl (MTS)	CDC Cyber	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM/CMS	IBM VM/
Agfa P400							SEP	SEP	
Canon					Utah	Utah			
DEC LN01									
DEC LN03					Utah	Utah			
Golden Laser 100					Utah	Utah			
HP LaserJet Plus					Utah	T A&M2 Utah			
IBM 38xx, 4250, Sherpa							GMD1 Heidelb'g	GMD1 Wash St	
Imagen	Arbor UBC		T A&M1	Stanford Vander	Columb. Utah	Utah	Arbor	Arbor W'mann	
Philips Elpho									
PostScript printers					Utah	Arbor Utah		Arbor	
QMS Lasergrafix	Arbor	U Wash2	T A&M1			T A&M2	Arbor GMD	Arbor GMD	
Symbolics					U Wash1				
Talaris							Talaris	Talaris	
Xerox Dover					CMU				
Xerox 2700II		Bochum			OSU2 Xerox			ENS	
Xerox 9700	Arbor U Mich						Arbor T A&M4	Arbor T A&M4	T A&

Low-Resolution Printers on Microcomputers and Workstations — Laser Xerographic, Electro-Erosion Printers

	Apollo	Atari ST	HP1000	HP3000	HP9000 200	IBM PC	Integrated Solutions	SUN
Agfa P400						SEP		
Canon		Utah				Utah	Utah	Utah
Cordata LP300						Pers		
DEC LN03		Utah				Utah Wash U	Utah	Utah
Golden Laser 100		Utah				Utah	Utah	Utah
HP 2680			JDJW	Pers				
HP 2688A			JDJW		HP			
HP LaserJet Plus	Arbor	Kettler Tools	TRC Finl'd		MPAE	Arbor Kettler Utah	Utah	Utah
Imagen	Arbor OCLC	Utah				A-W Arbor Pers Utah	Utah	Arbor Sun U Md Utah
Kyocera		LaserPrint				LaserPrint		
Philips Elpho		Kettler						
PostScript printers	Arbor				Arbor	A-W Arbor Pers Utah	Utah	Arbor MIT Utah
QMS Lasergrafix	Arbor Scan					A-W Arbor Pers		Arbor MIT U Del
Talaris						Talaris		Talaris
Xerox 9700	COS Scan							T A&M4

Typesetters

	Apollo	CDC Cyber	HP3000	IBM MVS	IBM PC	IBM VM/CMS	Siemens BS2000	Sperry 1100	SUN	U
Allied Linotype CRTronic										
Allied Linotype L100, L300P					A-W Pers					
Allied Linotype L202					Pers					
Autologic APS-5, Micro-5	COS Scan				Arbor Pers				Arbor	A
Compugraphic 8400			U Shef		Arbor Pers					
Compugraphic 8600		UNI.C			Arbor Pers	Wash St		U Wisc		
Compugraphic 8800					Arbor					
Harris 7500										S
Hell Digiset				GMD2			GMD			

An ASCII Previewer for TEX

Marcus Brown
Texas A&M University

A common complaint among users of TEX is the problem of the *edit–TEX–print* cycle, with the attendant delay of waiting on printer service. One solution has been to allow for viewing the formatted `dvi` file on the terminal. This has been effectively implemented on workstations. VorTEX for the SUN and a number of other previewers are all well documented. However, workstations are not universal for the TEX user.

Some earlier articles describe terminal previewers, but most have focused on graphics terminals, or on very low resolution representations on more common terminals such as the DEC VT100. These are generally not reasonable alternatives for the CRTs found on multi-user systems because of the limited I/O bandwidth. Most terminals are not capable of handling down-loaded fonts, and attempting a bit-map display frequently takes a minute or more to load a screen.

Basic requirements

We believe that significant time and cost could be saved by developing a previewer for the normal ASCII terminal found in most working environments. In order to construct a usable previewer, we must consider what the user typically needs to see in a preview of his document:

1. The user will want to see any obvious errors, such as overfull hboxes, misspellings or typographic errors.
2. The user will want to check on line breaks, page breaks, and the placement of floating inserts and tables.
3. The user will want to check on any use of special fonts. For example, he will want to know if a closing brace has been omitted causing the last 5 pages to be set in *italic font.*
4. The user will want to see if the corrections made to problems detected in the last run gave the desired result.

Most of these types of questions can be satisfactorily answered using an ASCII previewer, and they constitute a large enough percentage of the trial prints of a document to justify the development and use of a previewer for an ASCII terminal.

This paper was supported in part by a contract with the U.S. Forestry Service.

Design model

The following is a list of the relevant characteristics of ASCII terminals which we will use as a design model:

1. They have only a single fixed pitch font, although most have the ability to emphasize selected characters or regions of the screen by use of color, reverse-video, blinking, underlining, or brighter-than-normal characters.
2. They often have a limited graphics character set, but that is of little use in attempting to simulate typeset output such as math equations, for example. However, they usually allow for vertical and horizontal lines.
3. They normally have 24 vertical lines and 80 horizontal columns. This is obviously inadequate for viewing a typeset page, with 60 or more vertical lines and perhaps 100 horizontal characters plus margins on a typical page.

While this limited set of capabilities cannot do justice to a typeset document, it is able to provide enough information to the user to allow several types of corrections and adjustments. We propose that an ASCII terminal previewer should allow for the following options:

1. Margins. The previewer should allow viewing of the left and right margins of a page. The primary difficulty with this requirement is that the terminal does not have enough width to show a complete line, and in many cases the fixed width font obliterates TEX's efforts at right margin justification or alignment. With these two concerns in mind, the obvious solution is to allow viewing a subset of the page which shows the desired margins. We suggest three options for viewing margins. Each option is demonstrated using page 1 of the immortal *TEXbook.*

- **Left Margin.** The lines shown on the screen should begin at the left margin of the screen. All letters typeset will be shown on the screen until the point at which the line must be truncated due to screen width. This method is demonstrated in Fig. 1. Note that even when all the letters fit on the line, the right margin is not justified.
- **Right Margin.** The right margin of the screen less 1 column should coincide with the right margin of the paper. The last column is reserved for the dreaded *overfull rule* which is simulated on the first line of Fig. 2. Characters should be written to the screen in reverse order from the right margin. The effect should be

that of truncating the left portion of each line where necessary and adjusting the alignment such that the final character on each line is at its proper position on the page.

- **Both Margins.** The left and right margins should be displayed as described above. Characters in the middle of the line will be deleted or added to allow both margins to appear justified on the screen. Fig. 3 should make this option clear.

These viewing alternatives will allow for checking line breaks, indentations and related information. The user should be able to switch back and forth between these alternatives with a single key stroke and a minimum of display time.

2. Pages. The previewer should allow the display of page breaks. This could be represented by displaying the top and bottom of a single page, with the middle lines omitted. It would also be desirable to allow display of the bottom of one page, a horizontal line representing the page break, and the top of the succeeding page. (This option should be added to more advanced previewers like VorTEX.) Perhaps the most commonly used option would allow for scrolling up and down through a single page and across page breaks to previous or succeeding pages. Again, the user should be able to switch easily between these options.

3. Fonts. The abilities of the terminals mentioned above to highlight certain characters could be used. Standard Roman fonts would be shown in the default text of the terminal, while color, reverse video or other options could be used to signify bold, slanted, etc. It would be unwise to attempt too complex a coding scheme here. Probably the best scheme would be to use only one or two alternative display techniques, with a *catch-all* category for all other fonts. For example, let `\bf` be signified by using brighter text and `\it` be signified with underlined text. Then all other fonts would be signified by reverse-video. If the representation of fonts becomes too complex, the user will spend more effort remembering the coding scheme than is justified by the additional information gained. These options might be adjustable by the user for different documents.

Other issues

The display of tables should be done in a way that preserves the indentation of the left and right margins. If internal alignment is to be preserved when aligned columns are used, significant numbers of the characters in each column may be dropped, or some careful adjustment may be necessary when displaying the right margin. Horizontal and vertical rules surrounding the tables could be displayed using the limited graphics characters available.

The treatment of ligatures deserves consideration. One possibility might be to replace the ligatured characters with a number representing the number of characters included. Thus, 'ffl' would be replaced by '3', while 'fi' would be replaced by '2'. An alternative would be to replace the single ligature with the group of letters it represented. This would replace 'ffl' with '`ffl`'. This has the disadvantage of taking up more screen space, but the previewer will already be dropping characters from the line, and it would be easier for the user to interpret.

Other special characters in a font and all characters from a primarily symbolic font would be represented by the default error symbol on most terminals, the checkerboard box, or perhaps by the '?' character, as in `DVITYPE`.

These specifications are designed to be achievable on the lowest common denominator of terminal likely to be in use in the TEX community. It is probable that most TEX users have terminals with some special capabilities which could be used to enhance this type of previewer. For example, we are currently implementing an ASCII previewer on a Data General 461, which allows for down-loadable fonts. These special fonts may be used for some special characters such as ligatures or other symbols.

While a TEX previewer for an ASCII terminal has obvious limitations, we believe that the timeliness of information conveyed to the user will justify the effort expended. We expect to have the previewer mentioned above available for distribution in March. It will also be adapted to run on a DEC VT220.

```
English words like 'technology' stem from a Greek root beginning with the le
???...; and this same Greek word means art as well as technology. Hence the
name TEX, which is an uppercase form of ???.
     Insiders pronounce the ? of TEX as a Greek chi, not as an 'x', so that
TEX rhymes with the word blecchhh. It's the 'ch' sound in Scottish words lik
loch or German words like ach; it's a Spanish 'j' and a Russian 'kh'. When y
say it correctly to your computer, the terminal may become slightly moist.
     The purpose of this pronunciation exercise is to remind you that TEX
is primarily concerned with high-quality technical manuscripts: Its emphasis
on art and technology, as in the underlying Greek word. If you merely want
produce a passably good document--something acceptable and basically read-
able but not really beautiful--a simpler system will usually suffice. With T
the goal is to produce the finest quality; this requires more attention to d
but you will not find it much harder to go the extra distance, and you'll be
to take special pride in the finished product.
```

Figure 1: Display of Left Margin

```
h words like 'technology' stem from a Greek root beginning with the letters
???...; and this same Greek word means art as well as technology. Hence the
me TEX, which is an uppercase form of ???.
     Insiders pronounce the ? of TEX as a Greek chi, not as an 'x', so that
X rhymes with the word blecchhh. It's the 'ch' sound in Scottish words like
k or German words like ach; it's a Spanish 'j' and a Russian 'kh'. When you
 it correctly to your computer, the terminal may become slightly moist.
      The purpose of this pronunciation exercise is to remind you that TEX
rimarily concerned with high-quality technical manuscripts: Its emphasis is
 art and technology, as in the underlying Greek word. If you merely want to
 produce a passably good document--something acceptable and basically read-
e but not really beautiful--a simpler system will usually suffice. With TEX
l is to produce the finest quality; this requires more attention to detail,
u will not find it much harder to go the extra distance, and you'll be able
 special pride in the finished product.
```

Figure 2: Display of Right Margin

```
English words like 'technology' stem[.]reek root beginning with the letters
???...; and this same Greek word mea[.]art as well as technology. Hence the
name TEX, which is an uppercase form[.]???.
     Insiders pronounce the ? of TE[.] a Greek chi, not as an 'x', so that
TEX rhymes with the word blecchhh. I[.]he 'ch' sound in Scottish words like
loch or German words like ach; it's [.]ish 'j' and a Russian 'kh'. When you
say it correctly to your computer, t[.]minal may become slightly moist.
     The purpose of this pronunciat[.]n exercise is to remind you that TEX
is primarily concerned with high-qua[.]chnical manuscripts:~Its emphasis is
on art and technology, as in the und[.]ng Greek word.~If you merely want to
produce a passably good document--so[.]thing acceptable and basically read-
able but not really beautiful--a sim[.]ystem will usually suffice.~With TEX
the goal is to produce the finest qu[.]s requires more attention to detail,
but you will not find it much harder[.]e extra distance, and you'll be able
to take special pride in the finishe[.]
```

Figure 3: Display of Both Margins

A Screen Previewer for VM/CMS

Don Hosek

A good previewer is a useful tool for working with TEX, but unfortunately, there are very few available. For users of TEX under the IBM VM/CMS system, the only choice available used to be DVI82, a Versatec driver that, as an added feature, allowed previewing on IBM 3279 and 3179-G terminals.

To deal with this situation, I wrote DVIview, a TEX previewer that displays its output on VT640-compatible displays connected to an IBM mainframe via either a 3705 controller or a Series-1/7171 protocol converter. In addition, the output routines are modularized enough that it should be a fairly simple task to modify the program to drive any graphics terminal connected to the mainframe. (I have plans to include support for GDDM-driven displays in the near future.)

DVIview is a lengthy `WEB` program that interprets the instructions in a DVI file and displays them on the user's screen as determined by commands typed at the keyboard. The entire page may be viewed with block outlines of the characters, or smaller portions of the page may be selected and viewed using the actual shapes of the TEX fonts. Font information is read from PK files. (I cannot recommend the PK format enough to people writing new device drivers; the fonts take roughly half the space of GF files and about a third the space of PXL files. And PK readers are easier to write!)

The DVIview distribution includes two manuals: "Previewing TEX Output With DVIview" is the users' guide and explains how to use the program from a user's standpoint. Also included is "Installing and Customizing DVIview", intended for the systems person who installs DVIview. Instructions are given for installing DVIview as is, as well as instructions on adding changes to the file and a "Hitchhikers' Guide to `WEB`" (for those who don't care how they get where they're going as long as they don't have to ride the bus).

Due to the size of the program, it cannot be distributed over the networks. To obtain a copy of DVIview and its documentation, send $30 (to defray duplication costs), a blank tape, and a return mailer to:

Don Hosek
Platt Campus Center
Harvey Mudd College
Claremont, CA 91711

This article first appeared in TEXmag, 1987#7.

The program is public domain, so feel free to give it away. However, since it is still a young program, I'd like to keep track of who has copies for purposes of distributing updates.

Why TEX Should NOT Output PostScript — Yet

Shane Dunne
University of Western Ontario

In a recent TUGboat issue [1], Leslie Lamport suggested that since PostScript is becoming accepted as a standard page description language, perhaps TEX could be modified to output PostScript instead of DVI code. This is a good idea, but it should *not* be done yet for the following reason: At the moment, the available PostScript literature does not state precisely how drawn objects are to be rendered on the output raster. As I will show in this article, such a specification of PostScript's semantics is urgently needed to allow precision application programs such as TEX to properly use the language. I have written to PostScript's developers, Adobe Systems Inc. of Palo Alto, California, to draw their attention to this problem, and suggested that it be resolved publicly using TUGboat as a forum for discussion.

For readers unfamiliar with the PostScript language, a few words of explanation are in order. PostScript is a language designed specifically for specifying the output of raster printing devices. The language is interpreted, with the interpreter usually resident in the printer itself. It was intended to be human-readable, and hence uses only printable ASCII characters, but to simplify parsing it uses a rather cryptic postfix syntax. This is justified on the grounds that most PostScript programs will be written automatically as the output of other applications. PostScript incorporates a sophisticated device-independent drawing model in which a single transformation matrix (called the *current transformation matrix* or CTM) specifies the correspondence between the user and device coordinate systems. User coordinates are floating-point numbers with essentially infinite resolution; device coordinates are normally integers.

The incompleteness of the current PostScript semantic definition is apparent from the following example. Assume that the CTM of a PostScript device is set so that one unit in user space corresponds

to the distance between adjacent device pixels, and the point with coordinates (100, 100) is well within the visible part of the output page. (This is what our TEX driver does.) Now suppose the following code fragment is executed.

```
newpath
100 100 moveto
1 0 rlineto
0 1 rlineto
-1 0 rlineto
closepath fill
```

This draws an outline "path" which is a unit square with lower left-hand corner at (100, 100), and then fills it with black. It is reasonable to expect that a single device pixel will be blackened — after all, that *is* a black box one unit high by one wide, with the units we have chosen. However on our QMS PS800 laser printer, the result is a two-pixel by two-pixel box — four pixels are blackened. It turns out that whenever you ask for a box which is x units wide and y units high, you get one which is $x + 1$ pixels by $y + 1$ pixels. Similar things occur with the *stroke* command which draws lines — if you ask for a line width of one unit you get lines two pixels wide, two units becomes three pixels, and so on.

The practical upshot of this is that our DVI-to-PostScript driver, which outputs code according to what the PostScript reference manual says, always yields TEX rules which are one pixel too long and one pixel too high.

Attempting to second-guess the programmers of the PS800 PostScript implementation, I came up with the following scenario. We begin with the outline path with four vertices (100, 100), (101, 100), (101, 101), and (100, 101). Since we are working in one-to-one scale, multiplying these coordinate pairs by the CTM may add some translation factors, but should not make any multiplicative change to the values. The coordinates, which are real numbers, must next be converted to integers for the hardware, but they are already integers, and I have verified that the translation factors are also. Thus we can suppose without loss of generality that the coordinates are unchanged by the CTM. Now comes the strange part. The implementation seems to interpret integer-valued coordinate positions as *pixels*, and thus says that it must blacken all four different pixels identified by the four vertices (and, in this case, nothing else).

So apparently, each distinct coordinate *position* (a mathematical point in the plane with zero height and width) has been identified with a device *pixel* (something with very real height and width). Of course, if I am drawing a box 1 inch by 1 inch at 300 dots per inch, the error is only 1 part in 300, but if I am drawing *small* things (small with respect to the device resolution), the error can be quite serious, as shown by the above example. Unfortunately, typesetting and related applications involve *small* objects almost exclusively.

It would be more consistent with the PostScript philosophy to identify integer-valued coordinates (at 1 : 1 scale) with the lower left-hand corner of a pixel. This would require a small refinement to PostScript's fill algorithms.

The PostScript Language Reference Manual [2] says nothing definitive about the correspondence between coordinate positions and device pixels. It defines a virtual graphics machine separated from the real device by various mechanisms (such as the CTM) whose exact operation it does not define. Now in my experience, anything not defined in a software specification is usually defined by its implementation, which in turn means that I can expect different results from different printers, even at the same dot resolution.

It is of course tempting to say "Why worry about such details? If you want higher precision just go to a device with more dots per inch." There are two answers to this. The first is that 300 dpi laser printers, and lower-resolution mechanical dot-matrix printers, are probably going to be around for some time, and people will always want to use them at least for previewing. The second answer is that there really is no substitute for doing things right in the first place. If the sizes of drawn objects can be predicted with to-the-pixel accuracy, you can get the most out of whatever printer you have paid for. If not, you will always have to settle for less than what you know the machine can do.

As a developer of precision applications like TEX drivers, I need a formal definition of how PostScript's drawing operators (primarily `fill` and `stroke`) should be rendered on raster devices, relating the high-level virtual machine defined by the language to the low-level hardware. Such a definition could itself be device-independent, speaking in terms of a target device with x dots per inch resolution horizontally and y dots per inch vertically. It could take the form of a published article, perhaps here in the TUGboat.

Aside from the fill-outline problem I have already mentioned, at least two other aspects would have to be addressed (and now I must apologize for using terms which will be unfamiliar to some readers). First, are CTM-transformed coordinates rounded or truncated in order to be converted to

integers for the hardware? (I recommend truncation since it is fast, and the user may change it to rounding by adding .5 to the translation components in the CTM.) Second, what is the precise orientation of bitmap characters generated by `imagemask`, with respect to the *current point*. I suggest that the current point should coincide with the extreme lower left-hand corner of the rendered image. That is, when the CTM is as described in the earlier example, the current point should identify the lower left-hand corner of a pixel, and this pixel should be overlaid with the lower leftmost pixel in the bitmap image. (A related issue is that when the coordinate system is inverted vertically, the current point should coincide with the extreme upper left-hand corner of the image — this does not appear to happen with our printer.)

A description of how the PostScript software is structured, distributed, and implemented on specific devices would also help applications developers to understand its operation. My guess is that the basic PostScript interpreter is provided by Adobe Systems, and each device manufacturer writes their own driver, but this is only a guess. Perhaps device manufacturers tell Adobe how their machine works, and later receive a fully-configured interpreter in machine-code form. Just how much does the device manufacturer do, and by implication, how much can be expected from a given PostScript-compatible product?

The issues raised in this article came up in the course of research into musical score setting using TEX. I have been working with a modified DVI-to-PostScript driver which allows inclusion of arbitrary PostScript code into TEX source material using the `\special` primitive. The idea is to use the power of PostScript to draw all the variable elements of musical material (e.g. note stems of variable length but fixed width). The lack of a definitive explanation of how PostScript's graphic primitives work at the device level forced me to spend a great deal of time writing tiny PostScript programs and examining the results — with a microscope! — to figure out what the printer was doing. I needed the microscope only to measure the *extent* of various inaccuracies in size and position — at 300 dpi the *existence* of these inaccuracies is immediately obvious to the naked eye.

References:

[1] Lamport, L. *TEX Output for the Future*. TUGboat **8,1** (April 1987).

[2] Adobe Systems Inc. *PostScript Language Reference Manual*. Addison-Wesley, Reading, Massachusetts. 1985.

Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue has been prepared at the American Mathematical Society, on a VAX 8600 (VMS) and output on an APS-μ5 using resident CM fonts and additional downloadable fonts for special purposes.

- Apple LaserWriter (300dpi):
  ArborText advertisement, p. 110.
- TEXnology, Inc., advertisement, p. 103.
- Canon CX (300 dpi): Georgia Tobin, *The ABC's of special effects*, p. 15.
- Compugraphic 8600 (1301.5 dpi):
  TEXtl advertisement, p. 106.
- HP LaserJet (300dpi):
  Personal TEX advertisement, p. 99.
- Linotronic 100 (1270 dpi):
  Design Science advertisement, p. 105.
- Kellerman and Smith advertisement, cover 3.
- Micro Publishing advertisement, p. 101.
- Xerox 4500 (300 dpi): Greek sample text, in Silvio Levy, *Using Greek fonts with TEX*, p. 22, as indicated.

Site Reports

The Commodore Amiga: A Magic TEX Machine

Tomas Rokicki

The Commodore Amiga makes an impressive TEX machine, able to compete with computers costing several times its price. In this report on the status of AmigaTEX, features will be discussed that might well be profitably implemented on other machines. Let me tantalize you, first, by mentioning that it is possible with this package to go from your document in your favorite editor to the first page of your TEX'ed document in a preview window in about a second of real time and with a single keystroke or menu selection.

But before I discuss some of the more esoteric features, let me tell you what the basic package contains and requires. AmigaTEX comes on eight floppies including TEX 2.9, iniTEX, LATEX, BibTEX, preview, over 1200 previewer fonts, and some font conversion utilities. Printer drivers are available separately for the HP LaserJet Plus, standard PostScript printers, the QMS Kiss and SmartWriter, the NEC P6/P7 series, the Epson LQ series and MX series, and some other less popular dot-matrix printers. Plain TEX will run on a 512K machine; LATEX requires a megabyte of memory. A hard disk is not necessary; an extra two megabytes of memory is cheaper and much more useful. Two floppy drives are highly recommended. Three megabytes of memory gives you the best environment.

The first question that pops up is, how can a hard disk not be necessary if the package requires eight floppies to distribute? Is it really possible to put an entire TEX environment, including TEX, the preview program, a printer driver, the editor, previewer and laser printer fonts, and all of the system software onto two floppies and still have room for TEX source files? Indeed it is. The TEX software less the previewer and laser printer fonts requires less than 400K, including even the plain format file and an editor. The key feature that makes a floppy TEX environment practical is font caching.

Font caching is based on the idea that, of the thousands (literally) of fonts supplied with the previewer and a printer driver, the typical user will need only a few dozens, or maybe a hundred. The idea is to find which fonts the user needs. These fonts should be made easily available. This is easily accomplished. A directory is assigned to hold the commonly used fonts; this directory resides on a single disk that is always in one of the disk drives when the user is using TEX. This directory is initially empty. As the previewer or a printer driver requires a font and cannot find it in this font directory, it queries the user to insert the appropriate distribution disk on which the floppy was supplied. The driver program then copies it into this cache directory, so the next time it is needed, it is there.

This system works quite well in actual experience. The first few times a printer driver is run, the user has to swap some floppies. But after the fonts are in the cache directory, things go smoothly and quickly. And two floppies are now sufficient for a nice working environment.

But floppies are slow. Even fast floppies are slow. The data rate of standard double density 90 mm floppies is 250,000 data bits per second, or about 31,000 bytes per second maximum. Actual transfer rate is typically around 18,000 bytes per second on good days. Thus, to load the TEX executable image of 128,000 bytes takes seven seconds. This is a long enough delay to get annoying after a period of use.

On a single-tasking computer, there would not be much one could do about this. The Amiga comes with a true multitasking operating system, however, so it is a simple matter to run TEX in a loop mode, where after finishing one document, it hangs around and waits for the next rather than exiting.

The plain format file is almost as large as the TEX executable, so the delay in loading it is as long as the delay in loading TEX itself. But, since TEX is hanging around in memory, a copy of the format file might as well stay in memory too. On the Amiga, as TEX loads the format file the first time, it copies the portions of the data structures that will be destroyed as a document is processed into another area of memory from which it can be quickly restored as soon as the document is done. Since some portions of the format file contain data that does not change as a document is processed, such as the string pool, these areas need not be saved or reloaded after the first time.

Now, actually, it has not been mentioned that on a single-tasking computer, both TEX and the format file can be loaded into a RAM disk, from which loading is quick. Nonetheless, the above tricks require less memory and yield faster operation than a RAM disk alone would provide. In addition, while I've expounded these ideas in the light of floppy drive storage, they are also useful when all of the

files reside on a hard disk, although their impact is not as great. The difference is that using the above ideas, TEX is ready and preloaded at the instant you decide to use it.

Still, TEX spends most of its time actually processing documents. Even when run off floppies, the fourteen second load time is dwarfed by processing times of several minutes for sixty page documents. TEX is doing a lot of work, so it is doubtful that this processing time can be cut significantly using the current hardware. A 68020 board can always be plugged in, but the TEXbook will still take a good quarter hour or more. So what facilities can be provided that will allow the user to make the best possible use of the time during which TEX is working?

With a multitasking environment, the user can read net news. But more often than not, the user is wondering what the document looks like. With the Amiga, TEX can send each page as it is finished to the previewer. (Of course, the previewer stays around waiting for new jobs just like TEX does.) This way, as TEX is working on page twenty of that forty page report, the user can preview any page up to page twenty, and make changes in his editor buffer as he finds things to change.

It is true that the message-passing executive of the Amiga makes such communication easy to implement; TEX simply sends messages containing the page data as it completes each page, and the previewer actually writes and re-reads the DVI file as necessary. It should be possible to do a similar trick under, for instance, the Unix operating system using sockets.

So, at the moment, Joe User saves his file from the editor, clicks on the TEX window with his mouse and hits carriage return (it's easy to make TEX remember the last file name processed) and, just as soon as that first page is done, it pops up in a previewer window. But, if TEX encounters an error, Joe must find the place where the error happened in his file. In addition, he has no easy way to process just a small portion of his document, say the equation on page twenty-four, to see how his changes look.

At this point, it is not difficult to take care of the problem for Joe. All we need to do is add some small changes to his editor so that it understands a function key or two, and sends the proper messages to TEX. For instance, function key nine might be programmed to take the current cut region, append it to just the top of his file containing macros, saving this file and telling TEX to start processing. Joe uses EMACS, so such hacks are easily made. But Bob uses TxED (by Charlie Heath) and Paul likes vi, and Peter has his own homebrew editor. Source isn't even available for most of these editors, so how do you add such facilities?

Out of the sky appears William Hawes, author of ARexx, an implementation of the REXX language so revered on IBM mainframes. On the Amiga, ARexx is so much more than a script language. It is a general interprocess communication manager, programmable in an interpretive language so simple that anyone can use it. To make any program talk to any other program on the Amiga, all the developer must do is make it talk to ARexx. Then, a set of four-line ARexx macros can be written by either the user or the developer to transfer data back and forth between the applications.

As of this writing, only two editors exist with ARexx ports, so only these two can be used to make an integrated environment with AmigaTEX. But as more and more programs appear with ARexx ports, the versatility of all of them will increase dramatically. For instance, it is now possible to integrate an editor, TEX, the previewer, a terminal communications program and any number of other programs in a single, unified working environment that is remarkable to behold. A similiar situation will soon exist with OS/2 on the IBM PC's.

But enough soapboxing. You have to see it for yourself. If you send me, Tomas Rokicki, a letter, to Box 2081, Stanford, CA 94309, I will send you an Amiga diskette containing a demonstration version of the AmigaTEX package, and pricing and ordering information. For information on ARexx, write Bill Hawes at Box 308, Maynard, MA 01754.

DG Site Report

Bart Childs
Texas A&M University

We are now delivering TEX 2.9. It went in as easy as the previous releases. We are also delivering METAFONT 1.3. (We had been delivering out-of-date 1.0 versions for more than a year.) We now have a stochastic ArpaNet connection and should find it easy to stay current.

IBM VM/CMS Site Report

Dean Guenther
Washington State University

In February I sent the TEX 2.9 VM/CMS distribution tape to Maria Code. This tape includes a few new programs, upgrades, fixes, etc.

SLITEX has been added to the distribution. It has been requested for some time, but has been difficult to get ahold of. Thanks to Barbara Beeton, who sent me a copy of the fonts and macros. The other newcomer to the distribution is DVI3279, a screen preview program for the IBM 3279 or 3179g terminals. It works well on DVI files and METAFONT output run through GFTODVI. Thanks to Georg Bayer at the Technische Universitaet Braunschweig in Germany for this contribution. It is written in WEB, and requires GDDM and FORTRAN. The messages from this program are all in German, so höffentlich ist ihr deutsch gut.

Upgrades on the distribution tape along with TEX 2.9 include LATEX, thanks again to Barbara. I upgraded DVITYPE to version 2.9, and METAFONT and all of its MF files to version 1.3. GFTOPXL was also modified to output 1K blocks.

Bob Creasy made several upgrades and fixes to the IBM printer support. We worked out a scheme for DVI2LIST and PXLCVT to support larger magnifications, a requirement necessary to support SLITEX. Bob also modified DVI2LIST's absolute horizontal alignment to match DVITYPE. This fixed a problem reported by several, who were unable to get the following to line up:

```
{\obeylines\obeyspaces\tt
I         I         I         I
01234567890123456789012345678901234567890
I         I         I         I
}
```

This now works. Bob also made upgrades to his PFS font building EXECs, including allowing larger magnifications, and support for generating the PXL fonts used by DVI3279.

Fujitsu Announces TEX Port

Fujitsu Limited, a member of the TEX Users Group of Japan, announced that it has completed the porting of TEX V1.0 into their M-series, large-scale mainframe computer. This product, FACOM OSIV TEX V01L10, is now available to users. The command procedure provides support for various printing devices and formats; it allows users to specify execution units (INITEX, DVIwrite, printing), to set printing devices (including display) and printing sizes, and to establish execution environment (batch or TSS). For more information:

Shozo Taguchi
Deputy General Manager
Software Division
Fujitsu Limited
140 Miyamoto, Numazu-shi
Shizuoka-ken 410-03, JAPAN

UnixTEX Site Report

Pierre A. MacKay

The January 1988 upgrade of the UnixTEX distribution is the most important and far-reaching in several years. The changes in TEX itself since the last site report are relatively trivial, and correct bugs that only very advanced users of the program would ever be likely to run into, but almost every other part of the distribution has undergone major changes. TEX has now reached Version 2.9, and the source file has been slightly reformatted with ASCII form-feeds immediately preceding each starred module. This has no particular effect on UnixTEX, except in the change files, where the last starred module before the index:

```
^L@* \[54] System-dependent changes.
```

(the module for system-dependent code) is usually referenced, and will cause an error if the form-feed is not added to the change file.

The most important news associated with the compilation of TEX however, is the successful completion of TEX-to-C, an interpreter which takes the Pascal output from tangle, and converts it into clean C source code, lintable, and so far as we know, completely consistent with the ANSI draft C standard. This interpreter is the result of several years of work by Tom Rokicki and Tim Morgan. Rokicki did the ground work a few years ago, and

Tim Morgan has refined and restructured the whole system so that it now promises to make it possible to eliminate the proliferation of system-specific change files. The code generated is smaller and substantially faster in execution than even the best Pascal compilations, and it makes the generation of a truly immense TEX possible, since C is not subject to the Berkeley Pascal compiler restriction on array sizes. (Berkeley pc restricts array indices to a 16-bit range.) The code can be compiled in two modes, one with standard variables, which seems to work on just about every system tried so far, and one with register variables, which works if your compiler is clever enough, and produces executables that are about 10% faster than the non-register versions. (I note, with some amusement, that the 68010 Sun-2 compiles correctly with register variables under OS 3.2, but the 68020 Sun-3 does not.)

METAFONT is still at version 1.3. There have been some slight changes in the Computer Modern mf files over the past year, and it is probably advisable to compile all fonts again. There are small problems with low resolutions still in CMMI6, CMTT8 and CMTEX8, which produce strange paths when compiled at 118 dpi resolution for the BitGraph and Sun screen previewers. Readers of TEXhax will have followed the recent discussion of mode_specials added to compiled fonts in GF format. The purpose of these is to provide, in the font itself, a record of the settings used when the font was made. A typical group of mode_specials as printed out by the UNIX strings utility is:

```
cmr10
mag:=2.0736;
mode:=RicohFortyEighty;
pixels_per_inch:=300;
blacker:=0.2;
fillin:=-0.2;
o_correction:=0.5;
```

The macros to produce this information are part of the file U_Wash.mf in the directory ./utility-fonts/bases on the UnixTEX distribution tape.

Because we run both write-white and write-black printers at the University of Washington, we have had to think more clearly about the adjustment for write-white devices (discussed by Neenie Billawala in TUGboat Vol. 8, No. 1, pages 29–32). Both the changes to cmbase.mf described there occur in the font_setup macro, and there is really no need to have an entire write-white cmbase_w.mf and a separate preloaded version of METAFONT as we have had for the past year. The ./cmfonts/mf directory now contains the short white_setup.mf file, which is used in place of the write-black font_setup when appropriate through the addition of the line let font_setup=white_setup to all write-white mode_defs. Add the line input white_setup to your local mode_def file. The fonts currently on the tape have all been recompiled with the macros described above.

There has been only one significant change in TEXware since the last distribution. The dvitype program has been changed to reflect a change made in TEX itself. Make sure you have the latest version (2.9) before you try to trip, or you will get some misleading discrepancies in the dvitype output file.

A sub-directory of ./tex82/TeXware contains the CWEB programs, which extend the whole idea of integrated documentation to the C language (which desperately needs it). These programs are not to be confused with TEX-to-C, which starts from the original Pascal-based WEB files. CWEB is the work of Silvio Levy of Princeton.

A few minor bug-fixes have improved the suite of PK utilities in ./mf84/MFware. These, and some C versions of the same programs were provided by Tom Rokicki. A new directory, ./mf84/MFcontrib, includes some PostScript utilities and PXtoGF, which was brought into Unix compatibility by Karl Berry. It was the availability of this program which made it possible to convert the AMS Cyrillic and special symbol fonts into GF format. Tom Rokicki is presently putting the finishing touches on the WEB source for PKtoGF, which will make it possible to send out more fonts in less space without excluding the large number of output drivers which use GF format.

The most important unfinished business in all the above is the extension of the TEX-to-C approach to METAFONT, TEXware and MFware. If all the standard WEB programs could be interpreted into ANSI draft standard C code, as has been done with TEX, it would be possible to eliminate the proliferation of system-dependent change files from the UNIX distribution and to target the large and, so far, inadequately served System V UNIX community. Above all, we need a bootstrap tangle.c.

Up till now, the distribution tape has offered a small range of precompiled binaries of TEX and METAFONT. This makes less and less sense when even VAX no longer means a single architecture and the binaries will soon be dropped. It might be desirable, however, to send out a variety of precompiled tangle executables. I can offer tangle precompiled on a VAX11-750, a VAX8550, a Sun2

(MC68010) and a Sun3 (MC68020). If people will send me precompiled `tangle` executables for other architectures in `btoa` format, I will put them on the distribution as I receive them.

Since almost all DVI drivers now use either GF or PK format fonts, the `PXL` directory has been removed from the distribution. In compensation, a larger range of precompiled GF format files is being sent out, but even these can target only a small number of devices. Lowres and CanonLBP fonts (200 and 240 dpi) are still well represented, but I wonder how many 240 dpi devices are still in use? At 300 dpi, both write-white and write-black fonts are provided (the write-white is tuned for the Ricoh 4080 print engine, and the write-black for the ubiquitous CanonCX). If you have any 300 dpi device at all, one or other of these compilations will serve as a temporary resource, but you will probably want to recompile to get the best out of a print engine that is neither of the above. That unhappy compromise, the 118 dpi font, is also still with us. The AMS Cyrillic and special symbol fonts exist only in write-black versions because they were compiled long ago on a Tops20 machine in old METAFONT-in-SAIL, and I cannot recompile them.

Several of the output drivers have been revised by various contributors to the distribution. Scott Simpson of TRW has completely rewritten the driver for QMS/Talaris, which is now known as `quicspool`. The basic `README` file for this is worth reading even if you don't run a QMS machine. The entire `ctex` system has been reworked, and enhanced by the addition of System V code and routines for a previewer on the ATT 5620 supplied by Lou Salkind of NYU. There is also a new pair of previewers, `texsun` and `texx`, running under SunViews and X, respectively, which was contributed by Dirk Grunwald of the University of Illinois. The backbone of the system is the library developed by Chris Torek at the University of Maryland. The `ctex` library has already spawned a larger number of derived systems than any other, and it seems appropriate to suggest here that drivers written in C might profitably be adjusted to make use of as much common code from `ctex` as possible. GF and PK interpreter modules are beginning to proliferate, and they all do essentially the same thing. If there is some strong argument of increased efficiency in one of the interpreters outside `ctex` then surely the techniques could be incorporated into the `ctex` library.

The LN03 was provided with a new driver by Matt Thomas in September, but his shipment of code got lost in a mail crunch. My apologies for not unwrapping it earlier. In any case it is on the distribution now.

There is no major new development in `dvi2ps` ready for release as yet. I have increased the array sizes to reflect the fact that GF format permits 256 character fonts, and have changed the meaning of the `-d` flag so that it can be used to change pixel density. The default remains 300 dpi, but a flag value of `-d 600` is available for devices such as the Varityper VT600. The header file `tex.ps` needs to be completely rewritten to eliminate the dozen or so 300 dpi-isms, and allow for some sort of conditional coding. For now, a 600 dpi version has been added to the distribution under the name `tex6.ps`. This automatically replaces the default `tex.ps` when the 600 dpi flag is used.

The foreign-language `./babel` directory is at last beginning to grow. In addition to Portuguese there is now a German section, with a German hyphenation file, a Swedish section with a complete package of macros for Swedish-Language LaTeX, and a Semitics section with a first pass at `TeX-XeT`. In the near future, I hope to add a machine-independent change file for use with TeX-to-C, which will make `TeX-XeT` much easier to compile.

TeX to C Converter

Tim Morgan
University of California, Irvine

Tomas Rokicki and I have developed a set of programs, makefiles, shell scripts, and a changefile which can automatically convert `tex.web` into a C program. We are very glad to say that the entire conversion package has been placed in the public domain, and it is being distributed by the UNIXTeX site coordinator, Pierre MacKay, at the University of Washington. It's also available for anonymous ftp from Internet host "`ics.uci.edu`".

There are still a number of developments in progress as of this writing. Several people are working on adapting the conversion software to work for METAFONT, and this effort is nearing completion. We are currently looking for someone to convert the TeXware and METAFONTware programs to C as well, preferably via an automated process. Once *tangle*, in particular, is available in C, a site with only a C compiler will be able to bring TeX

and METAFONT up starting with only sources on the UNIXTEX distribution tape.

The C source which is generated is fairly efficient and portable. It passes through *lint* with no unexpected complaints, and I've tried the code on a number of different UNIX systems. In all cases, and with versions of TEX from 2.2 to 2.9, the resulting executables easily pass the trip test on the 4.2BSD-based machines available to me for testing. The environments in which I've tested the code include a VAX-11/750 running 4.2BSD, SunOS 3.2 on a Sun-3, Sequent Dynix 2.1.1, and Integrated Solutions UNIX version 3.07. The code turned up minor bugs in the VAX, Sun, and Sequent compilers, but workarounds were found in all cases. I was also able to compile the C sources and pass the trip test using the System V compilers supplied by Sun and Sequent. As long as there is sufficient memory available, I believe that this code can be easily ported to any C environment. It has been in production use at the University of California, Irvine, for over six months. Although improvements in the conversion process are still being made, the code which is generated remains almost the same, so I consider it to be highly stable.

Obviously, the most important thing is that TEX is converted automatically into C. This feature makes it easy to track new versions of TEX as they become available. A changefile is used to handle many of the necessary changes, and therefore a working *tangle* is required. The output from *tangle* is mainly PASCAL, although with a C flavor. It is then converted into C and split into multiple modules by a pipeline of other programs. The entire process is automated, so all the user need do is type "`make`".

VAX/VMS Site Report

David Kellerman
Kellerman & Smith

There is a new version of the VAX/VMS distribution available from Kellerman & Smith. TEX is at version 2.9 and changing quickly, METAFONT is at version 1.3. Most of the other software, most of the macro packages, the font sets, and the packaging of the release have also changed since our last release.

The repackaging is perhaps the biggest change to the new distribution. Over the years, our conversations with new users of TEX have made it clear that this enormous and complex package has all the intrigue of a giant puzzle for many users, but has, well, the frustration of a giant puzzle for many others. To help make TEX more accessible to the increasing number of users who hold the latter view, we have rearranged it into pieces that can be installed selectively with the standard VMSINSTAL mechanism. This should be of particular benefit to users who want to get TEX or LATEX going quickly, who need to fit a system into very little space, or who are working in a cluster environment. (You old pros can still install it by hand if you want to.)

The font sets, based on new Computer Modern METAFONT sources, have been expanded to mag-steps 0–8 for both the XEROX XP-12 (QMS) and the Canon CX (IMAGEN). They are distributed in PK format and come with a new utility program, XXtoXX, that converts between GF, PK, and PXL formats (all combinations) and converts from one RMS record format to another. The XXtoXX utility is fast, it can convert multiple files in one execution, and we have tested it more thoroughly than some of our earlier changes to font conversion utilities.

The LATEX and SLITEX macros are updated. Also, we now provide all the fonts they require, and the XXtoXX utility can produce SLITEX "invisible" fonts to match any normal font.

The contributed software on the release includes the LN03 driver from Flavio Rose, the PostScript driver and screen previewers from Andrew Trevorrow, and the MWEB software (Modula-2 WEB) from Wayne Sewell.

The software is now available on either 2400′ magtape at 1600 bpi or TK50 cartridge, and costs $200.00 (U.S.) including shipping within the U.S. and Canada. Add $50.00 (U.S.) for air parcel post shipment to other countries.

Typesetting on Personal Computers

Writer's Tools I: PC Spelling and Grammar Checkers

Alan Hoenig and Mitchell Pfeffer

In my pre-TEX days, I was "heavy" into spelling and grammar checkers. They were such convenient shortcuts that it was like getting something for nothing. These programs were tailored to suit my word processor, and I had to give them up when I replaced that word processor with TEX. None of them had any problem accepting my TEX source files — ASCII was usually easier for them than the special-format word processing files — but they usually stumbled over TEX's embedded commands. Since then, these programs have become much more robust. It's appropriate to take a look at them to see how helpful they might be today.

Spelling Checkers

Spelling checkers offer the fewest problems for use with TEX, and almost any decent one is worth your while. It's notoriously difficult to proofread your own work anywhere near accurately. All spell checkers let you build up auxiliary dictionaries for the TEX commands in your documents. I looked for speed and the absence of idiosyncratic behavior in the presence of TEX. One particularly annoying problem is the inability of some of them to use digits to delimit a word. Thus, "vskip1pt", "vskip2p", "vskip3p", and so on are all distinct words to these spell checkers. You can't, therefore, add "vskip" to an auxiliary dictionary. (This problem won't apply to all the Goody-Two-Shoes who conscientiously add proper spacing to separate their commands from their qualifiers.)

I tried several programs for this column, and two are worth commenting on. *Webster's New World Spelling Checker* was fast, but it suffered from the embedded digit syndrome I mentioned above. Its dictionary contains 110,000 words, and it will catch repeated words. Apart from the embedded digit problem, it was also difficult to run *NWS* across subdirectory boundaries. That is, if a file is in directory `\articles` and the software is in `\spell`, I could not invoke the program from the `\articles` subdirectory. This is a bit of a bother. The brightly-bound user manual is quite adequate; you'll have no problem getting this program up and running. *(Webster's New World Spelling Checker, Simon & Schuster Software, 1230 Avenue of the Americas, New York City 10020; $59.95)*

Altogether more satisfactory is the top-rated *MicroSpell* from Trigram Systems, even though its dictionary contains a 'mere' 80,000 words. It works like lightening, and I can run it from my document subdirectory. I can configure it to treat digits as word boundaries, so I can add "vskip" and so on to my auxiliary dictionary. It also will catch repeated words and do simple capitalization and punctuation checking. It's particularly good at guessing at what I really meant in instances when I really mangle a spelling. You'll have no problem with the manual, or with installing and using it. A winner. *(MicroSpell, Trigram Systems, 5840 Northumberland Street, Pittsburgh, PA 15217; (412) 422-8976); $69.95)*

I was unable to locate any spelling checker that treats the backslash as a letter. Therefore, while all could catch my "hobx-es" and my "medksips", none could tell me when I forgot the leading backslash.

Grammar Checkers.

Grammar checkers are more problematical. Are they ever any good? It's doubtful they can make you a great writer, or even a good writer, but their suggestions will prevent you from committing most major solecisms. They are particularly good at catching the passive voice syndrome ("it is to be hoped...") which infects so much technical writing. Since TEX's commands fit into no syntax scheme, grammar checkers usually stumble badly over TEX source. The problem is acute in those programs which provide on-line analyses of your writing. You can't tell them to bypass command sequences, about which they persist in complaining. It takes forever to analyze a file in these circumstances.

If used conscientiously, grammar checkers put themselves out of business. After awhile, one becomes adept at avoiding the errors grammar checkers pick up.

The best for use with TEX is *RightWriter*. It creates a separate file for its analysis. Although this contains a record of every unmatched delimiter (it won't look to the next line) and of every halign table (which is a long, complex sentence to it), at least you can zip past these messages and concentrate on matters of substance, like the passive voice, padded and redundant phrases, and archaic or technical usage. As a bonus, the program "grades" you by assigning numerical values to the readability, strength, degree of description, and

usage of jargon of your document. Yuppie writers will enjoy competing against themselves! Finally, when you've finished ignoring or implementing its comments, you run your file through the program to strip out its comments. *RightWriter* is easy to set up and use. *(RightWriter; RightSoft, Inc., 2033 Wood Street, Ste. 218, Sarasota, FL 33577, (813) 952–9211; $95)*

In non-typesetting situations, reviewers award first place to *Grammatik II*, and it's easy to see why. Its developers gave it slick packaging and included well-written manuals. Not only does it do its job well, but it is almost endlessly configurable. You can tell it which error types to ignore, and you can revise its rule dictionaries so *Grammatik* will search documents for your own idiosyncratic mistakes. You can instruct it to run silently and include its error reports within your file, but it lacks one essential capability for TEX users: the ability to strip these messages out when you're finished with them. (It's easy enough to create a Pascal filter to excise them because these inclusions all have a standard format. You may decide *Grammatik*'s extra functionality is worth this extra effort.) One of Reference Software's personnel told me that they have scheduled the release of a major upgrade, *Grammatik III*, for mid-spring. *(Grammatik II; Reference Software, 330 Townsend St., Ste. 131, San Francisco, CA 94107 (800) 872–9933; $89.00)*

A program called *Readability* is not quite a grammar checker, but not quite *not* one either. Originally developed in *Sweden* (of all places), it performs extensive statistical analyses on your prose, informing you (in sophisticated displays) how many long sentences you have, how many long words you use, how many *runs* of long words you have, and so forth. The program generates some overall comments on your document, and allows you to redo these analyses assuming your document fits into several different categories, such as novel, juvenile book, newspaper article, bureaucratic gobbledygook, and others. *Readability* won't catch the passive voice or repeated words, which is why it isn't a grammar checker. This program displays its non-English origins in some amusing typographical errors in its (otherwise excellent) manuals and screen displays.

My impression is that the quality of one's writing will soar should an author conscientiously use the results of *Readability* to re-work the document, particularly if you use it in conjunction with a program like *RightWriter*. The key word in that sentence is *conscientiously*. A lot of work is involved, and most of the people I know who put pen to paper (or the electronic equivalent) are wanting in that trait. *(Readability; Scandinavian PC Systems, 3 Brookside Park, Old Greenwich, CT 06870, (203) 698–0823; $59.95)*

In a forthcoming *TUGboat*, we'll cover additional writer's tools.

Grapevine Reports of Inexpensive Versions of TEX

Alan Hoenig and Mitch Pfeffer

The TEX community may be interested in two product announcements we received à propos of cheap versions of TEX available for PC-based systems. We have not yet verified any of the claims of these announcements, but we hope to report on them in greater detail in an upcoming issue.

Gary Beihl (`cad.beihl@MCC.COM`; postal address is MCC, 3500 West Balcones Center Cr., Austin, TX 78759; 512-353–0978) has recently completed a port of TEX2.7 to MSDOS using Datalight's C compiler. 'DosTEX' has passed the `TRIP` test and may be freely distributed provided that all copies are complete and unmodified, and that no fee may be charged for redistribution. Your system will need 640K memory and 4.5MB hard disk space. A modified version of Nelson Beebe's Epson `.dvi` driver is included. Output is 420h×216v dpi. DosTEX compiled the *TEXbook* at about 15 sec/page on a 6Mhz, 1 wait state, AT clone. The package contains a fairly complete `texinputs` directory, and a reasonable subset of the grand ensemble of CM fonts. DosTEX comes to you complete on seven 360K floppy diskettes plus installation instructions minus any warranty or support, although Gary will attempt to fix bugs in a timely fashion. DosTEX is priced at US$75 (US$85 foreign), and a check or money order payable to *Electronetics* should be mailed to *Electronetics, Inc., c/o Gary Beihl, 119 Jackrabbit Run, Round Rock, TX 78664.*

Richard Kinch has also ported TEX to the C language, and dubbed the result 'Turbo-TEX.' Pricing for the Turbo-TEX software and documentation range from $99 for the IBM-PC compatible version to $2000 for a VAX version. The price for a source-code kit for the IBM PC, including the Turbo-TEX C source code and the Microsoft C compiler 5.0, is $650. Complimentary demonstration copies of Turbo-TEX for PCompatibles, AT&T Unix

complex expressions) when encountering limits of a given C compiler.

Since we wrote the translator itself in C, the only compiler needed to build TurboTEX from `tex.web` on a new, stand-alone machine is a C compiler. (We also apply the translator to `TANGLE`.) We maintain portability for all the translation tools, and not just the TEX program.

A further goal was to have a single C source for all target machines and operating systems. C is suitably adept at conditional compilation with its standard preprocessor, and we used that language feature to keep a unified C source for both Unix System V and MS-DOS. We used only features common to the various C language standards to minimize the amount of conditional code. We also combed out all operating system dependencies into small, hand-written, separately-compiled source files. We maximized the portability of the source code by carefully designing the translator to produce only portable C constructs.

Finally, our watchwords were "correctness" and "certifiability". We designed a product that we could finish on a limited schedule and budget, but which would still be a full implementation of TEX. Thus we tended to decide design details in favor of conserving simplicity of the handwritten code *vs.* saving execution time or code size.

Creating PASCHAL, Our Pascal-to-C Translator

The bulk of the TurboTEX project involved the Pascal-to-C translator, which we named `PASCHAL`. We chose a name so close to "Pascal" because we wanted to emphasize the equivalency of the C output to the Pascal input. When speaking aloud, one may pronounce the name "`PASCHAL`" with a Mediterranean accent to distinguish it from Pascal, and thereby also emphasize the etymology of both terms in Greek (*Páscha*) and Hebrew (*Pesakh*). (Those unaccustomed to these languages may simply raise the pitch of the second syllable over the first.)

We were fortunate to have had experience writing large programs in both Pascal and C, so that we had a clear understanding of the issues involved in translating one into the other. We also have written several language-based translators for other applications.

The two languages are thoroughly similar in all but a few features, and TEX is restrained in the use of Pascal's distinctives. In the cases of a few *hapax legomena*, we simply rewrote the difficult Pascal statements in a more common dialect.

Our development system, an AT&T 3B1 running Unix System V, provided the YACC and Lex tools to automate the production of `PASCHAL`'s parser. We also referred to the grammar from the Berkeley `pc` Pascal compiler (likewise written in YACC) to guide us in designing and writing `PASCHAL`'s parser.

To date, `PASCHAL` has required 240 hours of senior programmer effort and a total of 3200 lines (10,800 words) of source code in YACC, Lex, and plain C. The `PASCHAL` executable on our machine is only 62K bytes (not counting dynamic memory used for storing pending output).

The various modules of `PASCHAL` organize as follows: the main function to interpret command-line options and input file names, the lexical analyzer, the syntax tables and semantic actions of the parser, dynamic memory allocation, string handling, symbol table, parameter list interpreter, subrange interpreter, non-scalar type analyzer, variant record decomposition, and function-return-value replacement.

Besides the `PASCHAL` translator proper, we created a run-time "Pascal Compatibility Library" in C to replace the Pascal run-time library, and a header file, `paschal.h`, which contains macros used by the `PASCHAL` output.

Translating TEX to "Pure" C

We will now discuss translating TEX in Pascal to TEX in C. We will defer the details of how we translated operating-system dependencies, such as the run-time library and I/O, until the following sections on Unix and MS-DOS.

The easiest parts to translate include most of the control structures, such as compound statements, conditionals, loops, and so on. Using YACC and a few string-handling functions does the job. Certain cases, like Pascal **for** loops, have a more complex meaning in Pascal than in C, and `PASCHAL` puts in extra C statements to achieve the absolutely equivalent effect.

One aspect we handle on the lexical level is conflicts in reserved words. Some of TEX's identifiers, like *int*, are reserved words in C, so we recognize them specially and prefix a "`PCL`" on them before they reach the parser. In this way we avoid any conflicts in the C output.

Also on the lexical level, we recognize the Pascal built-in functions and procedures, and change them into the names used in the "Pascal Compatibility Library." This library eventually links with the final executable.

The lexical analyzer also handles conversion of numeric and character constants into appropriate C constants. Most operators also have a simple lexical equivalent, such as Pascal's "<>" and C's "!=".

The operators and precedence in the two languages have a few subtle differences, requiring special care in certain cases. For example, Pascal's "/" is strictly real division, while in C it is either the integer modulo or real division, depending on the operands. PASCHAL recognizes all such special cases and is careful to clothe them in appropriate parentheses and type casts. Built-in functions can also be very tricky. For example, the Pascal "abs" adapts implicitly to the type of the operand while C's "abs" is strictly integer. PASCHAL cages chameleons like this with C preprocessor macros that mimic the Pascal behavior.

Most cases of the Pascal **type** statement translate into a C **typedef**. PASCHAL does an optimal mapping of each Pascal subrange type into a C **char**, **short**, **int**, or **long**, based on the storage or execution speed profile of the target machine. PASCHAL uses the C **unsigned** modifier to optimize use of storage or execution time. PASCHAL translates Pascal **const** statements into C preprocessor #define's, thus permitting the C compiler to reduce constant expressions at compile-time.

A Pascal **record** translates into an equivalent C **struct**. In the case of variant **record**'s, C **union**'s appear. The syntax of Pascal's variant **record**'s, however, is different enough from C's **struct**'s and **union**'s to make coding of the task difficult. In some cases PASCHAL must generate extra synthetic identifiers to label parts of the C **struct**.

Arrays pose several problems. Pascal separates items in lists of array subscripts with commas, while C requires them to be each in square brackets. Pascal permits array bounds to start from any integer, while C always uses a zero lower bound. Thus PASCHAL must keep a symbol table of array names and their lower bounds, and insert an offset into each dimension of each reference to an array element.

When an array appears on the left side of an assignment, we know in the case of TEX that it is a string assignment. PASCHAL outputs C code to copy a string in that case.

In Pascal, a function or procedure call may or may not have a parenthesized actual parameter list, so the translator has to maintain a symbol table to know whether an identifier reference is to a variable or a call.

Pascal and C take different approaches to declaring formal parameters to functions and procedures. Pascal does it sensibly, whereas C expects you to recite the identifier list once in the parentheses and once again (with types this time, please) after the parentheses.

Pascal and C use radically different methods to return a value from a function. Pascal requires that the returned value be assigned to the function name and that control flow exit at the end of the function body. C simply uses a built-in "**return**" statement. Thus PASCHAL must insert a synthetic returned-value variable declaration into each function, convert all Pascal assignments of return values to assignments to that variable, and insert a **return** statement at the end of the function to return the value of the variable.

To make separate compilation possible, PASCHAL will gather external declarations for the whole Pascal program together and create a C "include" file from them. The functions of the C source may then be arbitrarily split into separate source files for separate compilation. The PASCHAL accessory program splitp, given an arbitrary number of lines, splits the PASCHAL C output into smaller source files, each containing about that number of lines.

In a few cases of Pascal language features, we just change the TEX source (with the changefile) to avoid a difficult Pascal feature that is rarely used by TEX. For example, TEX uses Pascal's non-local goto's for error termination, and we follow Knuth's suggested strategy by substituting a call to the C exit() function.

Connecting TEX to the Unix System V Environment

The matter of file I/O does not permit much in the way of automatic translation. We modified by hand TEX's code for file opening/closing, text line input, and read and write statements. The changes convert these operations to use the C standard I/O library. While this hand-crafted modification was tedious, it amounted to only a few hundred lines and was mechanical.

To the list of handmade modifications we must add the "dirty Pascal" which Knuth cites in the index to *TEX: The Program.* C is able to perform these tricks as well as Pascal.

C provides a standard way to access command-line arguments, and we provide the customary TEX features in that respect. Also straightforward in C under Unix System V are: invoking a sub-shell for editing during a TEX run, detecting an interrupt key-press, and determining the date and time.

We use a novel method made possible by C to obtain pre-loaded versions of TurboTEX. We do not process core dumps into modified executable images, which has been the strategy of such probrams as the Berkeley Unis `undump` utility. Instead, we created a utility program `fmt2init` in C, which adds C initializers to the global declarations of the TurboTEX source code, thus creating source code for pre-loaded versions of TEX. The `fmt2init` utility determines the proper global variable initializations by executing the *initialize* function from the TurboTEX source, and by analyzing the `initex` format file which defines the preloaded state. This method has several important advantages over the core-dump method: (1) Complete portability of the pre-loading process, since C initializers are portable (as opposed to core-dump files which are inherently non-portable), (2) Smaller executable files and faster program loading, since C global variables which are initially undefined or zero take no space in executables compiled from C, and (3) Portability of the initialized data, since C correctly initializes items like character data for the underlying machine architecture.

Once we had debugged our first version of TurboTEX for Unix System V to where it would process complex documents, we were ready to try the TEX `TRIP`. To our satisfaction, TurboTEX processed the entire TEX `TRIP` without error on the first attempt.

Sugar-Coating Our Translation for MS-DOS PC's

Having finished our initial version for Unix System V, we turned our attention to porting the TurboTEX translation to the IBM PC and compatibles. For this effort we chose the Microsoft C Compiler version 5.0, since this compiler offered a run-time library compatible with the Unix standard I/O library, a reputation of being bug-free, and accommodating memory models for large programs like TEX.

The main issues of porting a large program like TEX to the PC have to do with the unfriendly architecture of the 8086 processor. Its limitations arise partly from its being a 16-bit processor with a 16-bit (64K byte) address space, and partly from an illogical design to expand the address space to 20 bits (1M byte, reduced to 640K on the PC).

Taking the verified TurboTEX C source to the PC and getting the code to compile and run correctly required an unpleasant amount of effort, almost as much as that to create `PASCHAL`. However, we were aided by `PASCHAL` itself in that as we discovered quirks in the way the PC executes C, we were able to make small changes to `PASCHAL` to correct whole classes of problems in the PC's use of the source code.

The main problems were: getting the program to run in 640K, getting correct coercion of parameters in function calls, and getting I/O to work properly with the changed operating system environment.

Finishing with Output Drivers and Utilities

We ported Nelson Beebe's output drivers in C (see TUGboat Volume 8, No. 1) to Unix System V. Beebe's source code, at the time we received it, required only a few changes to compile and run perfectly. At our site we use `DVIJEP` for the HP LaserJet+ and `DVIIBM` for IBM-compatible dot-matrix printers.

The font metric and bitmap files require no translation, since they are in a completely portable format to begin with.

Encore!

We have a number of items on our wish-list of future upgrades.

In the course of translating TEX we first translated `TANGLE`. The `WEB` and other Pascal-based utilities not strictly required to run TEX (`WEAVE`, `TFtoPL`, etc.) will follow soon.

Previewers must be written for specific graphics hardware, so we will first produce them for only a few machines, namely the AT&T 3B1 and the IBM PC, in order to have control over our own versions.

We would like to turn `PASCHAL` loose on METAFONT next. The METAFONT source provides some new challenges, but we are confident that our design will adapt well to any problems involved.

As to ports to other machines, we hope to complete them for the Apple Macintosh, IBM's OS/2, and VAX Unix and VAX/VMS. TurboTEX may run unchanged on the two VAX operating systems, but we have not yet attempted that port.

Since there is no object code standard for the hundreds of various Unix System V machines, those with Unix machines which we do not support should obtain the TurboTEX source code from us and compile it for their machine.

There are several optimizations to TEX which we hope to hand-craft in C for TurboTEX, including rewriting the "inner-loop" code, optimizing near and far pointers in the PC version, and using C dynamic memory to optimize at run-time the use of memory for arrays.

We would like to improve `PASCHAL`, or possibly write a post-processor, to improve the looks of the

resulting C code. Like processed cheese, the food value of the original is there, but the flavor is changed and the texture is gone.

Distributing the Product

We have elected to make TurboTEX a "semi-commercial" product within the US, that is, we will charge a modest license fee for each copyrighted copy of the binary and/or source code. However, unlike other commercial versions of TEX, the source code will still cost less than the other's binaries. We will distribute a complete package including TurboTEX, utilities, and printer drivers.

Late-breaking News. We have completed some preliminary benchmarks on the VAX BSD version of TurboTEX, with encouraging results. We compared TurboTEX in C to the public-domain Unix TEX distribution in Pascal on a VAX 750. We observed an execution speed-up factor of between 1.6 to 3.0 compared to the Stanford distribution TEX (the factor varies depending on the type of document being formatted). The size of the TurboTEX executable code is about 60% of the distribution version.

easyTEX

Ester Crisanti
Alberto Formigoni
Paco La Bruna

1 Introduction

1.1 easyTEX$_{1.0}$

TEX has introduced new powerful tools for scientific documents typesetting, allowing formulae to be easily built up through a linear language. As a new tool using TEX, a project was born in 1984 at the Istituto di Cibernetica (now Dipartimento di Scienze dell'Informazione) dell'Università degli Studi di Milano, Italy.

That project has produced easyTEX$_{1.0}$ that we propose as a new powerful tool for TEX documents typesetting.

easyTEX is an interactive Formula Processor, developed from the initial idea of Prof. Gianni Degli Antoni, Dipartimento di Scienze dell'Informazione, planned and implemented by TE CO GRAF with the collaboration of Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Milano.

It allows the interactive typewriting of mathematical formulae on IBM-compatible Personal Computers. The formulae produced by easyTEX are memorized in ASCII standard files, prepared in order to be processed by TEX, either including such files in other ones by means of the TEX command "`\input`", or using usual editor commands for file merge.

The formula being built up is displayed on the screen through the fonts created with METAFONT and it is also possible to use every symbol and mathematical font.

The use of easyTEX is very simple, since the user is driven in his work by a pop-up menu interface, by means of which the choice of operators and mathematical symbols is easily made. It is also possible to select some virtual keyboards which, because they can be displayed on the screen, achieve a correspondence with the physical keyboard, allowing insertion of characters belonging to different alphabets, like the greek, or a wide selection of mathematical symbols.

Also, complex mathematical formulae can be typeset in an easy way, similar to the one used in writing by hand the same formula. Both for the foregoing reasons, and because the positioning of the cursor is automatically obtained through an interactive construction of the formula on the screen, easyTEX offers to the user a good facility for the preparation of a TEX document.

easyTEX has been implemented using attributed grammar techniques, as developed by D.E. Knuth. Programs have been written in C language.

2 Functional characteristics

2.1 User interface

The user communicates with easyTEX using pop–up menus making the selection of commands simple and fast. Using easyTEX, it is not necessary to know editing languages or to learn a particular syntax for the commands, because everything is done in an interactive way.

2.1.1 The screen layout

The screen handled by easyTEX is structurally divided into three separated areas named:

- **Menu line**
- **Work area**
- **Status line**

The *Menu-line* is on the upper part of the screen and displays a sequence of names each representing a **Menu name**.

The *Status-line* is on the lower part of the screen and contains a set of information concerning easyTEX' state.

The *Work-area* occupies the whole part of the screen extending between the *Menu-line* and the *Status-line* and contains the formula the user is working on.

2.1.2 The pop–up menus

Every pop–up menu contains a sequence of elements:

- *Menu-name*: its selection displays a different pop–up menu
- *Command-name*: its selection executes the associated command

Selecting commands inside the pop–up menus is very easy. You have first to activate the "menu mode" through the **F1** key. On the screen you will see the pop–up menu and any command or menu may be selected by stroking the name capital letter or positioning the cursor on the name (through the two keys <↓> and <↑>) and then stroking the < ↵ > key. If you have chosen a *Command-name*, the operation is immediately performed; on the contrary, if you have chosen a *Menu-name*, on the screen you will see the corresponding pop–up menu.

The organization of pop–up menus in easyTEX is hierarchical. At the first level there is the *Menu-line* and at the second the **O**perator and **G**eneral pop–up menus. In the **O**perator pop–up menu there are some *Menu-name*s, whose selection performs the displaying of other pop–up menus forming the third level.

2.1.3 Virtual keyboards

The selection of the characters belonging to different character fonts is made through a system of **virtual keyboards**.

easyTEX associates to every character font a **virtual keyboard**, in which a symbol of the current character font corresponds to each key of the physical keyboard. During the working session of easyTEX it is possible to display the **virtual keyboard**. easyTEX provides five **virtual keyboards**:

- Italic
- Romanic
- Boldface
- Greek
- Symbols

2.2 How to build up a formula

Suppose you to want to input the formula:

$$V(x) = \int_0^1 F(x)dx$$

You first have to write the "$V(x) =$" string; this is done very easily: the association between the **virtual keyboard** and the physical one has not changed the meaning of the keys. We have therefore the following result, with the position of the cursor represented by the box:

$$V(x) = \square$$

Now you have to write the integral. After activating the "menu mode" by the function key **F1**, from the *Menu-line* you select the *Menu-name* **O**perator. On the screen you will see, under the element selected from the *Menu-line*, the corresponding pop–up menu:

Operator	General
Triple	
Fraction	
Root	
Exponent	
Index	
Block	
Matrix	
tri**G**	
Accent	
Dots	

Let's now select the *Menu-name* **T**riple and you will see now on the screen, by the side of the element selected by the pop–up menu, the window containing the corresponding pop–up menu:

Operator	General
Triple	**I**ntegral
Fraction	**S**um
Root	limi**T**
Exponent	lo**G**
Index	**L**n
Block	**M**in
Matrix	ma**X**
tri**G**	uni**O**n
Accent	interse**C**t
Dots	pro**D**
	i**N**f
	s**U**p
	limin**F**
	limsu**P**
	overbr**A**ce
	under**B**race

Let's finally select from the **T**riple pop–up menu the *Command-name* **I**ntegral, obtaining:

$$V(x) = \int_{\square}$$

Not only the integral symbol has been displayed, but also the cursor has been correctly positioned to write the lower limit and the font in use has been reduced from size 10 pt to size 7 pt. This size reduction has also changed the cursor dimension. It is now possible to insert the lower limit, obtaining:

$$V(x) = \int_{0\square}$$

In order to set the end of the lower limit you stroke $< \hookleftarrow >$.

$$V(x) = \int_{0}^{\square}$$

The cursor is automatically positioned on the upper limit. This is written as the lower limit and, when closing it through the $< \hookleftarrow >$, you obtain:

$$V(x) = \int_{0}^{1} \square$$

At last you input the "$F(x)dx$" string, obtaining the complete formula:

$$V(x) = \int_{0}^{1} F(x)dx \,\square$$

2.3 The editing commands

2.3.1 The cursor movements

A mathematical formula consists of strings (i.e., strings of alphanumeric characters and symbols) and of a class of structures, such as fraction, triple, root and so on.

Every complex mathematical formula may therefore be decomposed into simple formulae, which are in turn reduced to single characters. For instance, in the same way as the string consists of single characters

$$\boxed{a}\boxed{+}\boxed{b}\boxed{=}\boxed{c}$$

you can single out in a fraction the numerator and the denominator; each of these may in turn consist of other mathematical structures or of strings.

$$\frac{\boxed{a+b}}{\boxed{\sin\alpha}} \Rightarrow \frac{\boxed{\textit{numerator}}}{\boxed{\textit{denominator}}}$$

$$\frac{\boxed{a+b}}{\boxed{\sin}\boxed{\alpha}} \Rightarrow \frac{\boxed{\textit{string}}}{\boxed{\textit{trigonometric, string}}}$$

The splitting of a formula into strings and structures simplifies the movements across a mathematical text. easyTEX allows two movement modes:

- by character
- by structure

The initial movement mode is by character.

Let's now see how the structure movements take place, considering a simple formula:

$$\sum_{\boxed{n}=n_0}^{\infty} (-1)^n u_n$$

As you want to move in structure mode, you have to stroke the $<$ *home* $>$ key. On the screen you will now see:

$$\sum_{\boxed{n=n}_0}^{\infty} (-1)^n u_n$$

The cursor contains the "string" structure "$n = n$". This is consistent with what we said above; actually the lower limit consists of a string and of an index, and the cursor was positioned on one of its characters. By again stroking $<$ *home* $>$ the cursor gets the dimensions of the entire summation lower limit:

$$\sum_{\boxed{n=n_0}}^{\infty} (-1)^n u_n$$

By stroking $<$ *home* $>$ again the cursor gets the dimension of the entire "Triple Summation" structure:

$$\boxed{\sum_{n=n_0}^{\infty}} (-1)^n u_n$$

and at last, by stroking $<$ *home* $>$ again the cursor gets the size of the entire formula:

$$\boxed{\sum_{n=n_0}^{\infty} (-1)^n u_n}$$

Let's now stroke the $<$ *end* $>$ key. The cursor gets the size of the summation lower limit:

$$\sum_{\boxed{n=n_0}}^{\infty} (-1)^n u_n$$

and stroking again the $<$ *end* $>$ key, the cursor gets the size of the first structure of the lower limit:

$$\sum_{\boxed{n=n}_0}^{\infty} (-1)^n u_n$$

and by typing another $<$ *end* $>$, the cursor gets the size of the first string character:

$$\sum_{\boxed{n}=n_0}^{\infty} (-1)^n u_n$$

This explanation may appear very complex, but using the $<$ *home* $>$ and $<$ *end* $>$ keys is actually very easy and allows the crossing of a formula in a faster way than the one provided by the character mode. If any mistake is made, for example by including a structure more external than the one you want, this is immediately visible, as the cursor gets always the

size of its content, and the two keys < *home* > and < *end* > let the user re-establish the desired situation.

2.3.2 Deletion of characters and structures

easyTEX provides the possibility of deleting any character or mathematical structure that the cursor movements can visit. Two keys are available for deletion:

- < *BS* > (backspace key)
- < *del* > (delete key)

Backspace The usefulness of this key is evident during the input of a formula in case of a stroking mistake. For example, in the formula:

$$\sum_{n=n_a\square}$$

We have a wrong index. By stroking < *BS* > you obtain:

$$\sum_{n=n_\square} .$$

Delete By means of this command, it is possible to delete the portion of the formula included by the cursor. The combined use of < *home* > and < *end* > makes easier the selection of what is to be deleted.

2.3.3 Insertion of characters and structures

To add a new part to the formula, it is sufficient to stroke what you want to input, with the normal procedure. On the contrary, if the new part is to be inserted in the middle of the formula, you have first to move the cursor to the position immediately after where the new part is to be placed. Considering our example, if you want to vary the lower limit by adding the "$j = 0$," string, you have to stroke the key <←>, positioning the cursor as follows:

$$x_i = -\sum_{\boxed{k}=1}^{n} \lambda_{jk} x_k$$

and then to input the string, obtaining:

$$x_i = -\sum_{j=0,\boxed{k}=1}^{n} \lambda_{jk} x_k$$

If you want to input a mathematical structure, you have to use the same method. For example, let's insert a fraction before the summation. You have to move the cursor to the Summation, by using the keys <←> and < *home* >, and you get to this situation:

$$x_i = -\boxed{\sum_{j=0,k=1}^{m,n}} \lambda_{jk} x_k$$

Let's now select from the **O**perator pop–up menu the *Command–name* **F**raction, yielding:

$$x_i = -\frac{\square}{\square} \sum_{j=0,k=1}^{m,n} \lambda_{jk} x_k$$

Let's then insert the numerator and the denominator, yielding:

$$x_i = -\frac{\omega}{\varphi\square} \sum_{j=0,k=1}^{m,n} \lambda_{jk} x_k$$

easyTEX provides also the insertion of mathematical structures and of characters after the cursor positioning.

2.4 The TEX–interface command

easyTEX$_{1.0}$ is designed only to produce formulae to be inserted into TEX documents. Therefore, after building up the formula, easyTEX has to properly organize it so that it can be processed by TEX, i.e. easyTEX has to translate it into TEX source format. TEX is able to produce a formula in two different ways:

- in text mode: when the formula is on a line with normal text
- in display mode: when the formula is alone on a line.

The two representations differ remarkably only as to the size.

2.5 How to insert the formulae into the text

The combined use of easyTEX$_{1.0}$ and of TEX allows the creation of documents consisting of normal text and mathematical formulae.

You first have to recognize the parts of the document you have to produce with easyTEX and to associate a name to each one. Then, you have to build each formula up by easyTEX and to request its translation into TEX format.

Next, you build up the current text by means of the text–editor and then, in the proper positions, you input the mathematical formulae produced by easyTEX in one of the two possible ways:

- by using the TEX command:

  `\input` < *file name* >

- by using the text–editor command to *merge* two files (for every formula).

At this point, the document is ready to be processed by TEX.

3 Future developments

Two other releases will be carried out for easyTEX.

easyTEX$_{2.0}$ will have an interactive Word–Processor that will immediately show the action of all commands.

Besides all the usual word processing functions, easyTEX will allow the interactive and automatic pagination of the text and will perform wrap–arounds, applying, if necessary, an algorithm for hyphenation.

The document layout may be established by the author or selected from a library of standard document layouts. This library may be updated and extended by the author, thus allowing him to create his own document layouts library. The author may, anyway, locally change the document layout for special purposes.

A fundamental characteristic of the Word–Processor is the usage of fonts. easyTEX enables the use of several typographical fonts; the author may select different fonts within the text, and the resulting text image will be displayed interactively on the screen yielding a WYSIWYG interactive Word–Processor and Formula–Processor. Fonts can be selected from a library.

Since easyTEX, to set a page, looks up the different sizes of characters, the space between two lines is adjusted according to the biggest character (box) of the second line; the justification, on the other hand, is carried out re–arranging adequate units (pixels) of white space between words.

easyTEX$_{2.0}$ will have a Box–Processor that will allow text integration with "objects" (i.e., texts such as spread sheet tables, and images such as pictures and drawings) produced by other systems and whose file formats are known.

Using several commands, the author will be able to define, edit (copy, move, change, etc.) and format empty "boxes" within the text, which may be filled with the "objects" created by other systems, and contained in ASCII (for instance, PostScript files) or bit–map files whose file format is known; in the latter case, the images must already have all the characteristics necessary to make them printable on the target device, as easyTEX$_{2.0}$ performs scaling of images only between printer and screen formats. Anyway, easyTEX$_{2.0}$ is not an "Image Processor"; it is able, however, to give a text–image integration.

easyTEX$_{2.0}$ will also produce a source file for TEX, for more powerful processing through *passive* commands, i.e., commands ignored by easyTEX and passed to TEX for fine–tuning purposes.

We have received some other suggestions for extensions to easyTEX, such as the integration of another environment devoted to *graphs design*, useful in industrial project design; we are now evaluating the opportunity for such extensions.

We have also been requested to design a *Document Data Base*, based on a Local Area Network among PCs and a host system and using *CD–ROMS*, able to solve documentation (also technical) problems in industrial organizations. Such a system, based on TEX and easyTEX, builds on the experience we have gained with *SDDS*, together with Mondadori publishing company, CILEA and Università di Milano, Dipartimento di Scienze dell'Informazione, as one of the DOCDEL experiments supported by the Commission of the European Communities.

4 easyTEX hardware requirements

easyTEX runs on PC–IBM and compatibles equipped with the MS DOS operating system, release 2.0 or later.

easyTEX needs one of the following graphic cards:

- Hercules or Hercules–like graphic card (720x348 pixels),
- IBM Enhanced Graphic Adapter (640x350 pixels),
- OLIVETTI M24 Graphic Card (640x400 pixels),
- NCR PC6/8 Graphic Card (640x400 pixels),
- other graphic cards compatible with those described.

5 References

TE CO GRAF snc. is a company working on electronic publishing, in collaboration with the Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Milano, Italy.

Refer to Paco La Bruna for any question.

TE CO GRAF snc
Via Plinio, 11
20129 MILANO, ITALIA
Phone: 2–20 81 50 and 2–27 80 63
Telex: "340160 PER TECOGRAF"

easyTEX$_{2.0}$ will have an interactive Word–Processor that will immediately show the action of all commands.

Besides all the usual word processing functions, easyTEX will allow the interactive and automatic pagination of the text and will perform wrap–arounds, applying, if necessary, an algorithm for hyphenation.

The document layout may be established by the author or selected from a library of standard document layouts. This library may be updated and extended by the author, thus allowing him to create his own document layouts library. The author may, anyway, locally change the document layout for special purposes.

A fundamental characteristic of the Word–Processor is the usage of fonts. easyTEX enables the use of several typographical fonts; the author may select different fonts within the text, and the resulting text image will be displayed interactively on the screen yielding a WYSIWYG interactive Word–Processor and Formula–Processor. Fonts can be selected from a library.

Since easyTEX, to set a page, looks up the different sizes of characters, the space between two lines is adjusted according to the biggest character (box) of the second line; the justification, on the other hand, is carried out re–arranging adequate units (pixels) of white space between words.

easyTEX$_{2.0}$ will have a Box–Processor that will allow text integration with "objects" (i.e., texts such as spread sheet tables, and images such as pictures and drawings) produced by other systems and whose file formats are known.

Using several commands, the author will be able to define, edit (copy, move, change, etc.) and format empty "boxes" within the text, which may be filled with the "objects" created by other systems, and contained in ASCII (for instance, PostScript files) or bit–map files whose file format is known; in the latter case, the images must already have all the characteristics necessary to make them printable on the target device, as easyTEX$_{2.0}$ performs scaling of images only between printer and screen formats. Anyway, easyTEX$_{2.0}$ is not an "Image Processor"; it is able, however, to give a text–image integration.

easyTEX$_{2.0}$ will also produce a source file for TEX, for more powerful processing through *passive* commands, i.e., commands ignored by easyTEX and passed to TEX for fine–tuning purposes.

We have received some other suggestions for extensions to easyTEX, such as the integration of another environment devoted to *graphs design*, useful in industrial project design; we are now evaluating the opportunity for such extensions.

We have also been requested to design a *Document Data Base*, based on a Local Area Network among PCs and a host system and using *CD-ROMS*, able to solve documentation (also technical) problems in industrial organizations. Such a system, based on TEX and easyTEX, builds on the experience we have gained with *SDDS*, together with Mondadori publishing company, CILEA and Università di Milano, Dipartimento di Scienze dell'Informazione, as one of the DOCDEL experiments supported by the Commission of the European Communities.

4 easyTEX hardware requirements

easyTEX runs on PC–IBM and compatibles equipped with the MS DOS operating system, release 2.0 or later.

easyTEX needs one of the following graphic cards:

- Hercules or Hercules–like graphic card (720x348 pixels),
- IBM Enhanced Graphic Adapter (640x350 pixels),
- OLIVETTI M24 Graphic Card (640x400 pixels),
- NCR PC6/8 Graphic Card (640x400 pixels),
- other graphic cards compatible with those described.

5 References

TE CO GRAF snc. is a company working on electronic publishing, in collaboration with the Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Milano, Italy.

Refer to Paco La Bruna for any question.

TE CO GRAF snc
Via Plinio, 11
20129 MILANO, ITALIA
Phone: 2–20 81 50 and 2–27 80 63
Telex: "340160 PER TECOGRAF"

Macros

A Tutorial on \expandafter

Stephan v. Bechtolsheim

Introduction

I have found from my own experience teaching TEX courses that **\expandafter** is one of the instructions that people have difficulty understanding. After starting with a little theory, we will present a number of examples showing different applications of this instruction. Later in the article we will also deal with multiple **\expandafter**s.

This article is condensed from a draft of my book, *Another Look at TEX*. See the end of this article for more information about the book.

The theory behind it

\expandafter is an instruction that reverses the order of expansion. It is not a typesetting instruction, but an instruction that influences the expansion of macros. The term *expansion* means the *replacement* of the macro and its arguments, if there are any, by the *replacement text* of the macro. Assume you define a macro \xx as follows: `\def\xx{ABC}`; then the replacement text of \xx is ABC, and the *expansion of* \xx is ABC.

As a control sequence, **\expandafter** can be followed by a number of argument tokens. Assuming that the tokens in the following list have been defined previously:

\expandafter ⟨token$_e$⟩ ⟨token$_1$⟩ ⟨token$_2$⟩ ... ⟨token$_n$⟩ ...

then the following rules describe the execution of **\expandafter**:

1. ⟨token$_e$⟩, the token immediately following **\expandafter**, is saved without expansion.
2. Now ⟨token$_1$⟩, which is the token after the saved ⟨token$_e$⟩, is analyzed. The following cases can be distinguished:
   (a) ⟨token$_1$⟩ is a *macro*: The macro ⟨token$_1$⟩ will be expanded. In other words, the macro and its arguments, if any, will be replaced by the replacement text. After this TEX will **not** look at the first token of this new replacement text to expand it again or to execute it. See 3. instead! Examples 1–6 and others below fall in this category.
   (b) ⟨token$_1$⟩ is *primitive*: Here is what we can say about this case: Normally a primitive can not be expanded so the **\expandafter** has no effect; see Example 7. But there are exceptions:
      i. ⟨token$_1$⟩ is another **\expandafter**: See the section on "Multiple **\expandafter**s" later in this article, and also look at Example 9.
      ii. ⟨token$_1$⟩ is **\csname**: TEX will look for the matching **\endcsname**, and replace the text between the **\csname** and the **\endcsname** by the token resulting from this operation. See Example 11.
      iii. ⟨token$_1$⟩ is an opening curly brace which leads to the opening curly brace temporarily being suspended. This is listed as a separate case because it has some interesting applications; see Example 8.
      iv. ⟨token$_1$⟩ is **\the**: the **\the** operation is performed, which involves reading the token after **\the**.
3. ⟨token$_e$⟩ is stuck back in front of the tokens generated in the previous step, and processing continues with ⟨token$_e$⟩.

Examples 1 and 2: Macros and \expandafter

In these examples, ⟨token$_1$⟩ is a macro. Assume the following two macro definitions (Example 1):

```
\def\xx [#1]{...}
\def\yy{[ABC]}
```

We would like to call \xx with \yy's replacement text as its argument. This is not directly possible (\xx\yy), because when TEX reads \xx it will try to find \xx's argument **without** expansion. So \yy will **not** be expanded, and because TEX expects square brackets containing the argument of \xx on the main token list, it will report an error stating that "`the use of \xx doesn't match its definition`".

On the other hand **\expandafter\xx\yy** will work. Now, before \xx is expanded, the expansion of \yy will be performed, and so \xx will find `[ABC]` on the main token list as its argument.

Now assume the following additional macro definition is given (Example 2):

```
\def\zz {\yy}
```

Observe that **\expandafter\xx\zz** will not work, because \zz is replaced by its replacement text which is \yy. But then \yy is not expanded any further. Instead \xx will be substituted back in front of \yy. In other words the expansion in an **\expandafter** case is **not** "pushed all the way"; to

accomplish a complete expansion, one should use \edef, where further expansion can be prevented only with \noexpand. This example (using \zz as an argument to \xx), which would not work with \expandafter, does work with \edef:

```
% Equivalent to "\def\temp{\xx[ABC]}".
\edef\temp{\noexpand\xx\zz}
\temp
```

As a side remark: Example 1 can also be programmed *without* \expandafter, by using \edef:

```
% Equivalent to "\def\temp{\xx[ABC]}".
\edef\temp{\noexpand\xx\yy}
\temp
```

Example 3

In this example, $\langle \text{token}_1 \rangle$ is a definition.

```
\def\xx{\yy}
\expandafter\def\xx{This is fun}
```

\expandafter will temporarily suspend \def, which causes \xx being replaced by its replacement text which is \yy. This example is therefore equivalent to

```
\def\yy{This is fun}
```

Examples 4 and 5: Using \expandafter to pick out arguments

Assume the following macro definitions \Pick... of two macros, which both have two arguments and which print only either the first argument or the second one. These macros can be used to pick out parts of some text stored in another macro.

```
% \PickFirstOfTwo
% This macro is called with two
% arguments, but only the first
% argument is echoed. The second
% one is dropped.
% #1: repeat this argument
% #2: drop this argument
\def\PickFirstOfTwo  #1#2{#1}

% \PickSecondOfTwo
% ================
% #1 and #2 of \PickFirstOfTwo
% are reversed in their role.
% #1: drop this argument
% #2: repeat this argument
\def\PickSecondOfTwo #1#2{#2}
```

Here is an application of these macros (Examples 4 and 5) where one string is extracted from a set of two strings.

```
% Define macro \a. In practice, \a
% would most likely be defined
% by a \read, or by a mark.
\def\a{{First part}{Second part}}

% Example 4: Generates "First part".
% Pick out first part of \a.
\expandafter\PickFirstOfTwo\a

% Example 5: Generates "Second part".
% Pick out second part of \a.
\expandafter\PickSecondOfTwo\a
```

Let us analyze Example 4: \PickFirstOfTwo is saved because of the \expandafter and \a is expanded to {First part}{Second part}. The two strings inside curly braces generated this way form the arguments of \PickFirstOfTwo, which is re-inserted in front of {First part}{Second part}. Finally, the macro call to \PickFirstOfTwo will be executed, leaving only First part on the main token list.

Naturally the above \Pick... macros could be extended to pick out x arguments from y arguments, where $x \leq y$, to offer a theoretical example.

Example 6: \expandafter and \read

The \expandafter can be used in connection with \read, which allows the user to read information from a file, typically line by line. Assume that a file being read in by the user contains one number per line. Then an instruction like \read\stream to \InLine defines \InLine as the next line from the input file. Assume, as an example, the following input file:

```
12
13
14
```

Then the first execution of \read\stream to \InLine is equivalent to \def\InLine{12␣}, the second one to \def\InLine{13␣}, and so forth. The space ending the replacement text of \InLine comes from the end-of-line character in the input file.

This trailing space can be taken out by defining another macro \InLineNoSpace with otherwise the same replacement text. The space contained in the replacement text of \InLine matches the space which forms the delimiter of the first parameter of \temp in the following. Here, the macro \readn reads one line from the input file and defines the

macro \InLineNoSpace as that line without the trailing space:

```
\newread\stream
\def\readn{%
   \read\stream to \InLine
   % \temp is a macro with one
   % parameter, delimited by a space.
   \def\temp ##1␣{%
       \def\InLineNoSpace{##1}}%
   \expandafter\temp\InLine
}
```

Example 7: Primitives and \expandafter

Most primitives trigger no actions by TEX because in general, primitives can not be expanded (a few primitives are treated differently). In this sense, characters are primitives also. Let's look at an example:

```
\expandafter AB
```

Character A is saved. Then TEX tries to expand, but *not* print B, because B cannot be expanded. Finally, A is put back in front of the B; in other words, the two characters are printed in the given order, and we may as well have omitted the \expandafter. So what's the point here? \expandafter reverses the order of *expansion*, not of execution.

Example 8: \expandafter to temporarily suspend an opening curly brace

\expandafter can be used to *temporarily suspend an opening curly brace.* In the following case this is done to load a token register.

```
% Allocate token registers \ta and \tb.
\newtoks\ta             \newtoks\tb

% (1) Initialize \ta to "\a\b\c".
\ta = {\a\b\c}
% (2) \tb now contains "\a\b\c".
\tb = \expandafter{\the\ta}
% (3) \tb now contains "\the\ta".
\tb = {\the\ta}
% (4) \tb now contains "\a\b\c".
\tb = \ta
```

In (1) we load the token register \ta. In (2) the \expandafter temporarily causes the opening curly brace to be hidden, so that \the\ta is evaluated, resulting in the generation of a copy of \ta's content. In (3) \ta is not copied because when a token register is loaded, the new content of it, enclosed within curly braces, is not expanded. Finally, (4) is equivalent to (2), without using \expandafter.

More theory: Multiple \expandafters

Sometimes one needs to reverse the expansion, not of two, but of three tokens. This can be done. Assume that \a, \b and \c are macros without parameters and $\texttt{\ex}_i$ stands for the ith \expandafter. Then $\texttt{\ex}_1$ $\texttt{\ex}_2$ $\texttt{\ex}_3$ \a $\texttt{\ex}_4$ \b \c reverses the expansion of all three tokens. Here is what happens:

1. $\texttt{\ex}_1$ is executed causing $\texttt{\ex}_2$ to be saved. Then TEX looks at $\texttt{\ex}_3$.
2. $\texttt{\ex}_3$ is another \expandafter: **TEX will continue performing \expandafters!**
3. \a is saved (the list of saved tokens is now $\texttt{\ex}_2$ • \a).
4. $\texttt{\ex}_4$ is now executed: \b is saved (the saved token list is now $\texttt{\ex}_2$ • \a • \b), and \c is expanded.
5. Now *all* saved tokens $\texttt{\ex}_2$ • \a • \b are inserted back in front of the replacement text of \c.
6. The first token on the main token list which resulted from the operation of the previous step is $\texttt{\ex}_2$, another \expandafter, which causes \a to be saved again, and \b to be replaced by its replacement text.
7. Finally \a is inserted back into the main token list and replaced by its replacement text. The execution continues with the first token of the replacement text of \a.

One could summarize the net effect of this sequence of \expandafters and other tokens as follows (this is important for Example 8): *in front of the expansion of* \c *the expansion of* \b *is inserted and in turn the expansion of* \a *is inserted in front of that.*

The above example can be "expanded" to reverse the expansion of any number of tokens. Very rarely the reversed expansion of four tokens \a, \b, \c and \d is needed. Assume that all four tokens are macros without parameters. Here is how this is done:

```
\let\ex = \expandafter
% 7 \expandafters
\ex\ex\ex\ex\ex\ex\ex\a
% 3 \expandafters
\ex\ex\ex\b
% 1 \expandafter
\ex\c
\d
```

In general, to reverse the expansion of n tokens $\langle\text{token}_1\rangle \ldots \langle\text{token}_n\rangle$, the ith token has to preceded by $2^{n-i}-1$ \expandafters.

Example 8: Forcing the partial expansion of token lists of \writes

\expandafter can be used to force the expansion of the first token of a delayed \write. Remember that unless \write is preceded by \immediate, the expansion of the token list of a \write is delayed until the \write operation is really executed, as side effect of the \shipout instruction in the output routine. So, when given the instruction \write\stream{\x\y\z}, TEX will try to expand \x, \y and \z when the \shipout is performed, not when this \write instruction is given.

There are applications where we have to expand the first token (\x in our example) immediately, in other words, at the time the \write instruction is given, **not** when the \write instruction is later actually performed as side effect of \shipout. This can be done by:

```
\def\ws{\write\stream}
\let\ex = \expandafter
\ex\ex\ex\ws\ex{\x\y\z}
```

Going back to our explanation of multiple \expandafters: \ws corresponds to \a, { to \b, and \x to c. In other words \x will be expanded (!!), and { will be inserted back in front of it (it cannot be expanded). Finally, \ws will be expanded into \write\stream. Now \write will be performed and the token list of the \write will be saved without expansion. But observe that \x was already expanded. \y and \z, on the other hand, will be expanded when the corresponding \shipout instruction is performed.

Example 9: Extracting a substring

Assume that a macro \xx (without parameters) expands to text which contains the two tokens \aaa and \bbb embedded in it somewhere. You would like to extract the tokens between \aaa and \bbb. Here is how this could be done:

```
% Define macro to extract substring
% from \xx.
\def\xx{This is fun\aaa TTXXTT
        \bbb That's it}

% Define macro \extract with three
% delimited parameters.
% Delimiters are \aaa, \bbb, and \Del.
% Macro prints substring contained
% between \aaa and \bbb.
\def\extract #1\aaa#2\bbb#3\Del{#2}
% Call macro to extract substring
% from \xx.
% Prints "TTXXTT".
\expandafter\extract\xx\Del
% which is equivalent to:
\extract This is fun\aaa TTXXTT
     \bbb That's it\Del
```

In a "real life example" \xx would be defined through some other means like a \read. There is no reason to go to that much trouble just to print TTXXTT.

Example 10: Testing on the presence of a substring

Now let us solve the following problem: We would like to test whether or not a macro's replacement text contains a specific substring. In our example, we will test for the presence of abc in \xx's replacement text. For that purpose we define a macro \@TestSubS as follows: (\@Del is a delimiter):

```
\def\@TestSubS #1abc#2\@Del{...}
```

Now look at the following source:

```
\def\xx{AABBCC}
% #1 of \@TestSubS is AABBCC.
\expandafter\@TestSubS\xx abc\@Del
\def\xx{AABBabcDD}
% #1 of \@TestSubS is AABB.
\expandafter\@TestSubS\xx abc\@Del
```

Observe that

1. If \xx **does not** contain the substring abc we are searching for, then #1 of \@TestSubS becomes the same as \xx.
2. In case \xx **does** contain the substring abc, then #1 of \@TestSubS becomes that part of \xx which occurs before the abc in \xx.

We can now design \IfSubString. It is a simple extension of the above idea, with a test added at the end to see whether or not #1 of \@TestSubS is the same as \xx.

```
\catcode`@ = 11
% This conditional is needed because
% otherwise we would have to call the
% following macro \IfNotSubString.
\newif\if@TestSubString
% \IfSubString
% ============
% This macro evaluates to a conditional
% which is true iff #1's replacement
% text contains #2 as substring.
```

```
% #1: Some string
% #2: substring to test for whether it
%     is in #1 or not.
\def\IfSubString #1#2{%
    \edef\@MainString{#1}%
    \def\@TestSubS ##1#2##2\@Del{%
        \edef\@TestTemp{##1}}%
    \expandafter\@TestSubS
        \@MainString#2\@Del
    \ifx\@MainString\@TestTemp
        \@TestSubStringfalse
    \else
        \@TestSubStringtrue
    \fi
    \if@TestSubString
}
\catcode`@ = 12
```

Example 11: \expandafter and \csname

A character string enclosed between \csname and \endcsname expands to the token formed by the character string. \csname a?a-4\endcsname, for instance, forms the token \a?a-4. If you wanted to use this token in a macro definition you have to do it the following way:

```
\expandafter
    \def\csname a?a-4\endcsname{...}
```

The effect of the \expandafter is of course to give \csname a chance to form the requested token rather than defining a new macro called \csname.

Summary

These examples have shown some typical applications of \expandafter. Some were presented to "exercise your brains a little bit". I recommend that you take the examples and try them out; there is very little input to enter. I also encourage you to tell Barbara Beeton or me what you think about tutorials in TUGboat. There are many more subjects which could be discussed and which may be of interest to you.

This article is, as briefly mentioned in the introduction, an adaptation of a section of my book, *Another Look At TEX*, which I am currently finishing. The book, now about 800 pages long, grew out of my teaching and consulting experience. The main emphasis of the book is to give concrete and useful examples in all areas of TEX. It contains, to give just one example, 100 (!!) \halign tables. In this book you should be able to find an answer to almost any TEX problem.

Macros for Outlining

James W. Walker
Department of Mathematics
University of South Carolina

The purpose of this note is to describe standalone macros for the preparation of outlines in the standard format. For instance, the desired output might look like:

I. Vegetables
   A. Green ones
      1. lettuce
         a. iceberg
         b. leaf
      2. Broccoli, almost universally despised by children. The strong flavor is only made palatable by quick stir-frying.
   B. white ones
      1. potatoes
      2. turnips

II. Animals.

III. Minerals.

Notice that a topic is allowed to be a paragraph, not just one line, as in topic I.A.2. I wanted TEX to take care of the counting and indentation as painlessly as possible. Something like this can be done in LATEX using nested enumerate environments, but I wanted the input format to be even simpler.

When typing an outline, it is natural to show the structure by indenting with the tab key. This is particularly easy if one has a text editor with an automatic indentation feature. With that feature, hitting the Return key produces a new line with the same amount of indentation as the previous line. When the input is typed this way, we can tell the indentation level of a topic by counting tabs. We also need to mark the beginning of a topic, since not every line begins a new topic. I chose to mark a new topic with a pound sign (#). Thus, the input to produce the outline above could look something like:

```
\beginoutline
# Vegetables
        # Green ones
                # lettuce
                        # iceberg
                        # leaf
                # Broccoli, almost
                universally despised
                by children.  The
                strong flavor is
                only made palatable
                by quick stir-frying.
        # white ones
                # potatoes
                # turnips
# Animals.
# Minerals.
\endoutline
```

To make this work, an obvious step is to make the pound sign an active character which will typeset a label for a topic. However, there is no obvious way to make it look backwards and count tabs. Therefore I decided to make the tab character active also, and make it count itself. More precisely, the first tab on a line uses `\futurelet` to see whether the next token is a tab, a pound sign, or something else. If the next token is a tab, it increments a counter, gobbles the tab, and recursively looks for more tabs. If a pound sign is the first thing after a sequence of tabs, then the macro formats a topic at the appropriate indentation level. If the first thing after a sequence of tabs is anything else, nothing happens. Notice that the pound sign comes into play as an active character only for level 1 topics, i.e., when the pound sign is not preceded by any tabs.

And now, the macros. We begin by making sure that the macros are not loaded twice, and resetting the category code of the at sign. We save the old category code of the at sign, because in some formats (e.g., AMS-TEX) the at sign might have a category code other than "other".

```
\ifx\outlineformatloaded\relax
        \endinput
\else
        \let\outlineformatloaded=\relax
\fi

\chardef\oldatsigncatcode=\catcode`\@
\catcode`\@=11
```

There is one count register for each of 5 levels of indentation, which is perhaps a bit extravagant. The counter `\outline@lastlevel` is used to write an error message if the indentation level increases by more than 1 at a time. The only parameter that should be directly altered by the user is `\outlineindent`, the width of each indentation. If this is not large enough, and if the topic numbers get large, an overfull hbox could result.

```
\newdimen\outlineindent
\outlineindent=2em

\newcount\outline@i
\newcount\outline@ii
\newcount\outline@iii
\newcount\outline@iv
\newcount\outline@v
\newcount\outline@lastlevel
\newcount\outline@levelcount
```

Next we define `\beginoutline` and `\endoutline`. Be warned that we must not format the definition of `\beginoutline` with tabs, only with spaces.

```
{%
   \catcode`\#=\active
   \catcode`\^^I=\active
   \gdef\beginoutline{%
      \par
      \bgroup
      \outline@i=1
      \outline@lastlevel=0
      \catcode`\#=\active
      \let#=\outline@topicmarker
      \catcode`\^^I=\active
      \let^^I=\outline@selfcount
   }% End of \beginoutline.
}%

\def\endoutline{%
    \par
    \medbreak
    \egroup
}%
```

A level 1 topic is marked with an active pound sign, which is let equal to the following macro.

```
\def\outline@topicmarker{%
    \par
    \parindent=\outlineindent
    \medbreak
    \hang
    \indent
    \llap{\hbox to \outlineindent{%
            \global\outline@ii=1
            \uppercase
            \expandafter
            {\romannumeral
            \outline@i}%
            .%
            \hfil
    }}% end of \hbox and \llap.
    \global\advance\outline@i by 1
    \outline@lastlevel=1
    \ignorespaces
}% End of \outline@topicmarker.
```

The active tab character is made to count tabs using the following macros. Note that the parameter of `\outline@innerselfcount` will always be an `\outline@selfcount` token, which is just counted and then thrown away.

```
\def\outline@selfcount{%
    \outline@levelcount=2
    \futurelet\next\outline@next
}%

\def\outline@innerselfcount#1{%
    \advance\outline@levelcount by 1
    \futurelet\next\outline@next
}%

\def\outline@next{%
    \ifx\next\outline@selfcount
        \let\next
            =\outline@innerselfcount
    \else
        \ifx\next\outline@topicmarker
            \let\next=\outline@subtopic
        \else
            \let\next=\ignorespaces
        \fi
    \fi
    \next
}% End of \outline@next.
```

A sequence of tabs ended by a pound sign starts a subtopic.

```
\def\outline@subtopic#1{%
    \par
    \parindent=%
      \outline@levelcount\outlineindent
    \ifnum \outline@levelcount=2
        \smallbreak
    \fi
    \advance\outline@lastlevel by 1
    \ifnum \outline@levelcount>%
            \outline@lastlevel
        \errmessage{The outline level
            can't increase by more
            than 1 at a time!}%
    \fi
    \outline@lastlevel
        =\outline@levelcount
    \hang
    \indent
    \llap{\hbox to \outlineindent{%
        \ifcase\outline@levelcount
        \or % case 1: done elsewhere.
        \or % case 2: A, B, C, etc.
            \global\outline@iii=1
            \count0=\outline@ii
            \advance\count0 by `A%
            \advance\count0 by -1
            \char\count0.%
            \global\advance
                \outline@ii by 1
        \or % case 3: 1,2,3, etc.
            \global\outline@iv=1
            \number\outline@iii.%
            \global\advance
                \outline@iii by 1
        \or % case 4: a,b,c, etc.
            \global\outline@v=1
            \count0=\outline@iv
            \advance\count0 by `a%
            \advance\count0 by -1
            \char\count0.%
            \global\advance
                \outline@iv by 1
        \or % case 5: i,ii,iii,iv etc.
            \romannumeral\outline@v.%
            \global\advance
                \outline@v by 1
        \else % all deeper levels
            $\bullet$%
        \fi
        \hfil
    }}% end of \hbox and \llap.
    \ignorespaces
}% End of \outline@subtopic.
```

The outlining macros are now complete. There is one small problem: One might occasionally need to use the pound sign for its normal TEX function of marking a parameter in a \def or \halign, inside an outline. We can make that possible by providing a macro that temporarily changes the category code of the pound sign back to normal.

```
\def\normalpoundsign{%
    \bgroup
    \catcode`\#=6
    \innernormalpoundsign
}%
\def\innernormalpoundsign#1{#1\egroup}%
```

Thus an \halign could be enclosed in \normalpoundsign{...}.

Finally we restore the at sign to its former category code.

```
\catcode`\@=\oldatsigncatcode
```

A Macro Writing Tool: Generating New Definitions

Amy Hendrickson
TEXnology Inc.

Suppose you come upon a situation where you need a macro which will generate another new macro every time it is used. I came upon a solution to this problem and want to share it with TUG readers in case someone would find it an useful macro writing tool, or maybe just find it amusing.

The problem that I ran into that necessitated this kind of macro (it is by no means the only application) had to do with a set of macros that I was writing recently for slide generation: How can you take large chunks of text possibly containing tables, listings, verbatim text, or section headers, and a) print the chunk where it appears in the document, then b) send it to the end of the file to be printed in slide format. (This format would include larger font and baselineskip, possibly be in landscape mode, and have rounded corner edging.)

Since you cannot send a large body of text to an auxiliary file, the solution seemed to be to write one macro which would generate as many definitions as there were chunks of text to be made into slides, and send only the control sequence and slide formatting information to an auxiliary file. The auxiliary file can then be input at the end of the original file, and the definitions that were made earlier in the file will produce the slides.

But how can one generate such a series of definitions, each with a new name? The solution involves using the letters of roman numerals as the name of the each new macro. A counter is advanced to produce a new roman numeral each time the macro is used. With the right macro expansion, the roman numerals will be interpreted as a sequence of letters, and a new sequence of letters will be available each time.

For instance, say we set the counter equal to 637 to start, and advance it by one every time the macro is used. The first set of letters that will become a control sequence will be \dcxxxvii, the second \dcxxxviii, etc.

To make certain that these letters have not already been used in a definition, we can also supply, following the roman numeral, a sequence of letters that does not change, and thus make the possiblity of renaming a previously defined control sequence very small. That is the function of the \unique definition below.

Here is some code, showing how \newdefs can be used to define #1 as a new definition every time the macro is used.

```
\newcount\definitionnum  \definitionnum=2001

\def\newdefs#1{\advance\definitionnum by 1
  \def\unique{\the\definitionnum ZZZZ}
  \expandafter\gdef
    \csname\romannumeral\unique\endcsname{#1}}
```

In use,

```
\newdefs{This is a chunk of text}
```

will produce

```
\gdef\mmiiZZZZ{This is a chunk of text}
```

a control sequence that can be called for later in the file in whatever application it might be useful.

French in TEX

Alonzo Gariepy

Abstract

This paper describes a method of producing French documents with TEX that is much simpler than the other available alternatives. No preprocessing of `tex` files is required and the system operates with standard versions of the TEX program, the Computer Modern pixel files, and available device drivers. For IBM PC systems, accented letters can be directly entered from the 8 bit graphics character set. The preloaded version of TEX encompassing all of these changes is called FTEX. It fully hyphenates and kerns French text containing lowercase accented letters. I have used the work of M.J. Ferguson and J. Désarménien where applicable.

1 Introduction

TEX is particularly well known for the variety of its symbols, the accuracy of its hyphenation, the beauty of its output, and the standardization of its various implementations. All of these things suffer when TEX is used for French language typography.

The Computer Modern fonts are as much a part of standard TEX as the program itself. They do not contain accented letters, but provide separate letters and accent symbols that can be combined using TEX's `\accent` primitive. Computer Modern does not contain French quote characters (guillemets) or flattened accents for use with uppercase letters. So the symbol set is not really suitable for typesetting French.

Rules for hyphenating the French language can be formulated much more simply than can those for English. The comprehensive algorithm used by TEX handles French with a minimal set of hyphenation patterns. But the `\accent` primitive inserts *explicit kerns* when forming accented letters, and TEX has been specifically designed not to hyphenate in the vicinity of an explicit kern.

One of the things that makes TEX's output so beautiful is the automatic kerning of letter pairs. But the manner in which the `\accent` primitive operates, prevents this automatic kerning within French text.

Editor's note: The techniques described here seem to be architecture-dependent. Production of this article was not possible on either a TOPS-20 or a VAX/VMS system, but required that TEX be run on an IBM PC compatible, and the `.DVI` file transferred to the VAX for printing.

As a result of these three factors, any attempts at using TEX for French language typesetting have necessitated major deviations from the standard system. Three approaches to French language TEX are described in the following paragraphs.

One approach to this problem, implemented by W. Appelt, involves changing the `tfm` files so that accent/letter pairs become ligature characters with positions above '177 in the fonts.[1] Such ligatures have no character pattern, but are assigned attributes identifying the associated accent and letter. The `tfm` file can be modified so that automatic kerning takes place around the ligature. A special device driver must be created to output this dummy ligature using the character patterns for the accent and letter. The `dvi` file cannot be printed without the special driver and modified `tfm` files.

J. Désarménien came up with another approach that relies on Computer Modern based French fonts incorporating already-accented letters.[2] Some room can be found for these letters at the beginning of a Computer Modern text font by eliminating all the uppercase Greek letters. The vowels à and ù occur only in the words 'à' and 'où' and therefore need not be included in the French fonts for the purposes of hyphenation. These vowels may still be accented with the `\accent` primitive. The problem with this approach is the necessity to maintain a complete set of French Computer Modern `tfm` and pixel files in all needed sizes, resolutions, and magnifications. Such maintenance demands the possession and use of METAFONT and the storage and distribution of large amounts of data.

More recently, M.J. Ferguson has made changes to the TEX program itself to produce a multilingual version called TÊX.[3] This program circumvents the restrictions within TEX that prevent hyphenation of words containing accented letters. To this it adds a facility for loading multiple sets of hyphenation patterns and switching between them. There is no indication that TÊX will automatically kern accented letters from standard Computer Modern. Alas, TÊX is not really TEX.

[1] W. Appelt: *The hyphenation of non-english words with TEX*. Proceedings of the First European Conference on TEX for Scientific Documentation, Addison-Wesley, 1985, pp. 61–65.

[2] J. Désarménien: *How to run TEX in a French environment: hyphenation, fonts, typography*. TUGboat, Vol. 5 (1984), No. 2, pp. 91–102.

[3] M.J. Ferguson: *A Multilingual TÊX*. TUGboat, Vol. 6 (1985), No. 2, pp. 57-58.

2 French TEX for the IBM AT

At my installation we are using Addison-Wesley's MicroTEX (written by David Fuchs). We produce large quantities of unilingual French and English documents. The software is run on IBM ATs. MicroTEX and its French partner, FTEX, operate as components of an interactive text management system (TMS).

Computer editing French language documents that contain normal TEX accent macros (such as \^, \c, \' and \") is very awkward. With the aid of a macrokey program and the IBM PC graphics character set, accented letters can be typed directly into the word processing facilities of the TMS.

With limited disk space on both production and development machines, it is crucial that the TMS and TEX leave enough room for a large quantity of data. The space used by the files and programs needed to develop, distribute, operate, and maintain FTEX must be minimized.

The system maintenance and user support will be performed by a single individual. This will involve a large amount of user training and some TEX macro writing, leaving little time for anything else.

We expect the materials created with this system to have a long lifespan and eventually to be made available for interactive retrieval.

Thus, the constraints for our French language version of TEX:

- a minimum of software maintenance
- minimal storage requirements
- no maintenance of fonts
- easy distribution
- high quality output
- direct editing of accented text on screen
- portability of `tex` and `dvi` files

The three approaches to a French TEX already described were ruled out because of the requirement for developing and maintaining modified versions of device drivers, pixel files, or the TEX program itself.

2.1 Accenting with FTEX

FTEX produces words that can be hyphenated, by forming accented letters in a way that makes use of implicit kerns instead of explicit ones. This kerning is specified by the ligature/kern table in a `tfm` file modified for French.

To produce an accented letter, FTEX inserts three characters in the form: ⟨letter⟩⟨accent⟩⟨letter⟩. For example, on encountering `\^e` in the input file, FTEX will insert `e^e` into the `dvi` file, including the new implicit kerns from the modified `tfm` file.

This works in the following way:

- accent characters in Computer Modern (CM) are already at the right height to go above the lowercase letters
- words that contain uppercase letters will not often need hyphenation and can be accented in the conventional way
- letters that come before and after the `\^e` will automatically kern with the sequence `e^e` in the same way they would with a single `e`
- the three characters `e`, `^`, and `e` can be centered on one another by interposing, between the `^` and each `e`, identical kerns of value $-(\text{width}(\texttt{e}) + \text{width}(\texttt{\^{}}))/2$
- the sum of the widths of these three characters and the two kerns is equal to that of a single `e`
- font substitution (widely available on TEX's device drivers) allows the standard pixel files to be used in concert with FTEX's `tfm` files.
- the three characters, when superimposed on an output device, appear as the printed character pattern ê.

2.2 Modifying `tfm` files

I have written a program that produces a French `tfm` file (eg., `fmr10.tfm`) from the standard Computer Modern `tfm` file (eg., `cmr10.tfm`). The program, called `fkern`, actually works with `pl` files, which are easier to parse and modify, and leaves the translation between `tfm` and `pl` formats to the utilities `tftopl` and `pltotf`.

`Fkern` reads the widths of all of the accents and accentable letters, and adds new entries to the kern table for accent/letter pairs that occur in French. These kerns are negative and equal to the average of the widths of the two characters involved.

There are three strategies for integrating French `tfm` files into your TEX system. The first, already mentioned, is to use FM fonts in your TEX file, but substitute CM fonts when you run the device driver. A second way, requiring more disk storage, would be to make exact copies of all the CM pixel files, naming them FM instead.

The third alternative is to directly modify the CM `tfm` files and use them both for TEX and FTEX. The unusual kerns for French will not cause problems for most applications and could be used to create accented letters in TEX as well. This approach avoids the necessity of redefining the various fonts used by `plain.tex` or LATEX, and setting up font substitutions, but then the names of the files would not distinguish them from the standard set.

2.3 Inputting accented letters

Accented characters from the graphics portion of the IBM PC character set can be directly entered using many IBM PC editors (sometimes with the help of a macrokey program). The TMS we use for entering and managing our documents, allows us to do this.

One solution to the problem of converting these extended (8 bit) characters to a form that TEX can work with would exploit the ability of some editors to substitute any string of characters during output. The substitution facility of such an editor could be customized to convert the IBM graphic character ê to the TEX sequence `\^e` (or `e^e`). Unfortunately, the printer driver for our TMS and many editors does not allow this kind of thing.

When MicroTEX inputs an extended character, it entirely ignores the eighth bit, effectively subtracting 128 from the value of the character. Serendipitously, all French lowercase accented letters in the graphics character set fall into the range from 1 to 23 when the eighth bit has been stripped. These characters may be made active and defined to substitute the ⟨letter⟩⟨accent⟩⟨letter⟩ sequence described above.

One must be careful that the extended characters moved into this range do not conflict with character codes in use by TEX. Fortunately, none of these conflicts with ⟨return⟩. The characters ü, ë, ï and î conflict with ⟨control A⟩, ⟨tab⟩, ⟨control K⟩ and ⟨form feed⟩ defined by `plain.tex` to be equivalent to ⟨Subscript⟩, ⟨Space⟩, ⟨Superscript⟩ and ⟨`\par`⟩. The codes ⟨control A⟩ and ⟨control K⟩ are assigned this way because they correspond to the characters ↓ and ↑ on some non-PC keyboards. FTEX supersedes these definitions. The ë and î may safely be used as long as the input file contains no ⟨tab⟩ or ⟨form feed⟩ characters. These four accented vowels are used rarely enough in French that one could do without the direct entering of them, in order to avoid conflicts.

There are two French uppercase accented letters in the graphics character set. The É can be declared active but, due to its height, must be accented by TEX in the conventional manner. The Ç conflicts with the ASCII NULL when the eighth bit is stripped. FTEX currently uses no uppercase accented letters from the graphics character set.

The guillemets in the IBM graphics character set conflict with alphabetic characters when the eighth bit is stripped. The < and > characters can be used for this purpose in text input while retaining their meanings as relational operators in math mode. There are several ways, using Computer Modern, to create guillemets ≪ of a sort ≫.

The direct entry of 8 bit characters is very system dependent. This feature can be removed from FTEX for non IBM PC systems.

2.4 FTEX and hyphenation

FTEX's French hyphenation patterns are translated from those developed by M.J. Ferguson based on work by J. Désarménien. Examples of the kind of pattern that FTEX needs to perform its hyphenation include `.de´e3s2e´e3gr` and `1c¸c`. Direct entry of accented characters has been made to apply to the hyphenation patterns as well, so that if you edit `ftex.tex` you will actually see lines that look like `.dé3s2é3gr` and `1ç`. Hyphenation exceptions can also be entered in this fashion.

3 Putting it all together

The files that are required to create the preloaded version of FTEX include `ftex.tex`, `plain.tex` and the French Modern `tfm` files created by `fkern`. Two TEX primitives must be temporarily modified when inputting `plain.tex`. The `\patterns` primitive is redefined so that the English hyphenation patterns will be ignored, and the `\font` primitive is redefined to substitute `fm` fonts for `cm` or `am` fonts. A French version of LATEX can be generated this way if there is enough memory for FTEX's larger `tfm` files.

With the FM versions of all the necessary fonts preloaded, FTEX can be distributed as a single file (either `ftex.fmt` or `ftex.exe`). Alternatively, the utilities `fkern`, `tftopl`, and `pltotf` can be included for conversion of `tfm` files on site.

3.1 Limitations

FTEX formats French text only. Words containing uppercase accented letters cannot be hyphenated.

In this version of FTEX, none of the accented characters has been given a `\uccode`. For the `\uppercase` operation to have any effect it should be performed before the active characters have been expanded. Two approaches to setting `\uccodes` are demonstrated below:

```
%
%   if you want the uppercase letter accented
%
{\catcode`A=13 \gdefA{\'E}} \uccode`\é=`A
{\catcode`B=13 \gdefB{\^E}} \uccode`\ê=`B
%   etc.
```

```
%
%   if you do not want it accented
%
{\catcode`A=13 \gdefA{E}}    \uccode`\é=`A
\uccode`\ê=`A \uccode`\ë=`A \uccode`\è=`A
%   etc.
```

TEX device drivers must be careful how they correct the incremental roundoff errors accumulated while setting the letters in a word. The algorithms used ensure identical spacing within a word wherever it is used, while maintaining a correspondence between dvi coordinates and actual pixels. But, in the short run, one cannot be sure that two characters with the same dvi coordinates will be rounded to the same pixel. The obvious impact of this on FTEX is that identical superimposed letters may end up one pixel out of alignment, creating a slightly thicker letter. I believe that a minor improvement to the way that the device drivers are written will remove this imperfection.

4 The source of ftex.tex

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                             %
%  FTEX, Copyright 1987 by Alonzo M. Gariepy  %
%                                             %
%  NOTICE!  This file contains IBM graphics   %
%           characters                        %
%                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
\catcode`\{=1 % begin-group character
\catcode`\}=2 % end-group character
\catcode`\$=3 % math shift
\catcode`\&=4 % alignment tab
\catcode`\#=6 % macro parameter character
%
%
\let\fpatterns=\patterns  % save primitive
\def\patterns#1{}    % disable for English
%
\let\ffnt=\font      % save primitive
\def\font#1=#2#3{\ffnt#1=\ifx#3mfm\else
   #2#3\fi}
%
     \input plain
%
%    \input lplain
%
\let\patterns=\fpatterns     % enable
\let\font=\ffnt
%
\def\multi#1#2#3{\def\multI##1{\ifx\end##1\else
   #1##1#3\expandafter\multI\fi}\multI#2\end}
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Define macros to represent the accents as
%   category 12.  Assign appropriate lccodes.
%
\begingroup    % so category changes are local
%
\catcode`+=7     % we'll be using ^ temporarily
\catcode127=12   % make DELETE valid character
%
\multi{\catcode`}
    {++S ^ ++? ++R ++X ++P ++[ ++^}{=12}
%
\def\fdef#1=#2{{\global\def#1{#2}
    \global\lccode`#2=`#2}}
%
\fdef\Aa=++S
\fdef\Ac=^
\fdef\Ad=++?
\fdef\Ag=++R
\fdef\Cc=++X
\fdef\i =++P
\fdef\oe=++[
\fdef\OE=++^  \lccode`++^=`++[
\fdef\Ap='
%
\endgroup
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Activate the IBM PC graphics characters that
%  correspond to the lowercase French accented
%  letters. Comment out this section if you are
%  not using an IBM PC.
%
%  Let ^^A and ^^K be active in verbatim modes
%
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
    \do\#\do\^\do\_\do\%\do\~}
%
\multi{\catcode`}
    {\ü \é \â \à \ç \ê \ë \è \ï \î \ô \ö \û \ù}
    {=13}
%
\def ü{u\Ad u}         % 1     129
\def é{e\Aa e}         % 2     130
\def â{a\Ac a}         % 3     131
\def à{a\Ag a}         % 5     133
\def ç{c\Cc c}         % 7     135
\def ê{e\Ac e}         % 8     136
\def ë{e\Ad e}         % 9     137
\def è{e\Ag e}         % 10    138
\def ï{\i\Ad\i}        % 11    139
\def î{\i\Ac\i}        % 12    140
\def ô{o\Ac o}         % 19    147
\def ö{o\Ad o}         % 20    148
\def û{u\Ac u}         % 22    150
\def ù{u\Ag u}         % 23    151
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Modify plain's accent macros so they make
%   lowercase French accented letters with FTEX
%
\begingroup
%
\def\ftxset#1#2{\expandafter\gdef
    \csname#1#2\endcsname{#1}}
%
\ftxset\Aa e
\ftxset\Ac a
\ftxset\Ac e
\ftxset\Ac\i
\ftxset\Ac o
\ftxset\Ac u
\ftxset\Ad e
\ftxset\Ad\i
\ftxset\Ad o
\ftxset\Ad u
\ftxset\Ag a
\ftxset\Ag e
\ftxset\Ag u
\ftxset\Cc c
\endgroup
%
\def\ftxacc#1#2{\if#1\csname#1#2\endcsname
    #2#1#2\else {\accent\expandafter`#1#2}\fi}
%
\def\`{\ftxacc\Ag}
\def\'{\ftxacc\Aa}
\def\^{\ftxacc\Ac}
\def\"{\ftxacc\Ad}
\let\ftexc=\c           % save cedilla macro
\def\c#1{\if\Cc\csname\Cc#1\endcsname
    #1\Cc#1\else\ftexc#1\fi}
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   French spacing macros by J. Désarménien
%   given in TUGBoat, Volume 5 (1984), No. 2
%
\frenchspacing
\catcode`\;=13
\catcode`\:=13
\catcode`\!=13
\catcode`\?=13
\def;{\relax\ifhmode\ifdim\lastskip>0pt\unskip
  \kern\fontdimen2\font
  \kern-1.2\fontdimen3\font\fi\fi\string;}
\def:{\relax\ifhmode\ifdim\lastskip>0pt
  \unskip\nobreak\ \fi\fi\string:}
\def!{\relax\ifhmode\ifdim\lastskip>0pt
  \unskip\kern\fontdimen2\font
  \kern-1.2\fontdimen3\font\fi\fi\string!}
\def?{\relax\ifhmode\ifdim\lastskip>0pt
  \unskip\kern\fontdimen2\font
  \kern-1.2\fontdimen3\font\fi\fi\string?}
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Hyphenation patterns for French.  Accented
% letters in this table can be represented in
% any of three forms for FTEX:
%
%      1)  e\Ac e   or  \i\Ad\i
%      2)  \^e      or  \"\i
%      3)  ê        or  ï
%
% On non-IBM PC systems where you are not
% using the graphics characters, you will
% need to use grep or an editor to put the
% patterns into one of the other forms.
%
\patterns{
2'2 'a2 'é2 'e2 'o2 'ö2 'u2 'i2 .é2 1ba 1bâ 1be
1bé 1bè 1bê 1bi 1bî 1bo 1bô 1bu 1bû 1by 4be.
     ⋮
en1i2vr .en1o2 .eu2r1a2 extra1 extra2c extra2i
hémi1é hémo1p2t hypera2 hyperé2 hyper\oe2
hyperi2 hypero2 hypers2 hyperu2 hype4r1 hypo1a2
     ⋮
archi1é2pis moye2n1â2g po1astre uni1o2v
uni1a2x vélo1s2ki vol2t1amp tachy1a2 tchin3t2
chlo2r3a2c chlo2r3é2t n3s2at. n3s2ats.
}
```

German TEX

Hubert Partl
EDP Center of the
Technical University Vienna

Although TEX and LATEX have been designed for American standards only, they are being used all over the world and with a lot of different languages. This article is intended to show an example of the problems that arise when modifying TEX or LATEX for easier application with a language other than English.

One of the great advantages of TEX and LATEX is the portability of document files among all TEX installations. In order to prevent users from each inventing their own incompatible modifications, which would destroy that portability, the first step should be to standardize the user interface — i.e. the control sequences and commands to be used in the TEX input files. Together with this "standard", a "quick and dirty" or "prototype" solution should be provided, so that users can start to apply the new features. Then, usually in several steps, better and more complete and finally even optimized solutions should be developed in such a way, that the users' (authors') input files need not be changed, but only the style files, font files, hyphenation patterns and other files that comprise a TEX implementation are replaced or improved by the installation's TEX guru.

This is the way that has been adopted for "German TEX". As Joachim Lammarsch reported in TUGboat No. 8/3, the German TEX Users Group has agreed on a standard for a "Minimal Subset of German TEX Commands" at its 6th meeting in Münster (Germany) last October. These commands will make it easier to set German texts — both with Plain TEX and with the commonly used macro packages like LATEX, AMS-TEX etc. It is recommended that all TEX installations in the German speaking countries (Germany, Austria, Switzerland) implement at least these commands on all their mainframes and Personal Computers. Then, all TEX and LATEX input files that use these commands can be exchanged freely among all participating sites.

1 The User Interface

The German TEX commands fall into two categories:

- commands that provide additional features which are needed to typeset German texts, e.g. the German „Anführungszeichen",
- "shorthand" commands that are easier to type than the corresponding original TEX commands, e.g. `"s` instead of `\ss{}`, or `"ck` instead of the corresponding `\discretionary` command.

The standardized "Minimal Subset of German TEX Commands" consists of the following control sequences and commands:

- `"a` for the umlaut a (ä, short for `\"a`) — also for the other vowels,
- `"s` for the sharp s (ß, short for `\ss{}`),
- `"ck` for 'ck' that is to be hyphenated as 'k-k',
- `"ff` for 'ff' that is to hyphenated as 'ff-f' — also for certain other consonants,
- `"‘` or `\glqq` for German left double quotes and `"’` or `\grqq` for German right double quotes, to produce „deutsche Anführungszeichen", also known as „Gänsefüßchen",
- `\glq` for German left single quotes and `\grq` for German right single quotes, to produce ‚einfache Anführungszeichen',
- `"<` or `\flqq` for French left double quotes and `">` or `\frqq` for French right double quotes, to produce «French quotes», also known as «guillemets». These quotes are also used in certain German text styles, sometimes pointing »in« rather than «out».
- `\flq` for French left single quotes and `\frq` for French right single quotes ‹like this›,
- `"|` to disable forbidden ligatures in words that consist of several parts — e.g. to produce 'Auflage' (not 'Auflage') for the word meaning 'Auf-Lage',
- `"-` to mark a hyphenation exception within a long word (like `\-`, but without disabling automatic hyphenation in the rest of the word),
- `""` for an analogous hyphenation exception, where *no* hyphen sign is added in the case of hyphenation (e.g. in a hyphenated word like 'Eingabe-File'),
- `\dq` to print the quote character (").
- `\setlanguage{\xxx}` to switch to the language '*xxx*'. Arguments to this command are predefined command names like `\german`, `\austrian`, `\english`, `\USenglish`, `\french` etc. This command will switch everything that is language specific, e.g. the format of today's date, and the texts of the captions used with chapters, tables, figures and the like. In a more complete implementation, this should also include language specific hyphenation patterns and exceptions, special fonts or ligatures, different enumeration conventions

and so on. However, the German TEX commands remain available, regardless of the language specified. This is useful for multi-lingual documents, e.g. an article that is written in English but contains German citations (like this one).

- `\originalTeX` to reset everything to its original meaning in TEX or LATEX. This is needed to generate environments that are completely compatible with the rest of the TEX world.
- `\germanTeX` to switch on the German TEX commands (modifications) again.

The last three commands are usually applied locally for different parts of a multi-lingual document. They have been designed in such a way that they can be easily extended to other languages in the obvious way. The author expresses his hope that other national TEX Users Groups will adopt similar or perhaps even compatible conventions for their language specific TEX modifications.

Both the shorthand forms and the original forms of the commands for the Umlaute and for the sharp s will be modified so that automatic hyphenation remains in effect — either for the whole word (with Umlaute and sharp s included in the hyphenation patterns) or at least for the rest of the word, which can be accomplished by using constructs like

```
\nobreak\hskip\z@skip
```

which make TEX "think" that a new word is started after the umlaut. (This trick has been found and reported by Norbert Schwarz from Bochum.)

The standard does *not* include layout conventions. On the contrary, a variety of document layouts is encouraged. As with conventional typesetting methods, all authors, editors, and institutions should be free to chose their individually preferred document styles and should not be forced to an unnatural uniformity.

Tables 1 and 2 show examples for typical applications of the German TEX commands.

Table 1: Examples

`sch"on`	produces:	schön
`Stra"se`	produces:	Straße
`"'Ja, bitte!"'`	produces:	„Ja, bitte!“
`"<Merci bien!">`	produces:	«Merci bien!»
`Dru"cker`	produces:	Drucker or Druk-ker
`Ro"lladen`	produces:	Rolladen or Roll-laden
`Auf"|lage`	produces:	Auflage

Table 2: Date Formats

`\setlanguage`	`\today`
`\german`	31. Januar 1988
`\austrian`	31. Jänner 1988
`\english`	31st January 1988
`\USenglish`	January 31, 1988
`\french`	31 janvier 1988

With Plain TEX, the German commands are made available by an input command like

```
\input german
```

With LATEX, they are made available by specifying the document style option `german`, e.g. with

```
\documentstyle[11pt,german]{article}
```

In addition, the user should take care that the correct hyphenation patterns for his language are used — usually by specifying the appropriate format file when calling the TEX program. Among the German TEX users, the German hyphenation patterns generated by Jost Krieger and Norbert Schwarz at the University of Bochum are the preferred ones.

2 The Present Solution

A "quick and dirty" realization of these German TEX commands has been compiled by the author with the help of several other TEX users in Basle, Bonn, Bochum, Darmstadt, Stuttgart, and Vienna. The file, known as `GERMAN.TEX` or `GERMAN.STY`, is public domain. Mainframe installations can obtain it via Electronic Mail from several file servers: ArpaNet users can `FTP` it from the Rochester LATEX Style File Collection, and BitNet users can `GET` it from `NETSERV AT AEARN` in Linz (Austria) or from `LISTSERV AT DHDURZ1` in Heidelberg (Germany). PC users can obtain it on floppy disk from the German PCTEX distributor.

Besides being quick and dirty, this solution has the advantage that it can be used with the original versions of TEX and LATEX and with the fonts and hyphenation patterns as they are available now. Everything is defined and re-defined using TEX commands only, and it is just one TEX input file that can easily be ported to every computer (including Personal Computers) and is independent of the output devices used.

Care has been taken to make the same file usable both with Plain TEX and with LATEX and other macro packages. This has been accomplished by us-

ing Plain TEX commands only, with the only exception of the LATEX command \protect which is defined to \relax within this file if it has not been defined before.

The umlaut accent is redefined such that with the letters A, a, O, o, U, and u, the dots are positioned a bit lower than in the original version, and that the commands \nobreak and \hskip are added to enable automatic hyphenation in the rest of the word, as mentioned above.

For the sharp s (ß), the command \lccode is used to enable automatic hyphenation in words containing this letter.

The German left double quotes („) are formed by taking the English right double quotes (") and lowering them by the height difference between quotes and comma, with some extra kerning. The German right double quotes (") are the same as the English left double quotes except for the kerning. The German single quotes are formed in a similar way.

For the French quotes, the appropriate math symbols are used.

The quotes character (") is made an active character and is defined as a control sequence that takes the following character as its parameter and, depending on the value of this character, does the appropriate actions, i.e. it prints the corresponding umlaut or sharp s or quotes character, or it performs the required combination of \discretionary, \nobreak, and \hskip commands.

The quotes character is added to the \dospecials command which is used in the verbatim environments.

The different versions of today's date are obtained by re-definitions of the \today command in analogy to the original definition by Leslie Lamport.

The different versions of the chapter and table titles are obtained in the following way: The language changing commands re-define command names like \contentsname to contain the appropriate texts (e.g. 'Inhalt' for German texts and 'Contents' for English texts). With Plain TEX or other macro packages, this will have the desired effect only if these command names are actually used to print the respective title lines. With LATEX, it means that the original document style files have to be modified in the following way: The hard coded English words (like 'Contents') have to be replaced by the corresponding command names (e.g. \contentsname), and these command names have to be defined to contain the original words, e.g. with

```
\def\contentsname{Contents}
```

Leslie Lamport's comments in the DOC-Files provide help in finding all places where such modifications are necessary. There has been some discussion recently, whether these modified style files should be available from one official central "clearinghouse".

The three language switching commands are defined to switch on and off all the appropriate modifications. Finally, the command \germanTeX is executed, which switches on everything that is appropriate for typesetting German texts.

3 Future Work

For the future, a better realization of the German TEX commands is planned by a team of advanced TEXperts in Germany. This solution will include the following features:

- The Umlaute and special quotes will be designed with METAFONT as separate characters in the text fonts, and they will be accessed as ligatures.
- New hyphenation patterns will be generated that include the umlauted characters, the sharp s, the special ck, and the special double consonants that hyphenate as triple consonants.
- The multi-lingual TEX software will be used to enable the switching of hyphenation patterns for the different languages.

Due to the complexity of this project, it will take some time until this solution becomes available for all TEX installations (i.e. all computer types and all fonts for all output devices). However, the user interface (i.e. the TEX commands described above) will remain unchanged with this new solution. Therefore, users who start using them now will not have to change their TEX input files later, and they will still be able to exchange their TEX files with all installations where either the present or the future version of the German TEX commands is installed.

LaTeX

Contents of LaTeX Style Collection as of 4th February 1988

Ken Yap
University of Rochester

The LaTeX style collection now contains the files listed below. They are available for anonymous ftp from `cs.rochester.edu` in directory `public/latex-style`. You should retrieve the file `00index` first to obtain a brief description of current directory contents. The file `00directory` contains a reverse time sorted list of files; this may be helpful in keeping your collection in sync with LaTeX-style.

File	Description
`00directory`	
`00index`	
`00readme`	
`a4.sty`	Set page size to A4
`a4wide.sty`	Adjusts width too to suit A4
`aaai-instructions.tex`	Instructions to authors
`aaai-named.bst`	BiBTeX style to accompany `aaai.sty`
`aaai.sty`	Style file for AAAI conference 1987
`acm.bst`	ACM BibTeX style
`agugrl.sty`	AGU Geophysical Research
`agugrl-sample.tex`	Letters style, sample
`agujgr.sty`	AGU Journal of Geophysical
`agujgr-sample.tex`	Research style, sample
* `alltt.sty`	Like verbatim, but permits other commands inside
`amssymbols.sty`	Load AMS symbol fonts
* `apalike.doc`	American Psychological
* `apalike.sty`	Association style files
* `apalike.bst`	requires BiBTeX version 0.99a
`article.txt`	Standard files in text format,
`art10.txt`	with places to make
`art11.txt`	language specific
`art12.txt`	changes indicated
`biihead.sty`	Underlined heading
* `btxbst1.doc`	A master file for BibTeX styles
* `btxbst.readme`	with standard styles and some new ones
`cyrillic.sty`	Load cyrillic font
`dayofweek.tex`	Macros to compute day of week and phase of moon
	Examples of how to use TeX arithmetic capabilities
`deproc.sty`	DECUS Proceedings style
`deprocldc.tex`	Paper that describes the above
`docsty.shar`	Program to convert `.doc` to `.sty` by stripping comments
`doublespace.sty`	Double spacing in text
`draft.sty`	Draft option for documents for "debugging"
`drafthead.sty`	Prints DRAFT in heading
`dvidoc.shar1`	Sh archive of `DVIDOC`, DVI
`dvidoc.shar2`	to character device filter for Unix BSD systems
`epic.shar1`	Sh archive of extended
`epic.shar2`	picture environment
`espo.sty`	Style file for Esperanto
`format.sty`	Print FP numbers in fixed format
`fullpage.doc`	Get more out of a page
`fullpage.sty`	
`geophysics.sty`	Geophysics journal style
`german.sty`	Style file for German
`ieeetr.bst`	IEEE Transactions BibTeX style
`ist21.sty`	IST21 document style option for cover page
`latex.bug`	Latest listing of bugs found in LaTeX
`layout.readme`	Prints nice diagram
`layout.tex`	showing page parameters
`lcustom.tex`	Useful macros and definitions for LaTeX
`lfonts_ams.readme`	Use AMS symbols in LaTeX
`lfonts_ams.tex`	
`lgraph.shar`	Sh archive of data to graph command filter in Pascal
`local-suppl.tex`	Supplement to local guide; describes `tgrind`, `sfwmac`, `trademark`, `lcustom`, `xxxcustom`, and `xxxslides`
`memo.sty`	Memo style option
`mfr.sty`	Modifier to `memo.sty`
`mitthesis.sty`	Massachusetts Institute of Technology thesis format
`mitthesis-sample.tex`	
`natsci.bst`	Natural sciences generic BibTeX style
`natsci.sty`	Formats citations created with `natsci.bst`

newalpha.bst	Modified alphabetic BibTEX style
nl.sty	Style file customized for Dutch
nopagenumbers.doc	Remove page numbers
nopagenumbers.sty	
remark.sty	Like newtheorem but no \it
resume.sty	Format for doing resumes
resume-sample.tex	Sample file
rscsencode.shar	
sc21.sty	ISO/TC97/SC21 document style
sc21-wg1.sty	option for cover page
sfwmac.sty	Useful macros for Unix documentation
showlabels.sty	Shows labels and references to them
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
spacecites.sty	between citations
suthesis.doc	Stanford U thesis style
suthesis.sty	
texindex.shar	Style file and processor for index entries for VMS
texnames.doc	Define a couple more
texnames.sty	TEX names
tgrind.sty	Tgrind macros for LATEX instead of TEX
threepart.sty	Three part page headers
titlepage.txt	Style file in text format to go with article.txt
trademark.sty	Definitions of common trademarks
uct10.doc	U of California thesis style
uct11.doc	
uct12.doc	
ucthesis.doc	
ucthesis.readme	
vdm.doc	Vienna Development Method
vdm.sty	LATEX style
vdm.tex	
wsltex.shar	Wordstar to LATEX filter, C and Pascal versions
xxxcustom.tex	Supplementary macros for xxx-tex, for some xxx
xxxslides.sty	Supplementary macros for SLITEX, includes slides.sty

New entries since the last TUGboat listing are marked with an *. More submissions are very welcome. Send them to

Ken
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.

Sample FTP session for Internet users

```
% ftp cayuga.cs.rochester.edu   (a.k.a. cs.rochester.edu, 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
```

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` (`@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: (*don't forget your address*!)

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

@
user%site.bitnet@cunyvm.cuny.edu
00readme
00index
@
```

A word to the wise: it is best to fully qualify your mail address. Our mailer is pretty ignorant of Bitnet, CSnet or UUCP addresses unless they are in registered domains. It is best that you supply explicit gateway routes. Use the new domainized form or addresses whenever possible. Examples:

```
user%site.bitnet@cunyvm.cuny.edu
user%site.csnet@relay.cs.net
site!user@uunet.uu.net
```

Long UUCP paths are discouraged. System administrators get upset and your turnaround is very slow anyway.

If the `Subject:` line looks like:

```
Subject: @file request uuencode
```

or

```
Subject: @file request rscsencode
```

then the mail will be encoded with the requested scheme before sending. This *might* help sites that get mail through gateways with unfriendly EBCDIC/ASCII mappings. You can find sources for the two types of en/decoders in the collection. You may have to do some porting of sources.

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.

Distribution for IBM PC and clone users

There are two sources.

- David W. Hopper
  446 Main Street
  Toronto, Ontario
  Canada M4C 4Y2

has LaTeX style files only.

1. Either one 1.2 MB diskette or three 360KB diskettes, blank and formatted.
2. Indication of the format required,
3. A self-addressed mailer, and
4. A $5.00 donation per set of files, to cover postage and equipment wear & tear. (If you live outside North America, airmail delivery will probably require more postage. You should probably contact David for details.)
5. No phone calls or personal visits please.

- Jon Radel
  P. O. Box 2276
  Reston, VA 22090

has LaTeX style files and other goodies. For a list or other info send a SASE.

1. 360KB diskettes, blank and formatted.
2. A stamped, self-addressed mailer, and
3. $1.50 per disk. If you live outside North America, skip the stamps and send additional money or International Reply Coupons.

As a convenience for people who have more money than floppies, Jon will supply everything for $6.00 per disk to U.S./Canada/Mexico addresses.

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`

Additional volunteers should contact Ken.

The LaTeX User's Column

Jackie Damrau
University of New Mexico

Since the last column, I have received one question and two answers to questions that appeared in the last issue. Leslie Lamport has agreed to lend his fine knowledge in helping me to answer those questions that I cannot answer. Should you send a question, I will answer it as soon as possible via electronic mail and then publish the question and answer in the next issue of TUGboat.

A challenging question of my own is included. This question was given to me by one of my professors who said, "I am sure that LaTeX cannot do this." But after a little hard work on my part, I did manage to prove that LaTeX could do what he wanted.

Until the next TUGboat, happy LaTeXing.

Question 1

Jackie - I noticed in TUGboat that you were prepared to accept questions from beginners. Well I was wondering if you could suggest a way in LaTeX of allowing text to "wrap-round" a small square space (say 3 inches square) set flush right or flush left. The space would contain a line drawing or photograph. The macros given by Alan Hoenig in TUGboat Vol 8, No 2 would appear to do the job. Can I use these macros within a LaTeX document? Any help or suggestions would appreciated.

Ian Gibson
GUELPH2@WATDCS.BITNET

Answer

A quick glance at Hoenig's macros reveals no reason why they shouldn't work in LaTeX. However, users of such sophisticated macros should be aware that it is very difficult to make them robust, and there are bound to be LaTeX or Plain TeX commands that "break" when used with them. For example, I would not be surprised if errors resulted when a LaTeX `\footnote` command appears in one of the shaped paragraphs, or if Hoenig's macros are used inside a LaTeX 'figure' environment. So, my guess is that Hoenig's macros would work properly 95% of the time for a LaTeX user. A TeX hacker could probably figure out what to do the other 5% of the time; a naive user could be in trouble.

Even when the macros do work, they are not going to be easy to use; one will have to do page layout one page at a time, and changes to the document may require extensive manual reformatting. LaTeX was designed so that users don't have to worry about this kind of formatting; a user should think very hard about whether the advantage of this kind of figure placement is worth the hassle. I can think of no justification unless the user is producing camera-ready copy for a book–or perhaps for a journal article.

Answers to earlier questions

These answers to Questions 2 and 4 from the last TUGboat [Vol. 8 (1987), No. 3] were submitted by R. A. Bailey, Statistics Department, Rothamsted Experimental Station.

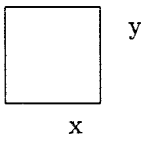
Answer (Question 2)

You have to be very careful with `verbatim`. After `\begin{verbatim}`, no other command is obeyed until `\end{verbatim}` is encountered. In particular, if `\begin{myenv}` is translated as `\begin{verbatim}`, then the `\end{myenv}` is processed as verbatim text, and so is not interpreted as `\end{verbatim}`. This is why `verbatim` must not appear in the argument of any command, including the `\newenvironment` command: see page 168 of the LaTeX manual.

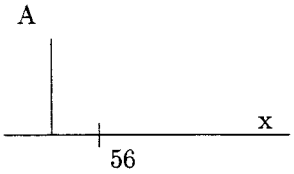
Answer (Question 4)

The `table` environment creates a box of exactly the right size, into which it puts the table contents and the caption. Unlike a page, this box has no predetermined height, so there is nothing for `\vfill` to stretch to. I can suggest only one method for achieving the requested result, and it is not very elegant: replace `\vfill` by a `\vskip` of a length calculated after a trial run. (I tried using the `picture` environment to put the table contents and the caption in the correct places in a box whose size is the same as the usual text, but you do not seem to be allowed to use `\caption` in this environment.)

Example



maximized area $= xy$
$224 =$ perimeter $= 2x + 2y$,
$y' = \dfrac{224 - 2x}{2} = 112 - x$
$A = xy = x(112 - x) = 112x^2$
$\dfrac{dA}{dx} = 112 - 2x = 0$ at $x = 56$
dim. for largest area are 56×56



Answer

```
\parbox[b]{1.5in}{
\setlength{\unitlength}{.25in}
\begin{picture}(5,5)
\put(1,1){\line(1,0){2}}
\put(1,3){\line(1,0){2}}
\put(1,1){\line(0,1){2}}
\put(3,1){\line(0,1){2}}
\put(2,0){\makebox(1,1){x}}
\put(3.25,2){\makebox(1,1){y}}
\end{picture}
} \quad
%
\parbox[b]{2.5in}{
maximized area $ = xy $ \\
$ 224 = $ perimeter $ = 2x+2y $, \\
$ y' = \dfrac{224-2x}{2} = 112-x $ \\
$ A = xy = x(112-x) = 112x^2 $ \\
$ \dfrac{dA}{dx} = 112-2x = 0 $ at $ x=56 $ \\
dim. for largest area are $ 56 \times 56 $} \
%
\parbox[b]{1.5in}{
\setlength{\unitlength}{.25in}
\begin{picture}(5,5)
\put(1,1){\line(1,0){6}}
\put(2,1){\line(0,1){2}}
\put(3,.75){\line(0,1){.5}}
\put(3,0){\makebox(1,1){56}}
\put(6,.75){\makebox(1,1){x}}
\put(1,3){\makebox(1,1){A}}
\end{picture}
}
```

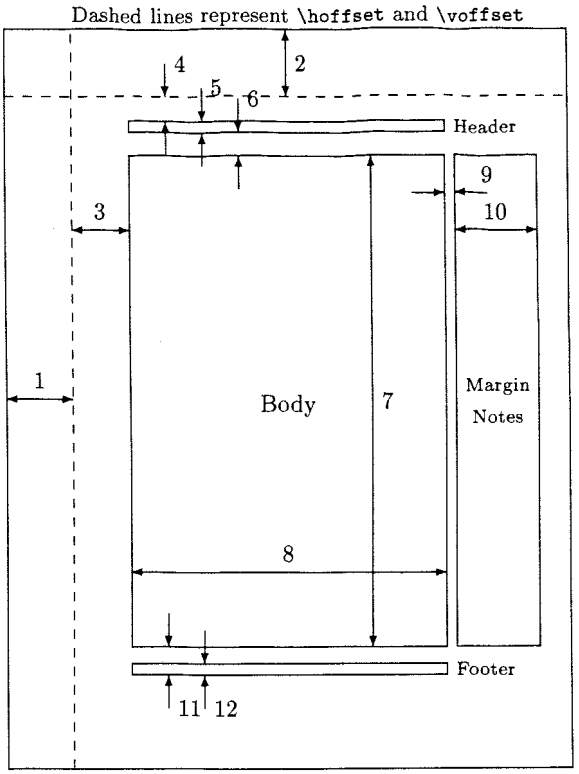
Page Layout in LaTeX

Kent McPherson
SLI Avionic Systems Corp.

One of the most frequently asked questions about LaTeX is "How can I change the layout of a page?" The answer is really not that difficult *if* one knows how the page is designed in the first place. Let me point out here that the author of LaTeX, Leslie Lamport, will be the first to point out that LaTeX is supposed to relieve the author of formatting concerns. However, there are cases where none of the styles defined for LaTeX will satisfy everyone's needs.

So, let's start with the basic layout of a page that is typeset using the `article` style in `10pt` type. See Figure 1. The first thing to note is that LaTeX assumes the page starts one inch down and one inch from the left as indicated by the dashed lines in Figure 1. The boxes that are identified as "Header," "Body," "Footer," and "Margin Notes" are where any text you write gets placed on the page. The issue, then, is to adjust the appropriate parameters so that the page layout is changed to the desired format.

Let's look at each of the page layout parameters individually. Again, refer to Figure 1.

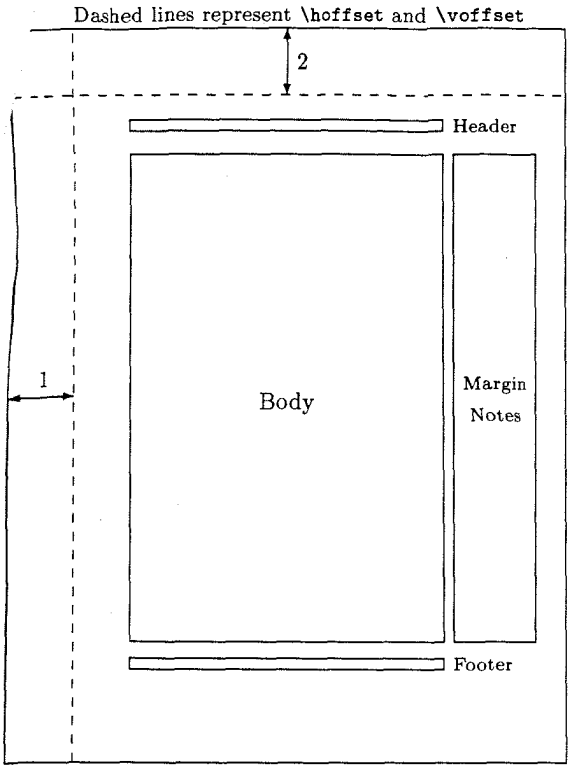



```
1 \hoffset = 0pt          2 \voffset = 0pt
3 \oddsidemargin = 63pt   4 \topmargin = 27pt
5 \headheight = 12pt      6 \headsep = 25pt
7 \textheight = 528pt     8 \textwidth = 345pt
9 \marginparsep = 11pt    10 \marginparwidth = 90pt
11 \footskip = 30pt       12 \footheight = 12pt
```

Figure 1: Sample LaTeX page layout

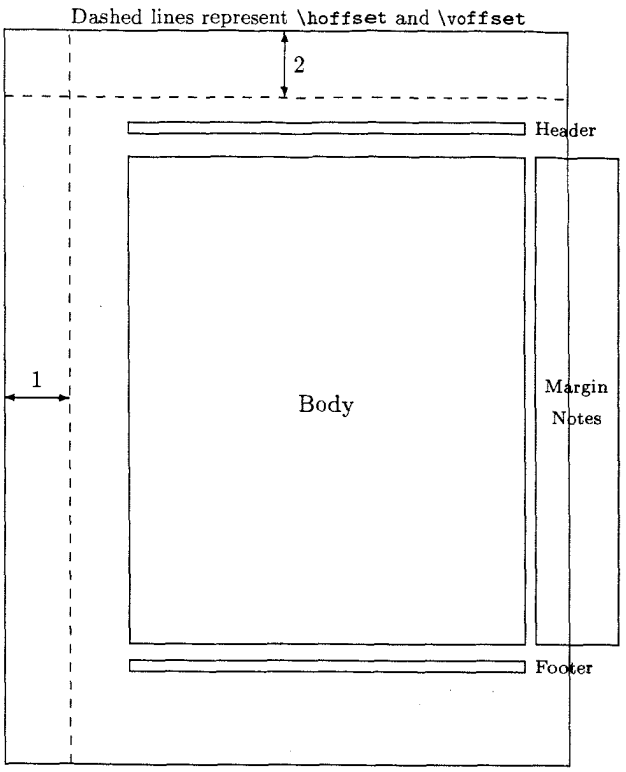
1. `\hoffset`: This is initially set to 0 points. This corresponds to a 1 inch horizontal offset.
2. `\voffset`: This is initially set to 0 points. This corresponds to a 1 inch vertical offset.
3. `\oddsidemargin`: This is the *additional* space that is added for the left margin, i.e. the true left margin is equal to `\oddsidemargin` plus one inch. This parameter can be negative. For example, if `\oddsidemargin` is set equal to `-.5in`, then the body will start $1/2$ inch to the left of the dashed line.
4. `\topmargin`: This is the *additional* space that is added for the top margin, i.e. the true top margin is equal to `\topmargin` plus one inch. This parameter can also be negative with the same relative effect as `\oddsidemargin`.
5. `\headheight`: This is the height of the box containing any header information.
6. `\headsep`: This is the distance between the header box and the body of the page.
7. `\textheight`: This is the height of the body of the page.
8. `\textwidth`: This is the width of the body of the page.
9. `\marginparsep`: This is the distance between the right edge of the body and the marginal notes box.
10. `\marginparwidth`: This is the width of the box containing marginal notes.
11. `\footskip`: This is the distance between the baseline of the last line in the body and the baseline of the footer box.
12. `\footheight`: This is the height of the box containing footer information.

With this in mind, I have designed a substyle option that will graphically show the layout of a page based on the page layout parameters discussed above.



1 \hoffset = 0pt 2 \voffset = 0pt
3 \oddsidemargin = 63pt 4 \topmargin = 27pt
5 \headheight = 12pt 6 \headsep = 25pt
7 \textheight = 528pt 8 \textwidth = 345pt
9 \marginparsep = 11pt 10 \marginparwidth = 90pt
11 \footskip = 30pt 12 \footheight = 12pt

Figure 2: Sample output from \layout command



1 \hoffset = 0pt 2 \voffset = 0pt
3 \oddsidemargin = 63pt 4 \topmargin = 27pt
5 \headheight = 12pt 6 \headsep = 25pt
7 \textheight = 528pt 8 \textwidth = 433pt
9 \marginparsep = 11pt 10 \marginparwidth = 90pt
11 \footskip = 30pt 12 \footheight = 12pt

Figure 3: Page layout with \textwidth increased to 6 inches

For example, if your test looked like

```
\documentstyle[layout]{article}
\begin{document}
\layout
\end{document}
```

you would get a page that looks like Figure 2.

Now, lets assume you want to make the body wider so that you can get more text printed per page. The obvious parameter to modify is \textwidth. The not-so-obvious parameters would be \oddsidemargin and possibly \marginparwidth. Let's say you want to make the body 6 inches wide with a 1 inch margin on both sides. If you set \textwidth to 6 inches without changing anything else, you would get what is shown in Figure 3. As you can see from Figure 3, the body has the same left margin and simply extends 6 inches to the right causing the margin notes box to be pushed partly off the page.

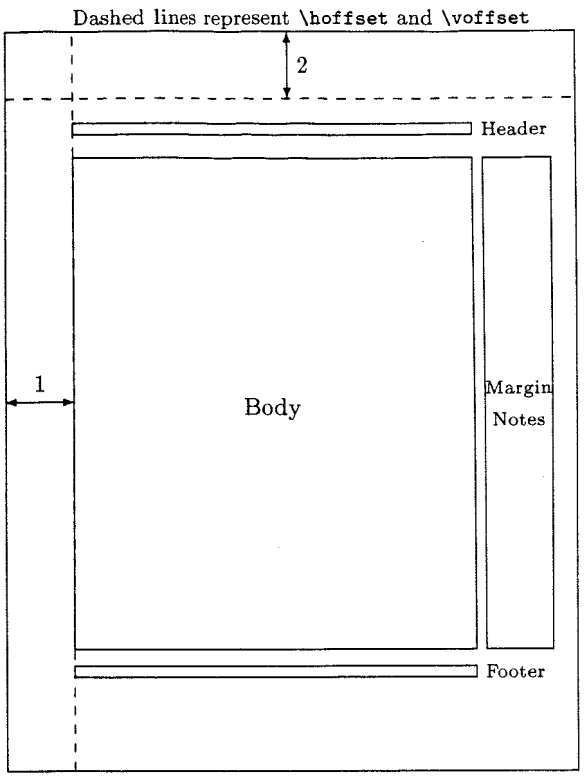
You can also reset \oddsidemargin and decrease the size of the margin notes box. For example (remembering that 1 inch $\approx$ 72pt), with

```
\documentstyle[layout]{article}
\setlength{\textwidth}{433pt}
\setlength{\oddsidemargin}{0pt}
\setlength{\marginparwidth}{72pt}
\begin{document}
\layout
\end{document}
```

you would get a page that looks like Figure 4.

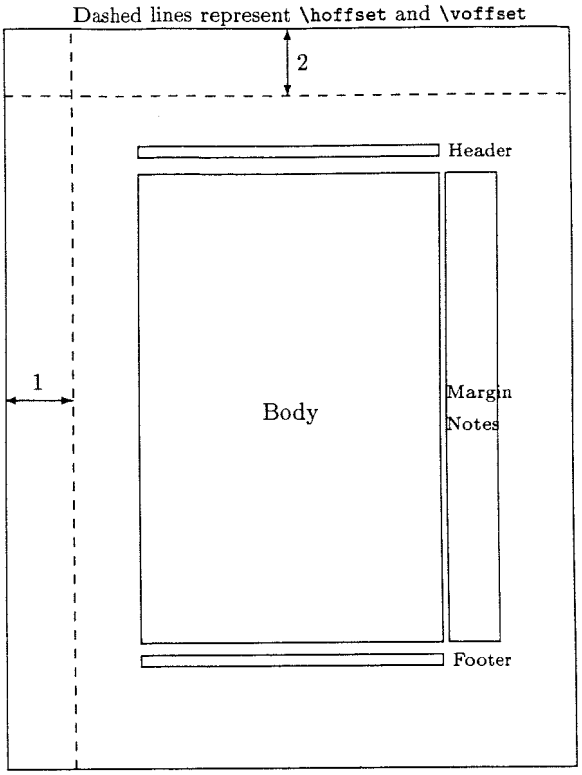
Another useful idea is to see the layout of existing LaTeX document styles. For example, the layout of the book style is shown in Figure 5.

In summary, when you want to change the layout of a page in LaTeX, remember the following:



1 \hoffset = 0pt 2 \voffset = 0pt
3 \oddsidemargin = 0pt 4 \topmargin = 27pt
5 \headheight = 12pt 6 \headsep = 25pt
7 \textheight = 528pt 8 \textwidth = 433pt
9 \marginparsep = 11pt 10 \marginparwidth = 72pt
11 \footskip = 30pt 12 \footheight = 12pt

Figure 4: Properly adjusted page layout



1 \hoffset = 0pt 2 \voffset = 0pt
3 \oddsidemargin = 74pt 4 \topmargin = 54pt
5 \headheight = 12pt 6 \headsep = 18pt
7 \textheight = 504pt 8 \textwidth = 325pt
9 \marginparsep = 7pt 10 \marginparwidth = 54pt
11 \footskip = 25pt 12 \footheight = 12pt

Figure 5: Page layout of 10-point book style.

- First, do you really need to change the layout? After all, if it is simply a matter of trying to make something look "prettier," I would say don't do it.
- Secondly, if it is necessary to change the page layout, remember to place all \setlength commands *prior* to the \begin{document} command.
- Thirdly, don't forget to adjust the not-so-obvious parameters.
- Lastly, when in doubt, use the \layout command to display the layout of a page as shown in the examples.

The following file must be placed in the TEX$INPUTS directory.

LAYOUT.STY

```
%
% This file should be called LAYOUT.STY
% and should be placed in the TEX_INPUTS
% directory.
%
% Define \bs if it is undefined, redefine
% it if it is already defined.
%
\@ifundefined{bs}{\newcommand\bs{\char '134 }}%
  {\renewcommand\bs{\char '134 }}
\@ifundefined{lb}{\newcommand\lb{\char '173 }}%
  {\renewcommand\lb{\char '173 }}
\@ifundefined{rb}{\newcommand\rb{\char '175 }}%
  {\renewcommand\rb{\char '175 }}
\newcount\hofset
\newcount\vofset
\newcount\hofref
\newcount\vofref
```

```
\newcount\omargin
\newcount\omarginref
\newcount\emargin
\newcount\emarginref
\newcount\marginref
\newcount\tmargin
\newcount\hheight
\newcount\hsep
\newcount\theight
\newcount\twidth
\newcount\mparsep
\newcount\mparwidth
\newcount\fskip
\newcount\fheight
\newcount\headref
\newcount\bodyref
\newcount\footref
\newcount\margnoteref
\newcount\oneinch
\newcount\eighthalfinch
\newcount\teninch
\newcount\eleveninch
%
% Constants
%
\oneinch=72
\eighthalfinch=615
\teninch=723
\eleveninch=795
%
% Define the calculations macro
%
\def\layout{
%
% Convert dimensions to scalar values
% for use in the picture environment
%
\hofset=\hoffset
  \divide\hofset by 65536
\hofref=\hofset
\advance\hofset by \oneinch
\vofset=\voffset
  \divide\vofset by 65536
\vofref=\vofset
\vofset=\teninch
  \advance\vofset by -\vofref
%
\tmargin=\topmargin
  \divide\tmargin   by 65536
%
\hheight=\headheight
  \divide\hheight   by 65536
\headref=\teninch
  \advance\headref by -\vofref
  \advance\headref by -\tmargin
  \advance\headref by -\hheight
%
\hsep=\headsep
  \divide\hsep      by 65536
%
\theight=\textheight
  \divide\theight   by 65536
\bodyref=\headref
  \advance\bodyref by -\hsep
  \advance\bodyref by -\theight
%
\fskip=\footskip
  \divide\fskip     by 65536
%
\fheight=\footheight
  \divide\fheight   by 65536
\footref=\bodyref
  \advance\footref by -\fskip
%
\omargin=\oddsidemargin
  \divide\omargin   by 65536
%\omarginref=\omargin
  \advance\omarginref by \oneinch
%\advance\omarginref by \hofref
%
\emargin=\evensidemargin
  \divide\emargin   by 65536
%\emarginref=\emargin
  \advance\emarginref by \oneinch
%\advance\emarginref by \hofset
%
\twidth=\textwidth
  \divide\twidth    by 65536
%
\mparsep=\marginparsep
  \divide\mparsep   by 65536
%
\mparwidth=\marginparwidth
  \divide\mparwidth by 65536
%
\if@twoside
    \ifodd\count\z@
%
% Twosided, odd page
%
  \typeout{Two-sided document
    style, odd page.}
  \margnoteref=\oneinch
    \advance\margnoteref by \hofref
  \advance\margnoteref by \omargin
```

```
  \marginref\margnoteref
  \advance\margnoteref by \twidth
  \advance\margnoteref by \mparsep

    \else
%
% Twosided, even page
%
  \typeout{Two-sided document
    style, even page.}
  \margnoteref=\oneinch
    \advance\margnoteref by \hofref
  \advance\margnoteref by \emargin
  \marginref\margnoteref
  \advance\margnoteref by -\mparsep
  \advance\margnoteref by -\mparwidth

    \fi
\else
%
% Not twosided, do odd page
%
\typeout{One-sided document style.}
\margnoteref=\oneinch
  \advance\margnoteref by \hofref
\advance\margnoteref by \omargin
\marginref\margnoteref
\advance\margnoteref by \twidth
\advance\margnoteref by \mparsep
\fi
Dashed lines represent
{\tt \bs hoffset} and
{\tt \bs voffset}.
\medskip
%
% Define the picture to be drawn
%
\setlength{\unitlength}{.5pt}
\begin{picture}(\eighthalfinch,\eleveninch)
\centering
\thicklines
%
% Page box and reference lines
%
\put(0,0){\framebox(\eighthalfinch,
  \eleveninch){\mbox{}}}
\put(0,\vofset){\dashbox{10}
  (\eighthalfinch,0){\mbox{}}}
\put(\hofset,0){\dashbox{10}(0,
  \eleveninch){\mbox{}}}
%
% Header
%
\put(\marginref,\headref){\framebox
  (\twidth,\hheight){\footnotesize Header}}
%
% Body
%
\put(\marginref,\bodyref){\framebox
  (\twidth,\theight){Body}}
%
% Footer
%
\put(\marginref,\footref){\framebox
(\twidth,\fheight){\footnotesize Footer}}
%
% Marginal notes
%
\put(\margnoteref,\bodyref){\framebox
  (\mparwidth,\theight)%
{\footnotesize
\shortstack{Margin\\Notes}}}
\end{picture}

\medskip
%
% Display the settings used to make
% the picture.   Note: fractional
% points are truncated, i.e.,
% 72.27pt is displayed as 72pt
%
{\tt
\begin{tabular}{l@{\hspace{20pt}}l}
\bs hoffset = \number\hofref pt &
\bs voffset = \number\vofref pt \\
\bs
   \if@twoside
     \ifodd\count\z@ oddsidemargin
     \else evensidemargin
     \fi
   \else oddsidemargin
   \fi
= \number
   \if@twoside
     \ifodd\count\z@ \omargin
     \else \emargin
     \fi
   \else \omargin
   \fi
pt & \bs topmargin = \number\tmargin pt \\
\bs headheight = \number\hheight pt &
  \bs headsep = \number\hsep pt \\
\bs textheight = \number\theight pt &
  \bs textwidth = \number\twidth pt \\
\bs marginparsep = \number\mparsep pt &
\bs footskip = \number\fskip pt &
\bs footheight = \number\fheight pt \\ [10pt]
\multicolumn{2}{c}{72pt $\approx$ 1 inch}
\end{tabular}}
}  % end of \def\layout
```

Queries

Automatic Page Balancing Macros Wanted

One of the shortcomings of TEX (all right, all right, the only shortcoming) is its inability to handle page makeup simply. If one wishes (as one should) to find the most elegant pagebreaks, one must go into one's file and manually insert \breaks and {\looseness=1\tolerance=2000...\par}s or add lines of the form

```
\ifnum\pageno=68 \global
   \advance\vsize by-1\baselineskip\fi
\ifnum\pageno=72 \global
   \advance\vsize by1\baselineskip\fi
```

to the end of \plainoutput.

Altering the \vsize by one line in either direction (but not more) is a standard trick of good typesetting, but both pages of a spread must be the same size.

If one is unfortunate enough to have an article or chapter that has long unbreakable displays and more insertions than pages, one can end up spending altogether too much time determining pagebreaks. While, admittedly, it may be perfectly legitimate to expect an author to struggle with his file over half a dozen or so runs to make it perfect, a typesetter's time is much too valuable to indulge in such foolishness.

I had thought that it would be a simple matter to alter \plainoutput to {\ifodd\pageno \oddpageout \else \evenpageout \fi}, where \evenpageout would define a \vbox called \evenpagebox while the analogous \oddpagebox was created. Once one has the two boxes, one would add the badnesses and, if the sum is greater than, say, 2000, lengthen or shorten the \vsize by 1 \baselineskip and \unvbox both boxes.

What a simple idea, I thought. All I was worried about at first was insertions: I had no very clear idea how they would go back on the list of recent contributions. Then I discovered that there was *no way* for TEX to tell me how bad a \vbox was before it was shipped out!

Is there a Grandmaster or Wizard out there who can show me how to discover the badness of a box before it's too late?

Frederick H. Bartlett
The Bartlett Press, Inc.

Inverted Pyramidal Titles

Stephen C. Lipp's request (**8**(3): 326) for a title macro that (i) requires only spaces between words, (ii) capitalizes, (iii) double-spaces, (iv) centers, (v) has an inverted pyramidal shape, (vi) preferentially breaks at commas, and (vii) smoothly varies line length is hereby granted:

```
\newcount\initiallinesdone
\newcount\finallinesdone
\newcount\endhere
{\catcode`\,=\active
\gdef\invpyramid#1#2{{\catcode`\,=\active
    \def,{\char"2C\penalty-5000\ }%
  \multiply\normalbaselineskip by2
    \normalbaselines
  \parfillskip=0pt\parindent=0pt
  \leftskip=0pt plus9pt minus9pt
    \rightskip=\leftskip
  \endhere=0\initiallinesdone=#2
  \loop
    \everypar={\prevgraf=\initiallinesdone}
    \global\setbox0=\vbox{
    \parshape=12 1pc25pc 2pc23pc 3pc21pc
      4pc19pc 5pc17pc 6pc15pc 7pc13pc
      8pc11pc 9pc9pc 10pc7pc 11pc5pc
      12pc3pc
    \noindent\uppercase{#1}\endgraf
    \global\finallinesdone=\prevgraf}
    \ifnum\endhere=1
      \global\advance\endhere by1
    \else
      \ifnum\finallinesdone>12
        \global\advance\initiallinesdone
        by-2\global\finallinesdone=11
        \global\advance\endhere by1
      \fi
      \ifnum\finallinesdone<12
        \global\advance\initiallinesdone
          by1
      \else
        \global\advance\endhere by2
      \fi
    \fi
  \ifnum\endhere<2
  \repeat
  \box0}}}
```

Example (I fudged the example with a \kern -4.16667pc since this column is $8\frac{1}{3}$ picas narrower than the measure assumed for \invpyramid):

```
\invpyramid{This is a long, ses\-qui\-
  pe\-da\-lian, verbose title}7
```

THIS IS A LONG, SES-

QUIPEDALIAN,

VERBOSE

TITLE

The second group reflects the user's guess of how many lines the title should take up. I thought that this one would take four, so I set `\initiallinesdones` to 7. If the guess is too high, the result may be rather ugly, so use 11 − ⟨guess⟩ instead of 12 − ⟨guess⟩. (This is not strictly necessary, but it speeds the process up at a small cost in human thought.)

The `\parshape` is not variable; that is, it must be determined afresh for each `\hsize` (unless the user wishes to use a variable dimension, as Knuth did in his answer to Exercise 14.18 on p. 315 of *The TEXbook*, and further complicate the `\looping`).

If the user wishes to break at commas even more often, he could increase the `\penalty` to `-9999`.

If the user wishes to discourage (or encourage) hyphenation, he could give `\pretolerance`, `\exhyphenpenalty`, and `\hyphenpenalty` new values.

If the user wishes to allow more variation from a perfect pyramid, he could increase the `\leftskip` and `\rightskip`.

The subtle part of this macro is the `\loop` and its use of `\prevgraf`.

Without the `\loop`, it is obvious that only titles that are exactly 12 lines long will have appropriately narrow last lines. So, at the end of the first pass, TEX checks `\finallinesdone`. If `\finallinesdone` is 12, we stop and set the `\vbox`. If (as is more likely) `\finallinesdone` is less than 12, we add 1 to `\initiallinesdone`, which fools `\parshape` into setting the first line shorter, and try again.

If `\finallinesdone` is greater than 12, 1 is deducted from `\initiallinesdone` and 1 is added to `\endhere`. Being interpreted, this means, "There's no way to make the last line 3 picas wide, so I'm going to do the next best thing."

It is assumed that the user's title is less than approximately 168 picas long. If it is longer, then the `\parshape` will have to be respecified for more than 12 lines. The attentive reader will note that an error (and a terrible result) would ensue if this condition is not met, for that would ensure that the very first pass would produce a `\finallinesdone` greater than 12, which would cause `\initiallinesdone` to be reduced to −1, which TEX will not allow. Thus, instead of 12 lines, you'd get about 20, the last 9 of which would all be 3 picas wide. In ten-point roman, however, 168 picas is about forty-five words, which is too long for any reasonable title anyway.

Frederick H. Bartlett
The Bartlett Press, Inc.

Logarithmic Time Scales

I should like to make a first (approximate) stab at responding to James Alexander's request in (**8**(2): 216) for a time scale macro.

I should admit immediately that there is a lot of grunt work to be done: I haven't (1) provided the necessary code to read dates and events from a separate file, (2) given a method for producing a linear time scale, (3) provided a method for determining the length, starting point, and finishing point of the scale, (4) met Alexander's specifications for typing the entries, or (5) addressed the problem of clustered entries.

Items (1) and (2) seem to me quite straightforward; I'm just too lazy to complete them. Item (3) is slightly more difficult: if you absolutely must have only two parameters to the `\entry` macro, you could have TEX determine the value of what I call `\exponent` by dividing the length of a `\hbox` containing the date string by the width of a number in the current font. A new command, `\parse`, say, could then determine `\integer` and `\decimal` (`\def\parse#1#2//{\integer=#1\decimal=#2}`).

Item (4) is more difficult still: you must either have the user specify the three ⟨dimen⟩s or run TEX on the file twice: once to get the logarithms of the first and last dates and once to set the scale (if the length of the scale is not equal to `\vsize`, the user would have to specify it—unless you want TEX to determine an optimum scale length).

Item (5) is a real bear. I don't see how TEX could remember an arbitrary number of dates to see if they are "too close" to one another (if it can, then TEX could also give the optimum scale length). Given a presorted list, however, it might be possible. The solution seems simple for a pair of close entries; if there are three or more close entries, though, I don't see an immediate solution (besides increasing the scale length).

What attracted me to this query was the challenge of coercing TEX into doing logarithms. In solving this puzzle, I discovered (and if this is documented in *The TEXbook*, I, at least, couldn't find it) that, while TEX will add, subtract, and divide in the range ±2147483647, it will multiply only in the range ±1073741823.

Thus, my approximation algorithm was limited by $\max(\log(n+1) - \log n) \times \max(\backslash\text{ddecimal}) = 1073741823$.

```
\newcount\clogi         \clogi=-32767
\newcount\clogii        \clogii=0
\newcount\clogiii       \clogiii=19167
\newcount\clogiv        \clogiv=32767
\newcount\clogv         \clogv=43316
\newcount\clogvi        \clogvi=51934
\newcount\clogvii       \clogvii=59222
\newcount\clogviii      \clogviii=65534
\newcount\clogix        \clogix=71102
\newcount\clogx         \clogx=76083
\newdimen\alog          \newcount\integer
\newdimen\decimal       \newcount\ddecimal
\newcount\exponent      \newcount\base
\newcount\basea         \newcount\tare
\tare=32768
\def\entry#1.#2E#3#4{\integer=#1
  \decimal=.#2pt\exponent=#3
  \findlog
  \vskip\alog
  \vbox to0pt{\vss\line{\vrule height
    3.59267pt depth-3.35177pt width.3in
    \quad#4\hfil}\vss}
  \vskip-\alog\vskip-\baselineskip}
\def\findlog{%
  \ifnum\integer=0
   \base=0\basea=\clogi
  \else
   \base=\csname clog\romannumeral\integer
      \endcsname{\advance\integer by1
      \global\basea=\csname clog%
      \romannumeral\integer\endcsname}%
  \fi
  \advance\basea by-\base
  \advance\decimal by-.5pt
  \ddecimal=\decimal
  \multiply\ddecimal by\basea
  \multiply\basea by\tare
  \advance\ddecimal by\basea
  \divide\ddecimal by108850
  \advance\base by32767
  \multiply\base by10337
  \divide\base by17169
  \multiply\exponent by65536
  \alog=\ddecimal sp
  \advance\alog by\base sp
  \advance\alog by\exponent sp
  \multiply\alog by50}
```

This is only a rough-and-ready version; I wanted to get the basic ideas down for others to use and improve (especially since poor Prof. Alexander has been waiting for this for some months!). I suspect that a mathematician (or a more skilled TEXpert, or both) could squeeze some more accurate logarithms out of TEX; at a scaling factor of 50, this is only accurate to within four dots, or .9636 pts, on my laser printer.

Frederick H. Bartlett
The Bartlett Press, Inc.

Calendar

1988

Vanderbilt University, Nashville, Tennessee

Mar 7–11 Beginning TEX
Mar 7–11 Intensive Beginning/Intermed. TEX

Northeastern University, Boston, Massachusetts

Mar 21–25 Beginning TEX
Mar 21–25 Intensive Beginning/Intermed. TEX

University of Illinois, Chicago, Illinois

Mar 21–25 Beginning TEX
Mar 21–25 Intermediate TEX
Mar 21–25 Advanced TEX/Macro Writing

Apr 4 **TUG:** Paper selection for **1988 Annual Meeting**; notification sent to speakers.
Apr 20–22 International Conference on Electronic Publishing, Document Manipulation and Typography, Nice, France (see announcement, TUGboat Vol. 8, No. 1, page 78)
Apr 26 GUTenberg (Groupe des Utilisateurs TEX), Paris, France (for information, contact B. Gaulle, UCIR001%FRORS31.BITNET)

Carleton University, Ottawa, Ont., Canada

Apr 25–29 Beginning TEX
Apr 25–29 Intensive Beginning/Intermed. TEX
Apr 25–29 Intensive Course in LATEX
May 2–6 Advanced TEX/Macro Writing
May 2–6 LATEX Style Files

May 9–12 TEX Wizard Course, TV Guide Office, Radnor, Pennsylvania
May 16 **TUGboat Volume 9, No. 2:** Deadline for receipt of manuscripts

University of New Mexico, Albuquerque

May 16–20 Advanced TEX/Macro Writing
May 23–27 Beginning TEX
May 23–27 Intensive Beginning/Intermed. TEX

Jun 6–8 Expert Communication 88: Artificial Intelligence in Electronic Publishing; San Jose, Calif. For information, contact Marion Elledge, Graphic Communications Association, Arlington, Va., (703) 841-8160
Jun 20 **TUG: Deadline** for camera copy of papers to appear in Proceedings of **1988 Annual Meeting**
Jul 18–20 TEX88 Conference, University of Exeter, England. To be placed on the mailing list, contact Cathy Booth (Janet: booth.cm@uk.ac.exeter) or Malcolm Clark (Janet: texline@uk.ac.ic.cc.vaxa)
Aug 1–5 ACM SIGGRAPH; Atlanta, Ga. For information, call (312) 644-6610

TEX Users Group 1988 Conference McGill University, Montréal, Québec

Aug 15–19 Intensive Beginning/Intermed. TEX
Aug 16–19 Short Course in METAFONT
Aug 22–24 **TUG Annual Meeting** See announcement, page 87.
Aug 25–26 Short Course (topic to be announced)

Sep 12 **TUGboat Volume 9, No. 3:** Deadline for receipt of manuscripts (tentative).

For additional information on the events listed above, contact the TUG office (401-272-9500, ext. 232) unless otherwise noted.

Status as of 25 February 1988

TEX Users Group 1988 Annual Meeting

Program Coordinator
Dean Guenther
Washington State University

The 1988 TEX Users Group meeting will be held at McGill University in Montréal, Québec, Canada, from August 22nd to the 24th. There will be courses before and after the meeting (to be announced). Expect a separate announcement and registration forms later this Spring. The focus of this year's meeting will be on TEX in a production environment. Judging by the amount of interest so far, it looks like we will have the largest number of papers ever presented at an annual meeting. A preliminary list of topics includes:

- TEX previewers
- Book production with TEX
- Implementing TEX in a production environment
- Producing families of manuals from the same sources
- TEX and the Macintosh: A clash of cultures
- SGML and TEX
- The art of teaching TEX for production
- Mathematics textbook publishing with Japanese TEX
- TEX extension for Semitic languages

Videotapes of Knuth's Software Design Course Based on TEX: The Program

During Stanford's Spring Quarter, 1987, Donald Knuth presented a special course, TEX: The Program: A case study in software design. This course consisted of nineteen 60–75-minute lectures, which were transmitted to remote locations and videotaped by the Stanford Instructional Television Network (SITN).

The implementation of TEX was discussed as an example of the design and documentation of a medium-size software system. Also discussed was the `WEB` system of structured documentation. With knowledge of the TEX and Pascal languages as a prerequisite, enough information about the innards of TEX was presented for students to learn to make extensions to the system. The text for the course was *TEX: The Program.*

Videotapes available for rent

TUG now has these lectures available on videotape and has permission to rent them to TUG members, although SITN has stipulated that TUG may not rent them to profit-making organizations. (Commercial organizations should contact SITN directly at Stanford University, Stanford, CA 94305.)

The videotapes are available in VHS format; it may be possible to make them available also in Beta format if there is sufficient demand. Due to anticipated scheduling requirements, the *normal* rental period will be six weeks. The rental fee for the 19 videotapes for that duration will be $800. (A longer rental period may be arranged and the rental fee will be prorated.) This includes all costs except shipping/handling/insurance charges for their return to TUG. 1988 TUG Institutional Members are entitled to a 20% discount. Professor Knuth's book, *TEX: The Program*, the text for the course, may also be ordered from TUG.

If you feel this course might be useful to your organization, please contact Alan Wittbecker at the TUG office: 401-272-9500, ext. 232.

Institutional Members

Addison-Wesley Publishing Company, *Reading, Massachusetts*

The Aerospace Corporation, *El Segundo, California*

American Mathematical Society, *Providence, Rhode Island*

ArborText, Inc., *Ann Arbor, Michigan*

ASCII Corporation, *Tokyo, Japan*

Aston University, *Birmingham, England*

Brookhaven National Laboratory, *Upton, New York*

California Institute of Technology, *Pasadena, California*

Calvin College, *Grand Rapids, Michigan*

Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*

City University of New York, *New York, New York*

College of St. Thomas, Computing Center, *St. Paul, Minnesota*

College of William & Mary, Department of Computer Science, *Williamsburg, Virginia*

Columbia University, Center for Computing Activities, *New York, New York*

COS Information, *Montreal, P. Q., Canada*

Data General Corporation, *Westboro, Massachusetts*

DECUS, L&T Special Interest Group, *Marlboro, Massachusetts*

Department of National Defence, *Ottawa, Ontario, Canada*

Digital Equipment Corporation, *Nashua, New Hampshire*

dit Company, Ltd., *Tokyo, Japan*

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*

General Motors Research Laboratories, *Warren, Michigan*

Geophysical Company of Norway A/S, *Stavanger, Norway*

Grinnell College, Computer Services, *Grinnell, Iowa*

Grumman Corporation, *Bethpage, New York*

GTE Laboratories, *Waltham, Massachusetts*

Hart Information Systems, *Austin, Texas*

Hartford Graduate Center, *Hartford, Connecticut*

Harvard University, Computer Services, *Cambridge, Massachusetts*

Hewlett-Packard Co., *Boise, Idaho*

Hobart & William Smith Colleges, *Geneva, New York*

Humboldt State University, *Arcata, California*

Hutchinson Community College, *Hutchinson, Kansas*

IBM Corporation, Scientific Center, *Palo Alto, California*

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*

Louisiana State University, *Baton Rouge, Louisiana*

Marquette University, Department of Mathematics, Statistics, and Computer Science, *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*

National Center for Atmospheric Research, *Boulder, Colorado*

National Institutes of Health, *Bethesda, Maryland*

National Research Council Canada, Computation Centre, *Ottawa, Ontario, Canada*

New Jersey Institute of Technology, *Newark, New Jersey*

New York University, Academic Computing Facility, *New York, New York*

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 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, Department of Computer Science, *Stony Brook, New York*

Stratus Computer, Inc., *Marlboro, Massachusetts*

Syracuse University, *Syracuse, New York*

Talaris Systems, Inc., *San Diego, California*

Texas A & M University, Computing Services Center, *College Station, Texas*

Texas A & M University, Department of Computer Science, *College Station, Texas*

TRW, Inc., *Redondo Beach, California*

Tufts University, *Medford, Massachusetts*

TV Guide, *Radnor, Pennsylvania*

TYX Corporation, *Reston, Virginia*

UNI.C, Danmarks EDB-Center, *Aarhus, Denmark*

University College, *Cork, Ireland*

University of Alabama, *Tuscaloosa, Alabama*

University of British Columbia, Computing Centre, *Vancouver, British Columbia, Canada*

University of British Columbia, Mathematics Department, *Vancouver, British Columbia, Canada*

University of Calgary, *Calgary, Alberta, Canada*

University of California, Berkeley, Academic Computing Services, *Berkeley, California*

University of California, Berkeley, Computer Science Division, *Berkeley, California*

University of California, Irvine, Department of Mathematics, *Irvine, California*

University of California, Irvine, Information & Computer Science, *Irvine, 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 Crete, Institute of Computer Science, Research Center, *Heraklio, Crete, Greece*

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 Vermont, *Burlington, Vermont*

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*

Vereinigte Aluminium-Werke AG, *Bonn, Federal Republic of Germany*

Villanova University, *Villanova, Pennsylvania*

Vrije Universiteit, *Amsterdam, The Netherlands*

Washington State University, *Pullman, Washington*

Widener University, Computing Services, *Chester, Pennsylvania*

John Wiley & Sons, Incorporated, *New York, New York*

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
	1988 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $30.00 each Renewal: [ ] $40.00; [ ] $30.00 – reduced rate if renewed before February 1, 1988	
	1988 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 February 1, 1988	
	TUGboat back issues, 1980 (v. 1) 1981 (v. 2) 1982 (v. 3) 1983 (v. 4) 1984 (v. 5) 1985 (v. 6) 1986 (v. 7) 1987 (v. 8) $15.00 per issue (19), #1 #1,2,3 #1,2 #1,2 #1,2 #1,2,3 #1,2,3 #1,2,3 circle issue(s) desired:	

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.

Each Institutional Member is entitled to:

- designate up to 7, 12 or 30 individuals to receive TUGboat subscriptions, depending on category of membership chosen; named individuals will be accorded full status as individual TUG members;
- reduced rates for TUG meetings/courses for *all* staff members, and for rental/purchase of videotapes;
- be acknowledged in every issue of TUGboat published during the membership year.

Instructions: Attach a list of the names and addresses of individuals to whom you would like TUGboat subscriptions mailed, to include answers to the questions on both side of this form–as approrpiate, in particular those regarding the status of TEX and the computer(s)/operating system(s) on which it runs or is being installed. (For IBM and VAX, especially, the operating system is more relevant than model.) It would be particularly useful if you could provide this information as it relates to each individual or group using the same hardware. Please make as many copies of this form as needed or contact the TUG office for additional copies.

- *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.

Institution/Organization: _____

_____

_____

_____

Principal contact: _____

_____

_____

_____ Phone: _____

QTY	1988 INSTITUTIONAL MEMBERSHIP (JAN.-DEC.)									AMOUNT
	Category A (incl. 7 subs.): educational $295; non-ed. $395; add'l subs. $30/ea.									
	Category B (incl. 12 subs.): educational $425; non-ed. $525; add'l subs. $25/ea.									
	Category C (incl. 30 subs.): educational $795; non-ed. $895; add'l subs. $20/ea.									
	TUGboat back issues, $15.00 per issue (19), 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	1987 (v. 8) #1, 2, 3	

Air mail postage is included in the rates for all memberships outside North America.

TOTAL ENCLOSED: _____
(Prepayment in U.S. dollars required)

Membership List Information

Institution:

Principal contact:
Phone:
Specific applications or reason for interest in TEX:

This 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)

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, IBM MVS 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 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 tape.

The price of the tapes 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 may be put on a single 2400' foot reel but you will be charged for 2 tapes. 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 am and 2:30 pm 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

TEX Distribution tapes
____ ASCII generic format
____ EBCDIC generic format
____ VAX/VMS Backup format
____ DEC 20/TOPS 20 Dumper format
____ IBM VM/CMS format
____ IBM MVS format

Font Library Tapes (GF files)
____ 300 dpi VAX/VMS format
____ 300 dpi generic format
____ 200/240 dpi generic format
____ IBM 3800 CMS format
____ IBM 4250 CMS format
____ IBM 3820/3812 CMS format

Tape prices: $92.00 for first tape, $72.00 for each additional tape
Total number of tapes ____ (for postage allow 2 lbs for each tape)

Documents:

	Price $	Weight	Quantity
TEXbook (vol. A) softcover	25.00	2	____
TEX: The Program (vol. B) hardcover	37.00	4	____
METAFONT book (vol. C) softcover	22.00	2	____
METAFONT the Program (vol. D) hardcover	37.00	4	____
Computer Modern Typefaces (vol. E) hardcover	37.00	4	____
LATEX document preparation system	25.00	2	____
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	____

* 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, which is free, 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: _____ Telephone: _____

The TEXniques Series

Through this series of publications on TEX and the TEX environment, the TEX Users Group hopes to provide useful documentation to the TEX community. The first four volumes of the series are:

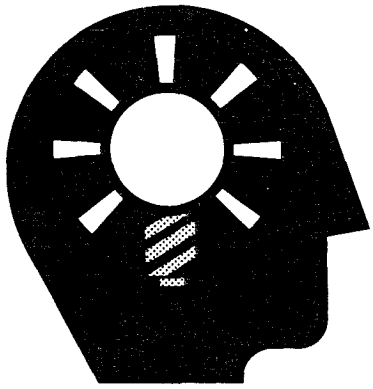
1. VAX Language-Sensitive Editor (LSEDIT) Quick Reference Guide for use with the LATEX Environment and LATEX Style Templates, by Kent McPherson.
2. Table-Making with INRSTEX, by Michael J. Ferguson.
3. User's Guide to the IdxTEX Program, by R. L. Aurbach.
4. User's Guide to the GloTEX Program, by R. L. Aurbach.

The *Proceedings of the Eighth TEX Users Group Annual Meeting* (University of Washington, Seattle, August 24–26, 1987), has been published as Number 5 of the *TEXniques* Series, with Dean Guenther as Editor. The principal theme of this meeting was "TEX in the Humanities". These papers are included in the volume.

- Bart Childs
  We've come a long way, and ?
- Christina Thiele
  TEX, linguistics and journal production
- Silvio Levy
  Typesetting Greek
- Walter Andrews and Pierre MacKay
  The Ottoman texts project
- Nobuo Saito and Kazuhiro Kitagawa
  What should we do for Japanese TEX
- Yasuki Saito
  Japanese TEX: [jTEX]
- Robert McGaffey
  Developing TEX DVI driver standards
- Nelson Beebe
  A TEX DVI driver family
- David Ness
  The use of TEX in a commercial environment
- Silvio Levy
  Literate programming in C
- Rick Simpson
  Porting TEX to the IBM RT
- Allen Dyer
  Text formatting and the Maryland lawyer
- Leslie Carr
  Of METAFONT and PostScript

For more information on acquiring the volumes in this series and other publications available from the TEX Users Group contact the TUG office.

IDEAS TRANSLATE



LIKE YOU IMAGINE

TYPESET books & MANUALS, FORMULAS & TABLES,
in the same FORMAT or dIfFerEnT ONES
BIG or SMALL WIDE • OR • NARROW LIGHT or BOLD.
IT'S NICE TO KNOW pcTEX WON'T LIMIT YOUR IMAGINATION.

WITH

Great ideas should look great on paper. And the translation is easy with pcTEX®: the best-selling full implementation of Prof. D. Knuth's revolutionary TEX formatting/typesetting program. It offers PC users the capabilites & advantages—and looks—of professional typesetting.

In a word, pcTEX gives you 'control'. Control—of design format, type & symbols, quality—for complex mathematical & engineering material, statistical tables or straight matter. You get camera/publisher-ready manuscripts to be proud of, quick & simple.

So whether you're writing the next starshot manual, a thesis on relativity or the great American novel, for a professional presentation that doesn't lose your ideas in the translation, depend on pcTEX.

From Personal TEX, Inc., starting at $249; VISA/MC welcome. Satisfaction guaranteed.

PERSONAL TEX INC

To order or for information, call:
415/388-8853
or write: Personal TEX, Inc.
12 Madrona Avenue
Mill Valley, CA 94941 USA

Distributed in 14 countries; distributor inquiries welcome.

pcTEX FORMATTING/TYPESETTING SYSTEM:

FINE TYPESET QUALITY from dot matrix or laser printers, or phototypesetters.

A COMPLETE PRODUCT. Includes • our specially written pcTEX Manual that lets you use TEX immediately • custom 'macro package' formats for letters, manuals, technical documents, etc. • the LaTEX document preparation system (with user's manual) macro package for article, book, report preparation • AMS-TEX, developed by the American Mathematical Society for professional mathematical typesetting.

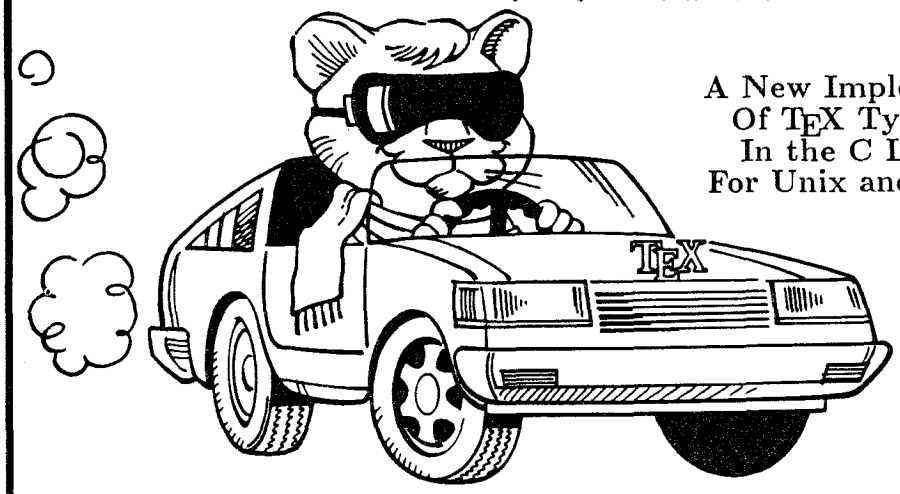
OUTPUT DEVICE DRIVERS for • Epson FX, LQ • Toshiba • Cordata LP-300 Series • HP LaserJet Series • Apple LaserWriter • APS-5 phototypesetter • Linotronic • Compugraphic • Screen preview, with EGA or Hercules card.

REQUIRES: IBM PC/XT, AT or compatible, DOS 2.0 or higher & 512K RAM; hard disk for printer drivers & fonts.

(Also available: Printer drivers; interfaces to Bitstream Fontware; Metafont, to design-your-own-type, PC Paintbrush, PC Palette, FancyFont & Fontrix. Complete packages, including laser printer, printer driver, pcTEX, and screen previewer from $2850. Site licenses and volume discounts available.)

This ad is typeset & composed using: pcTEX, Bitstream fonts & laser printer. Logotype & black backgrounds done photographically.

Introducing TurboTEX



A New Implementation Of TEX Typesetting In the C Language For Unix and MS-DOS

☐ *Executables* $100
☐ *With source* $200

TurboTEX, the new cousin to the TEX lion, is racing into town with a new implementation of TEX for high-performance typesetting. He's driving a speedy vehicle loaded with all the options:

- TRIP-certified INITEX, VIRTEX, and preloadable TEX
- The Computer Modern fonts—both bitmaps and metrics
- Printer drivers for HP Laserjet, Postscript, and dot-matrix printers
- LATEX macros and styles

If you'd like to open the hood and do your own tune-ups, order the TurboTEX source code in portable C. You will receive some 50,000 lines of TEX and TurboTEX source code, including our WEB system in C, and PASCHAL, our Pascal-to-C translator. TurboTEX fits C portability standards like POSIX, SVID, and Kernighan & Ritchie.

Availability: Executables are available now for IBM PC's and compatibles, and AT&T 3B1 and 3B2 Unix. Unix and MS-DOS source provided on 360K 5-1/4" PC floppy disks. Source compiles with Microsoft C 5.0 on the PC or the Unix cc. TurboTEX is forthcoming for Berkeley Unix, VAX/VMS, Macintosh, and OS/2. TurboTEX requires 640K (PC) or 576K (Unix) memory and hard disk.

No-risk trial offer: Examine the documentation and run TurboTEX for 10 days. If you are not satisfied, return the software to us for a 100% refund or credit. (Offer applies to executables only.)

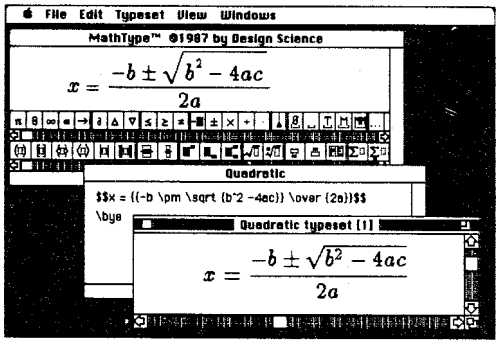
To order: Telephone and written orders accepted. Terms: Check with order, VISA, Mastercard (*free* shipping and media); Net 30 to well-rated firms and public agencies (shipping and media extra). Quantity, educational, and resale discounts available.

Kinch

The Kinch Computer Company
Publishers of TurboTEX
501 South Meadow Street
Ithaca, New York 14850
Telephone (607) 273-0222

MathType

Equations for Word Processing and TEX on the Macintosh



- MathType is a mathematical equation editing desk accessory for the Macintosh
- MathType can be used both with WYSIWYG word processors and TEX
- TEX translation is 2-way so that equations can be edited at any time by pasting back into MathType
- MathType has the easiest, most intuitive user interface of any equation editor available
- Mathematics can be entered easily by both non-mathematicians and non-TEXnicians

What People are Saying About MathType

... we recommend MathType because it is the easiest to use, ...
– David Sachs, MacWEEK, January 5, 1988.

... MathType is easier to use and produces better-looking results, ...
– Andy Oeftering, Publish!, December 1987.

Very intuitive, even a dummy can use it. Prints beautifully.
– Technology Manager, DuPont.

It looks so good that it makes us want to include more math in our documents.
– Advanced Research Analyst, McDonnell Douglas.

MathType is far easier to use than MacΣqn.
– Ken Milburn, Computers in Science, September 1987.

Design Science

- MathType is available now for only $149
- Call or write for a ***free*** demonstration disk and a brochure with sample output.

6475-B East Pacific Coast Highway, Suite 392
Long Beach, CA 90803 •• (213) 433-0685

TEXT1

Have you ever needed to create a new style sheet for LaTeX, AMS-TeX or Plain TeX, and been frustrated with the work involved in modifying or recoding macros? There is an alternative. *TEXT1*, like AMS-TeX and LaTeX adds to the basic Plain TeX macros such as table of contents, indices, parts, chapters, subtitles, running head and foot text, lists, and auto numbered footnotes and endnotes.

What makes *TEXT1* unique is its *global* formatting macros which give you the ability to easily modify the style sheet (formatting) of each of the local macros, without the need of an experienced TeXnician. For example, if you wanted your chapters to be lettered instead of numbered, or if you wanted to alternate the author's name and chapter title between even and odd pages, you would simply use one global command and change the format. Global format commands let you letter footnotes, or have them print in front of a word instead of following a word. Another global format command allows you to change the list hierarchy from "1., a., i., 1), a), i)" as defined by Turabian to an outline style, "I., A., 1.".

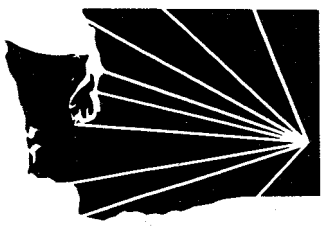
TEXT1 **has been used by thousands of students, staff and faculty since 1986 to meet their text processing needs.**

TEXT1 **can be run on any computer that runs TeX.**

TEXT1 **comes with a complete** *Refrence Manual* **and** *Users Guide*.

TEXT1 **optionally has available an** *International Phonetic Alphabet* **font in** *GF*, *PK* **or** *PXL* **format. Also available is a** *Compugraphics 8600* **driver.**

To order TEXT1, write to



TEXT1 Distribution
Computing Service Center
Washington State University
Pullman, WA 99164-1220

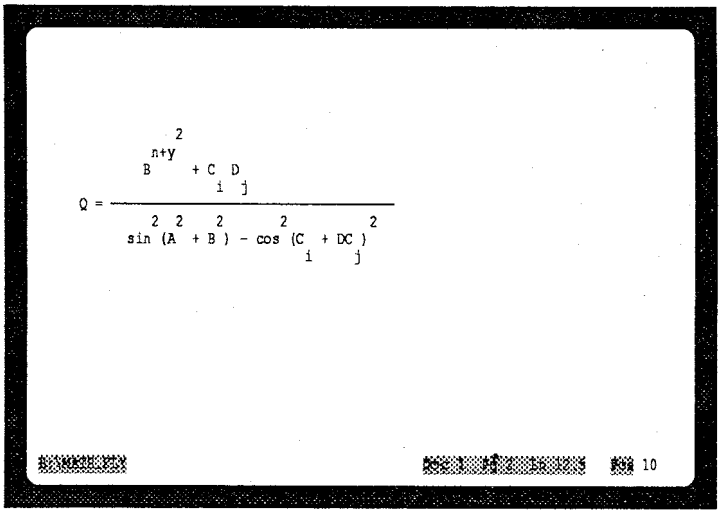
or call

TEXT1 Distribution
(509) 335-0411

Publishing Companion 1.03

Desktop Publishing Has Never Been Simpler and Will Never Be the Same

Mathematics



What you see on the Word-Perfect screen

Is fully translated to TEX as shown below

$$Q = \frac{B^{n+y^2} + C_i D_j}{sin^2(A^2 + B^2) - cos^2(C_i + DC_j)^2}$$

K-TALK® COMMUNICATIONS

3920 Olentangy River Road
Columbus, Ohio 43214 U.S.A.
(614) 459-9711

European dealers:

Bruce Wolman of KAOS A/S, Boks 3169 Elisenberg, 0208 Oslo 2 NORWAY -- Telephone +47 02-59-02-94
Knud Lønsted of Interbase, Dantes Plads 1, DK-1556 København V Denmark -- Telephone +45 01-93-28-29
Ewart North of UniTEX Systems, 12 Dale View Road, Beauchief, Sheffield S8 0EJ. -- Telephone (0742) 351 489

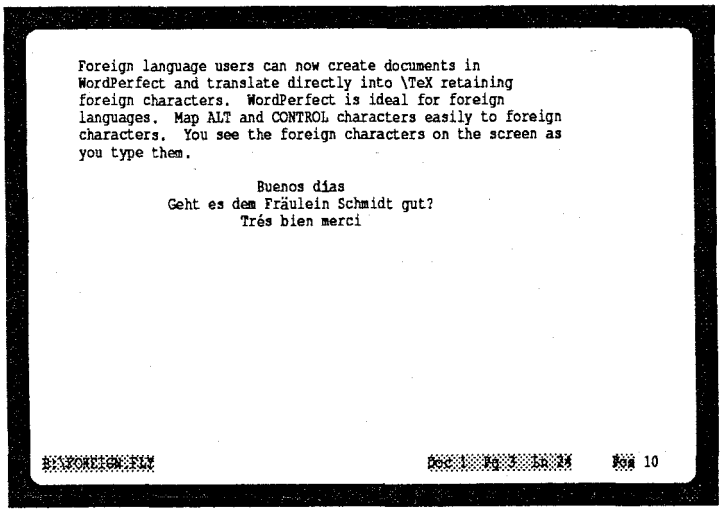
Publishing Companion priced at $179.00. Screen capture program used for above screen available for $79.00. Add shipping and handling of $4.00 for US & Canada or $35.00 for Europe. Prepayment or purchase order required. A remittance from outside the United States must be payable in U.S. dollars and can be an international money order or a check drawn on a U.S., Canadian or European bank.

Printed in U.S.A. 8802-2

Publishing Companion 1.03

Desktop Publishing Has Never Been Simpler and Will Never Be the Same

Foreign Language Support



What you see on the Word-Perfect screen

Is fully translated to TEX as shown below

Foreign language users can now create documents in WordPerfect and translate directly into TEX retaining foreign characters. WordPerfect is ideal for foreign languages. Map ALT and CONTROL characters easily to foreign characters. You see the foreign characters on the screen as you type them.

Buenos días
Geht es dem Fräulein Schmidt gut?
Trés bien merci

K-TALK® COMMUNICATIONS

3920 Olentangy River Road
Columbus, Ohio 43214 U.S.A.
(614) 459-9711

European dealers:

Bruce Wolman of KAOS A/S, Boks 3169 Elisenberg, 0208 Oslo 2 NORWAY -- Telephone +47 02-59-02-94
Knud Lønsted of Interbase, Dantes Plads 1, DK-1556 København V Denmark -- Telephone +45 01-93-28-29
Ewart North of UniTEX Systems, 12 Dale View Road, Beauchief, Sheffield S8 0EJ. -- Telephone (0742) 351 489

Publishing Companion priced at $179.00. Screen capture program used for above screen available for $79.00. Add shipping and handling of $4.00 for US & Canada or $35.00 for Europe. Prepayment or purchase order required. A remittance from outside the United States must be payable in U.S. dollars and can be an international money order or a check drawn on a U.S., Canadian or European bank.

Printed in U.S.A. 8802-1

MAKE THE

Feature	Preview 4.6	MAXview 2.n
Select DVI file	on-the-fly	at start-up
Select page size	at start-up	at start-up
Select page offsets	at start-up	at start-up
Multiple page mode	x	
Two-up page mode	x	
Display current page number	x	
Display font, size, position, and scaling of character at cursor	x	
Select magnification level	on-the-fly	at start-up
Magnification levels	15,000	4
Search for words	x	
Reposition page	x	x
Pica ruler	at cursor	at edge of screen
Physical page outline	x	x
User-specified font directories	x	x
Support PXL and PK font format	x	x
Substitution for missing fonts	any number, specified in config or sub file	single font, specified at start-up
Font substitution for existing fonts	any number	
Can use printer driver fonts	x	x
Additional fonts provided	x	
PostScript screen fonts provided	x	
PC/AT startup time (simple document with one font)	8 seconds	8 seconds
PC/AT time to draw 20 pages for simple document	84 seconds	127 seconds
PC/AT startup time (complex document with 14 fonts)	9 seconds	19 seconds
PC/AT minimum time to draw 10 pages for complex document	34 seconds	60 seconds
Price	$175	$125

535 W. William St. Suite 300 Ann Arbor, MI 48103 (313) 996-3566 FAX (313) 996-3573

COMPARISON.

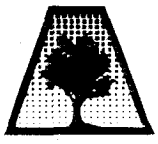
ArborText has released a new Preview for the PC which is *30% faster* than our previous version. As you can see from our benchmarks on the opposite page*, there is no comparison.

We've also added landscape to DVI-LASER/HP, our Hewlett-Packard LaserJet Plus and LaserJet II driver, and a soft font conversion utility that allows you to create fonts .pk and .pl formats for use in Preview or on other printers. Both programs are available as upgrades from Personal TEX

REQUIREMENTS FOR Preview 4.6 ARE:

- ☐ IBM PS/2 or PC (or true compatible) equipped with one of IBM EGA (mono or color), MCGA, VGA, Hercules Graphics Card, Olivetti Monochrome Graphics Card, Tecmar Graphics Master card, Genius VHR Full Page Display Monitor, ETAP Neftis monitor, Toshiba 3100, or AT&T 6300.
- ☐ PC TEX or DVI files

So if you're looking for a full-featured page previewer that won't slow you down, remember...
make the comparison.



ARBORTEXT INC.

For more information, call our distributor Personal TEX at (415) 388–8853 or ARBORTEXT INC. at (313) 996–3566.

This advertisement was created using *The Publisher.*

TEX is a trademark of the American Mathematical Society. LaserJet Plus and LaserJet II are trademarks of Hewlett-Packard, Co. PS/2 and PC are trademarks of IBM, Corp.

* The benchmarks on the opposite page are based on ArborText trials.

535 W. William St. Suite 300 Ann Arbor, MI 48103 (313) 996-3566 FAX (313) 996-3573

Index of Advertisers

COMPUTERS & TYPESETTING:
Errata and Changes

As of 15 February 1988

This document contains all known errata and changes to *Computers & Typesetting*, Volumes A–E, as compiled by the TEX Project at Stanford, as of the date shown above. An up-to-date log of these changes is available on-line to users with Internet access. Changes through 15 June 1987 are in the file `<tex.doc>ERRATA.THR`, and later changes, in `<tex.doc>ERRATA.TeX`; both files are on the Score system at Stanford (`@SCORE.Stanford.Edu`).

Beginning with this edition, updated versions of the full errata list will be published only once a year, for distribution with the first issue of TUGboat. Supplements will appear as necessary in subsequent TUGboat issues. The date of the last entry in each section of this document is now listed in the contents, below, so that it is no longer necessary to check the detail to see whether anything has been added.

CONTENTS

Distributed with TUGboat Volume 9 (1988), No. 1. Published by

TEX Users Group

P. O. Box 9506

Providence, R.I. 02940-9506, U.S.A.

Colophon for *Computers & Typesetting*

The five volumes of *Computers & Typesetting* were composed by TEX (TEX82) using the Computer Modern (CM85) fonts as produced by METAFONT (METAFONT84). Thus, the books themselves describe exactly how they were prepared for printing. Camera-ready copy was set on an Autologic APS Micro-5 typesetter at Stanford University from font images resident on the host computer and shipped to the typesetter a character at a time, as needed.

The proof-style illustrations in Volume E were also set on the Micro-5, each figure comprising two images that were combined photographically with the text material after the images of the character shapes had been screened. The "color separation" to produce those proofs was done by a program written for that purpose.

All the copy on the cover, spine, and book jackets was also typeset by the APS, using Computer Modern fonts, except for the ISBN number.

The books were printed on Finch Opaque, basis 50 lb. acid-free paper, which has a life expectancy of several hundred years. The hardcover edition was printed and bound by Halliday Lithograph Corp., Hanover, Massachusetts, as were the spiral-bound editions of *The TEXbook* and *The METAFONTbook*.

Colophon for these errata

The errata and changes were composed by TEX running on a VAX 8600 (VMS) at the American Mathematical Society. Camera-ready copy was prepared on an Autologic APS Micro-5, using a preliminary version of disk-resident CM fonts (these fonts will be available from Autologic as soon as a few remaining bugs have been exterminated).

This document was printed in the printing department of the AMS, on Weyerhauser Cougar 50 lb. acid-free paper.

Bugs in *Computers & Typesetting*

15 February 1988

This is a list of all corrections made to *Computers & Typesetting*, Volumes A–E. It also includes all corrections made to the softcover version of *The TEXbook*, beginning with the sixth printing (January 1986); these are the same as corrections to Volume A. Corrections to the softcover version of *The METAFONTbook* are the same as corrections to Volume C. Some of these corrections have already been made in reprintings of the books.

Page A7, fourth line from the bottom (6/28/86)

---

since control sequences of the second kind always have exactly one symbol after

---

Page A35, second-last line (1/31/87)

He may run who reads.

— HABAKKUK 2:2 (c. 600 B.C.)

He that runs may read.

Page A43, lines 8–9 (8/23/86)

---

of Appendix B, which defines % to be a special kind of symbol so that you can use it for comments, defines the control sequence \% to mean a percent sign.

Page A45, lines 10–13 (8/23/86)

TEX adds 64. Hence code 127 can be typed `^^?`, and the dangerous bend sign can be obtained by saying `{\manual^^?}`. However, you must change the category code of character 127 before using it, since this character ordinarily has category 15 (invalid); say, e.g., `\catcode`\^^?=12`. The `^^` notation is different from `\char`, because `^^`

Page A76, line 7 (8/23/86)

---

and extra space; for example, these quantities are 3.33333 pt, 1.66666 pt, 1.11111 pt,

---

Page A83, bottom line (5/19/87)

[This line should be flush right.]

Page A111, 7th-last line, right-hand column (2/15/87)

---

if $b = 10000$ and $-10000 < p < 10000$ and $q < 10000$;

Page A117, second-last line (6/10/87)

---

marks; sometimes also `$\|$` (‖). You can say, e.g., '`\footnote\dag{...}`'.

Page A124, lines 6–11 (2/26/87)

---

of insertion; an additional '`\penalty-10000`' item is assumed to be present at the end of the vertical list, to ensure that a legal breakpoint exists.) Let u be the natural height plus depth of that least-cost box, and let r be the penalty associated with the optimum breakpoint. Decrease g by uf, and increase q by r. (If `\tracingpages=1`, the log file should now get a cryptic message that says '`% split` n `to` v`,`u `p=`r'. For example,

```
% split254 to 180.2,175.3 p=100
```

Page A158, lines 6–8 (2/20/87)

the second atom, which has subscript i; the superscripts are empty except for the last atom, whose superscript is $\overline{n+1}$. This superscript is itself a math list consisting of one atom, whose nucleus is $n+1$; and that nucleus is a math list consisting of three atoms.

Page A159, line 22 (2/15/88)

'`\nolimits`' if the normal `\displaylimits` convention has been overridden; a Rad

Page A171, line 20 (1/26/86)

will be surrounded by more space than there would be if that subformula were enclosed

Page A176, line 1 (8/23/86)

You can insert '`\noalign{`⟨vertical mode material⟩`}`' just after any `\cr` within

Page A213, lines 34–35 (12/23/87)

text will be a single control sequence token, defined to be like `\relax` if its meaning is currently undefined.

Page A248, line 17 (6/17/86)

'`&`' or '`\span`' or '`\cr`', it needs some way to decide which alignment is involved.

Page A249, line 20 (6/17/86)

line (see Chapter 8). If you don't want a `\cr` at the end of a certain line, just type

Page A276, line 19 (1/27/86)

| `\font`⟨control sequence⟩⟨equals⟩⟨file name⟩⟨at clause⟩
| ⟨global assignment⟩

[The bottom line of p. 276 will now move to the top of p. 277.]

Page A277, lines 31–32 (1/27/86)

⟨font assignment⟩ ⟶ `\fontdimen`⟨number⟩⟨font⟩⟨equals⟩⟨dimen⟩

Page A286, sixth-last line (4/28/87)

`\sfcode` table as described in Chapter 12; characters numbered 128 to 255 set the

Page A287, line 19 (2/15/87)

▪ `\-`. This "discretionary hyphen" command is defined in Appendix H.

Page A292, lines 9–10 (2/15/87)

▪ `\-`. This command is usually equivalent to '`\discretionary{-}{}{}`'; the '`-`' is therefore interpreted as a hyphen, not as a minus sign. (See Appendix H.)

Page A308, lines 25–26 (6/1/87)

```
\def\appendroman#1#2#3{\edef#1{\csname
      \expandafter\gobble\string#2\romannumeral#3\endcsname}}
```

Page A312, lines 10–14 (8/23/86)

12.11. The interline glue will be zero, and the natural height is $1 + 1 - 3 + 2 = 1$ pt (because the depth of `\box2` isn't included in the natural height); so the glue will ultimately become `\vskip-1pt` when it's set. Thus, `\box3` is 3 pt high, 2 pt deep, 4 pt wide. Its reference point coincides with that of `\box2`; to get to the reference point of `\box1` you go up 2 pt and right 3 pt.

Page A312, line 21 (8/23/86)

up 4 pt to get to the upper left corner of `\box4`; then down -1.6 pt, i.e., up 1.6 pt, to

Page A319, line 20 (3/31/87)

make ordinary periods act like `\cdot` symbols: Just define `\mathcode'.` to be `"0201`,

Page A326, line 12 (12/12/87)

its natural width. The `\hbox` version also invokes `\everymath`.

Page A328, lines 18–19 (5/14/87)

not performed while the expansion is taking place, and the control sequences following `\def` are expanded; so the result is an infinite string

```
A\def A\def A\def A\def A\def A\def A\def A...
```

Page A329, lines 14–15 (8/23/86)

20.5. The `##` feature is indispensable when the replacement text of a definition contains other definitions. For example, consider

Page A356, lines 6–7 (1/30/87)

```
  \spaceskip=.3333em \xspaceskip=.5em\relax}
\def\ttraggedright{\tt\rightskip=0pt plus2em\relax}
```

Page A356, line 33 (6/1/87)

```
    \vbox to.2ex{\hbox{\char'26}\vss}\hidewidth}}
```

Page A357, tenth-last line (10/13/86)

```
\let\sp=^ \let\sb=_        {\catcode`\_=\active \global\let_=\_}
```

Page A357, third-last and second-last lines (2/17/87)

```
\def\pr@m@s{\ifx'\next\let\nxt\pr@@@s \else\ifx^\next\let\nxt\pr@@@t
  \else\let\nxt\egroup\fi\fi \nxt}
```

Page A364, fifth-last line (1/30/87)

```
\def\fmtname{plain}\def\fmtversion{2.3} % identifies the current format
```

Page A368, bottom line (2/26/86)

that includes the symbols $\leftarrow$, $\downarrow$, $\neq$, $\leq$, and $\geq$, and he finds that this makes it much more

Page A379, line 15 (10/12/87)

```
\def\deleterightmost#1{\edef#1{\expandafter\xyzzy#1\xyzzy}}
```

Page A396, line 13 (8/23/86)

```
\hyphenpenalty=10000 \exhyphenpenalty=10000
```

Page A414, line 10 (3/4/86)

```
\font\titlefont=cmssdc10 at 40pt   % titles in chapter openings
```

Page A427, line 7 (2/23/86)

the author's book *Computer Modern Typefaces.*)

Page A428, lines 18–20 (6/15/87)

The first eight of these all have essentially the same layout; but `cmr5` needs no ligatures, and many of the symbols of `cmti10` have different shapes. For example, the ampersand becomes an 'E.T.', and the dollar changes to pound sterling:

Page A434, lines 25–28 (8/17/86)

from \nu (ν). Similarly, `\varsigma` (ς) should not be confused with `\zeta` (ζ). It turns out that `\varsigma` and `\upsilon` are almost never used in math formulas; they are included in plain TEX primarily because they are sometimes needed in short Greek citations (cf. Appendix J).

Page A447, line 32 (6/1/87)

ters also affect mathematical typesetting: dimension parameters `\delimitershortfall`

Page A454, lines 23–29 (8/13/87)

 If a suitable starting letter is found, let it be in font f. Hyphenation is abandoned unless the `\hyphenchar` of f is between 0 and 255, and unless a character of that number exists in the font. If this test is passed, TEX continues to scan forward until coming to something that's not one of the following three "admissible items": (1) a character in font f whose `\lccode` is nonzero; (2) a ligature formed entirely from characters of type (1); (3) an implicit kern. The first inadmissible item terminates this part of the process; the trial word consists of all the letters found in admissible items. Notice that all of these letters are in font f.

Page A455, new paragraph to follow line 9 (2/15/87)

 The control sequence `\-` is equivalent to `\discretionary{\char`h`}{}{}`, where h is the `\hyphenchar` of the current font, provided that h lies between 0 and 255. Otherwise `\-` is equivalent to `\discretionary{}{}{}`.

Page A457, left column, fifth-last line (2/17/87)

Page A458, left column, line 6 (2/15/87)

Page A458, left column, line 19 (2/15/88)

Page A458, left column, near the bottom (5/19/87)

\! (exclamation point), 51, *72*, 73, 75, *169*.

[This saves a line that otherwise would make the index too long on page 481!]

Page A458, right column, line 10 (11/27/86)

~ (tilde), 38, 51, 343, 353; *see also* ties.

Page A458, right column (6/14/87)

*\accent (general accent), 9, 54, 86, 283, 286.

Page A461, entry for boxes (3/16/87)

boxes, 63–67, 77–83, 221–229.

Page A461, entry for \centering (1/28/86)

\centering, 347, 348, 362.

Page A462, left column, line 7 (10/9/87)

152, 178, 360.

Page A462, entry for ⟨code assignment⟩ (1/27/86)

⟨code assignment⟩, 277.

Page A464, left column, line 3 (2/15/87)

discretionary hyphens, 28, 95–96, 453, 455.

Page A465, entry for \everymath (12/12/87)

[Include also a reference to page 326.]

Page A465, right column, line 8 (5/3/87)

expansion of expandable tokens, 212–216, 238,

Page A466, entry for \font, second line (1/27/86)

271, 276.

Page A466, new entry (2/3/87)

⟨fontdef token⟩, 271.

Page A467, entry for \hideskip (1/28/86)

\hideskip, 347, 348, 354.

Page A468, left column line 2 (2/15/87)

351, 395, *414*, 454, 455.

Page A470, entry for manfnt (1/15/86)

manfnt, 44, 408, 414.

Page A471, entry for \medbreak (10/13/86)

\medbreak, 111, 113, 353, *355*, *419*, *422*.

Page A471, entry for \moveright (2/27/87)

*\moveright, 80–81, *221*, 282.

Page A471, entry for Mozart, second line (3/19/86)

Gottlieb (= Theophilus = Amadeus), 409.

Page A472, the entry for \not (2/12/87)

[The overprinting here is intentional, since \not is a character of width zero. More than a dozen people have reported this as an error, but it is not!]

Page A473, entry for 'page builder' (8/13/87)

when exercised, 122, 280–283, 286–287.

Page A477, entry for \span (5/3/87)

*\span, 215, 238, 243, *244*, 245, 248, 249,
282, *330*, 385.

Page A479, entry for ties, second line (11/27/86)

173, 353, *404*.

Page A480, changes to various entries (6/14/87)

*\underline, *130–131*, 141, 291, 443.
*\unhbox, 120, 283, 285, *354*, *356*, *361*, *399*.
*\unhcopy, 120, 283, 285, *353*.
*\unkern, 280.
*\unpenalty, 280.
*\unskip, 222–223, 280, 286, *313*, *392*, *418–419*.
*\unvbox, 120, 254, 282, 286, *354*, *361*, *363*, *364*, *392*, *399*, *417*.
*\unvcopy, 120, 282, 286, *361*.
*\vadjust, 95, 105, 109, 110, 117, 259, 281, 393, 454.
*\valign, 249, 283, 285–286, 302, *335*, *397*.
*\vcenter, 150–151, 159, 170, 193, 222, 242,
*\vfil, 71, 72, 111, 256, 281, 286, 417.
*\vfill, 24, 25, 71, 72, 256–257, 281, 286.
*\vfilneg, 72, 111, 281, 286.
\voidb@x, 347, 348.

Page A480, right column (2/15/88)

\vdots (⋮), *177*, 359.

Page A481, left column (6/14/87)

*\vss, 71, 72, *255*, 281, 286.

Page A481, right column (7/3/87)

\z@, 347, 348.
\z@skip, 347, 348.

Volume B, in general (7/28/86)

[A number of entries were mistakenly omitted from the mini-indexes on the right-hand pages. Here is a combined list of all the missing items; you can mount it inside the back cover, say, as a secondary mini-index when the first one fails...]

active_base = 1, §222.
aux = macro, §213.
begin_name: **procedure**, §515.
big_switch = 60, §1030.
choice_node = 15, §689.
cur_boundary: 0 .. *save_size*, §271.
cur_c: *quarterword*, §724.
cur_group: *group_code*, §271.
cur_i: *four_quarters*, §724.
cur_level: *quarterword*, §271.
do_extension: **procedure**, §1348.
dvi_buf: **array**, §595.
dvi_gone: *integer*, §595.
dvi_limit: *dvi_index*, §595.
dvi_offset: *integer*, §595.
dvi_ptr: *dvi_index*, §595.
end_graf: **procedure**, §1096.
error: **procedure**, §82.
error_stop_mode = 3, §73.
font_base = 0, §12.
font_info: **array**, §549.
get_token: **procedure**, §365.
glue_base = 2626, §222.
half_buf: *dvi_index*, §595.
handle_right_brace: **procedure**, §1068.
hash_base = 258, §222.
head = macro, §213.
hyf_distance: **array**, §921.
hyf_next: **array**, §921.
hyf_num: **array**, §921.
index = macro, §302.
inf: *boolean*, §448.
init_col: **procedure**, §788.
init_span: **procedure**, §787.
input_ln: **function**, §31.
interaction: 0 .. 3, §73.
limit = macro, §302.
line_width: *scaled*, §830.
macro_call: **procedure**, §389.
main_control: **procedure**, §1030.
mem: **array**, §116.
mem_bot = 0, §12.
mem_end: *pointer*, §118.
mem_top = macro, §12.
mlist_to_hlist: **procedure**, §726.
mode = macro, §213.
mode_line = macro, §213.
more_name: **function**, §516.
mu: *boolean*, §448.
name = macro, §302.
nest: **array**, §213.
off_save: **procedure**, §1064.
open_log_file: **procedure**, §534.
output_active: *boolean*, §989.
p: *pointer*, §498.
param_stack: **array**, §308.
pool_file: *alpha_file*, §50.
pool_ptr: *pool_pointer*, §39.
prefixed_command: **procedure**, §1211.
prev_depth = macro, §213.
prev_graf = macro, §213.
prev_prev_r: *pointer*, §830.
print_err = macro, §73.
r: *trie_pointer*, §960.
reconstitute: **function**, §906.
resume_after_display: **procedure**, §1200.
save_ptr: 0 .. *save_size*, §271.
save_stack: **array**, §271.
scan_dimen: **procedure**, §448.
scan_math: **procedure**, §1151.
short_display: **procedure**, §174.
show_node_list: **procedure**, §182.
start = macro, §302.
state = macro, §302.
str_pool: **packed array**, §39.
str_ptr: *str_number*, §39.
str_start: **array**, §39.
tail = macro, §213.
trap_zero_glue: **procedure**, §1229.
trie: **array**, §921.
trie_char = macro, §921.
trie_link = macro, §921.
trie_op = macro, §921.
vlist_out: **procedure**, §629.
write_loc: *pointer*, §1345.

Volume B, in general (4/6/87)

[The percent signs in all the comments (for example, on pages 7 and 50) are in the wrong font! Change '**%**' to '%'.]

Page Bvi, bottom line, and top line of next page (10/12/86)

puter Science Report 1097 (Stanford, California, April 1986), 146 pp. *The WEB programs for four utility programs that are often used with TEX: POOLtype, TFtoPL, PLtoTF, and DVItype.*

Page B2, line 32 (12/23/87)

define *banner* ≡ `'This␣is␣TeX,␣Version␣2.9'` { printed when TEX starts }

Page B7, new line after line 25 (1/28/87)

if *max_in_open* ≥ 128 **then** *bad* ← 6;

Page B13, first three lines (4/7/87)

The '*name*' parameter, which is of type '**packed array** [⟨*any*⟩] **of** *char*', stands for the name of the external file that is being opened for input or output. Blank spaces that might appear in *name* are ignored.

Page B14, line 30 (4/7/87)

31. The *input_ln* function brings the next line of input from the specified file into available

Page B18, line 30 (5/22/86)

str_ptr: *str_number*; { number of the current string being created }

Page B21, first line of mini-index, right column (6/14/87)

pool_name
= "`string`", §11.

Page B34, lines 5–6 (6/14/87)

to delete a token, and/or if some fatal error occurs while TEX is trying to fix a non-fatal one. But such recursion is never more than two levels deep.

Page B52, line 5 (8/13/87)

cannot be done, i.e., if $hi\_mem\_min = lo\_mem\_max + 1$, we have to quit.

Page B55, lines 12–13 (4/21/87)

if $r = p$ **then if** $rlink(p) \neq p$ **then** ⟨ Allocate entire node p and **goto** *found* 129 ⟩;

Page B57, lines 25–28 (6/14/87)

The first of these has *font* = *font_base*, and its *link* points to the second; the second identifies the font and the character dimensions. The saving feature about oriental characters is that most of them have the same box dimensions. The *character* field of the first *char_node* is a "*charext*" that distinguishes between graphic symbols whose dimensions are identical for typesetting purposes. (See the METAFONT manual.) Such an extension of TEX would not be difficult; further details are left to the reader.

Page B58, second line of section 136 (7/23/86)

the values corresponding to '`\hbox{}`'. The *subtype* field is set to *min_quarterword*, since that's

Page B66, lines 2–8 (4/21/87)

location is more efficient than dynamic allocation when we can get away with it. For example, locations *mem_bot* to *mem_bot* + 3 are always used to store the specification for glue that is '`0pt plus 0pt minus 0pt`'. The following macro definitions accomplish the static allocation by giving symbolic names to the fixed positions. Static variable-size nodes appear in locations *mem_bot* through *lo_mem_stat_max*, and static single-word nodes appear in locations *hi_mem_stat_min* through *mem_top*, inclusive. It is harmless to let *lig_trick* and *garbage* share the same location of *mem*.

Page B67, line 23 (4/13/87)

{ previous *mem_end*, *lo_mem_max*, and *hi_mem_min* }

Page B71, line 17 (4/15/87)

begin while $p > mem\_min$ **do**
[Now *null* can be removed from the mini-index.]

Page B74, line 24 (4/15/87)

procedure *show_node_list*(*p* : *integer*); { prints a node list symbolically }

Page B74, line 33 (4/15/87)

while $p > mem\_min$ **do**

Page B84, line 12 (2/15/87)

define *relax* = 0 { do nothing (\relax) }

Page B86, third line of section 210 (8/23/86)

that their special nature is easily discernible. The "expandable" commands come first.

Page B88, line 23 (5/22/86)

procedure *print_mode*(*m* : *integer*); { prints the mode represented by *m* }

Page B93, lines 3–4 (8/17/86)

In the first region we have 128 equivalents for "active characters" that act as control sequences, followed by 128 equivalents for single-character control sequences.

Page B130, ninth-last line (5/7/87)

This variable has six possible values:

Page B151, line 9 (4/22/87)

begin if (*end_line_char* < 0) ∨ (*end_line_char* > 127) **then** *incr*(*limit*);
if *limit* = *start* **then** { previous line was empty }

Page B154, lines 25, 29, 34 respectively (9/20/87)

cvl_backup, *radix_backup*, *co_backup*: *small_number*; { to save *cur_val_level*, etc. }
co_backup ← *cur_order*; *backup_backup* ← *link*(*backup_head*);
cur_order ← *co_backup*; *link*(*backup_head*) ← *backup_backup*;

Page B155, new entry for mini-index (9/20/87)

cur_order: *glue_ord*, §447.

Page B156, line 28 (12/23/87)

begin *eq_define*(*cur_cs*, *relax*, 256);

Page B157, mini-index (12/23/87)

Delete the entries for '*eqtb*' and '*frozen_relax*'; replace them by the following: *eq_define*: **procedure**, §227.
relax = 0, §207.

Page B160, lines 17–20 (7/28/86)

389. After parameter scanning is complete, the parameters are moved to the *param_stack*. Then the macro body is fed to the scanner; in other words, *macro_call* places the defined text of the control sequence at the top of TEX's input stack, so that *get_next* will proceed to read it next.

Page B200, top line (5/5/87)

495. When we begin to process a new `\if`, we set *if_limit* ← *if_code*; then if `\or` or `\else` or `\fi`

Page B217, lines 15–16 (6/14/87)

`DVI` format.

Page B224, lines 4–7 of section 560 (10/22/86)

name and area strings *nom* and *aire*, and the "at" size s. If s is negative, it's the negative of a scale factor to be applied to the design size; $s = -1000$ is the normal case. Otherwise s will be substituted for the design size; in this case, s must be positive and less than 2048 pt (i.e., it must be less than 2^{27} when considered as an integer).

Page B224, second-last line (4/28/87)

```
done: if file_opened then b_close(tfm_file);
  read_font_info ← g;
```

Page B229, lines 6–8 (11/17/87)

than 2^{27}. If $z < 2^{23}$, the individual multiplications $b \cdot z$, $c \cdot z$, $d \cdot z$ cannot overflow; otherwise we will divide z by 2, 4, 8, or 16, to obtain a multiplier less than 2^{23}, and we can compensate for this later. If z has thereby been replaced by $z' = z/2^e$, let $\beta = 2^{4-e}$; we shall compute

Page B229, lines 11–12 (11/17/87)

if $a = 0$, or the same quantity minus $\alpha = 2^{4+e}z'$ if $a = 255$. This calculation must be done exactly, in order to guarantee portability of TEX between computers.

Page B230, lines 2–5 (11/17/87)

```
begin alpha ← 16;
while z ≥ '40000000 do
  begin z ← z div 2; alpha ← alpha + alpha; end;
beta ← 256 div alpha; alpha ← alpha * z;
```

Page B245, new entry for mini-index (8/7/87)

cur_s: *integer*, §616.

Page B254, line 29 (8/7/87)

```
cur_s: integer;  { current depth of output box nesting, initially −1 }
```

Page B254, line 31 (8/7/87)

[Remove the statement '*cur_s* ← −1;' and put it on page B244 at the end of line 31.]

Page B255, mini-index at the bottom (4/15/87)

mag = macro, §236.

Page B257, lines 11–13 (6/14/87)

```
if c ≥ qi(128) then dvi_out(set1);
dvi_out(qo(c));
```

Page B259, line 13 (11/9/87)

begin *rule_wd* ← *rule_wd* + 10; { compensate for floating-point rounding }
edge ← *cur_h* + *rule_wd*; *lx* ← 0; ⟨ Let *cur_h* be the position of the first box, and set

Page B259, line 17 (11/9/87)

cur_h ← *edge* − 10; **goto** *next_p*;

Page B260, lines 7–8 (4/15/87)

In the case of *c_leaders* (centered leaders), we want to increase *cur_h* by half of the excess space not occupied by the leaders; and in the case of *x_leaders* (expanded leaders) we increase

Page B263, line 21 (11/9/87)

begin *rule_ht* ← *rule_ht* + 10; { compensate for floating-point rounding }
edge ← *cur_v* + *rule_ht*; *lx* ← 0; ⟨ Let *cur_v* be the position of the first box, and set

Page B263, line 25 (11/9/87)

cur_v ← *edge* − 10; **goto** *next_p*;

Page B266, line 8 (8/7/87)

dvi_out(*eop*); *incr*(*total_pages*); *cur_s* ← −1;

Page B266, new code between lines 31 and 32 (8/7/87)

```
while cur_s > -1 do
  begin if cur_s > 0 then dvi_out(pop)
  else begin dvi_out(eop); incr(total_pages)
    end;
  decr(cur_s);
  end;
```

Page B267, mini-index at the bottom (8/7/87)

cur_s: *integer*, §616. *mag* = macro, §236. *pop* = 142, §586.

Page B271, line 10 (8/23/86)

which will be ignored in the calculations because it is a highly negative number.

Page B285, lines 23 and 24 (5/4/83)

the current string would be '`.^._/`' if *p* points to the *ord_noad* for *x* in the (ridiculous) formula '`\sqrt{a^{\mathinner{b_{c\over x+y}}}}$`'.

Page B296, lines 3–5 (5/8/87)

box *b* and changes it so that the new box is centered in a box of width *w*. The centering is done by putting `\hss` glue at the left and right of the list inside *b*, then packaging the new box; thus, the actual box might not really be centered, if it already contains infinite glue.

Page B338, second-last line (8/19/87)

q ← *link*(*head*); *s* ← *head*;

Page B339, line 4 (8/19/87)

```
    s ← q; q ← link(q);
```

Page B339, new code to insert after line 10 (8/19/87)

```
  if o ≠ 0 then
    begin r ← link(q); link(q) ← null; q ← hpack(q, natural);
    shift_amount(q) ← o; link(q) ← r; link(s) ← q;
    end;
```

[These new lines also imply changes to the index that aren't shown in this errata list.]

Page B346, line 19 (5/19/87)

```
pass_number: halfword;   { the number of passive nodes allocated on this pass }
```

Page B350, lines 36 and 37 (1/28/87)

```
v: pointer;   { points to a glue specification or a node ahead of cur_p }
t: integer;   { node count, if cur_p is a discretionary node }
```

Page B353, lines 8–22 (1/28/87)

```
  s ← cur_p;
  if break_type > unhyphenated then if cur_p ≠ null then
      ⟨ Compute the discretionary break_width values 840 ⟩;
  while s ≠ null do
    .
    .
    .          [as before, but indented one less notch]
  end;
```

Page B354, line 6 (1/28/87)

will be the background plus l_1, so the length from *cur_p* to *cur_p* should be $\gamma + l_0 + l_1 - l$, minus the length of nodes that will be discarded after the discretionary break.

Page B354, lines 12–18 (1/28/87)

```
  begin t ← replace_count(cur_p); v ← cur_p; s ← post_break(cur_p);
  while t > 0 do
    begin decr(t); v ← link(v); ⟨ Subtract the width of node v from break_width 841 ⟩;
    end;
  while s ≠ null do
    begin ⟨ Add the width of node s to break_width and increase t, unless it's discardable 842 ⟩;
```

Page B354, new line after line 21 (1/28/87)

```
  if t = 0 then s ← link(v);   { more nodes may also be discardable after the break }
```

Page B354, lines 26–34 (1/28/87)

[Change '*s*' to '*v*' throughout this section (8 times).]

Page B354, line 9 from the bottom (1/28/87)

842. ⟨ Add the width of node *s* to *break_width* and increase *t*, unless it's discardable 842 ⟩ ≡

Page B355, lines 1–3 (1/28/87)

```
    hlist_node, vlist_node, rule_node: break_width[1] ← break_width[1] + width(s);
    kern_node: if (t = 0) ∧ (subtype(s) ≠ acc_kern) then t ← −1   { discardable }
      else break_width[1] ← break_width[1] + width(s);
    othercases confusion("disc2")
    endcases;
  incr(t)
```

Page B355, patches to mini-index at bottom (1/28/87)

acc_kern = 2, §155.
incr = macro, §16.
t: *integer*, §830.
v: *pointer*, §830.

Page B372, lines 12–14 (1/28/87)

```
    ⟨ Change discretionary to compulsory and set disc_break ← true 882 ⟩
  else if (type(q) = math_node) ∨ (type(q) = kern_node) then width(q) ← 0;
```

Page B380, fifth-last line (5/7/87)

`b` and `c`, the two patterns with and without hyphenation are `ab-cdef` and `abcdef`. Thus the

Page B386, lines 2–4 (5/21/87)

hyphenation, TEX first looks to see if it is in the user's exception dictionary. If not, hyphens are inserted based on patterns that appear within the given word, using an algorithm due to Frank M. Liang.

Page B396, bottom line (12/12/87)

```
trie_link(0) ← 0; trie_char(0) ← 0; trie_op(0) ← min_quarterword;
```

Page B397, line 28 (5/21/87)

$h = z - c$. It follows that location *trie_max* will never be occupied in *trie*, and we will have

Page B415, the mini-index (4/6/87)

[Delete the spurious entry for 'c'.]

Page B419, mini-index entry for *c* (4/6/87)

c: *integer*, §994.

Page B422, line 24 (8/23/86)

```
  prev_p: pointer;   { predecessor of p }
```

Page B435, line 16 (10/12/86)

```
  width(p) ← font_info[k].sc;   { that's space(f) }
  stretch(p) ← font_info[k + 1].sc;   { and space_stretch(f) }
  shrink(p) ← font_info[k + 2].sc;   { and space_shrink(f) }
```

[And the mini-index gets three new entries: *space* = *macro*, §558. *space_shrink* = *macro*, §558. *space_stretch* = *macro*, §558.]

Page B495, lines 18 and 19 (2/15/87)

[delete these lines, since the cases cannot occur]

Page B510, line 8 (12/15/86)

```
("Pretend␣that␣you're␣Hercule␣Poirot:␣Examine␣all␣clues,")
```

Page B527, new line to follow line 13 (6/17/86)

This program doesn't bother to close the input files that may still be open.

Page B534, fourth-last line (5/4/87)

define *write_stream*(#) ≡ *info*(# + 1) { stream number (0 to 17) }

Page B544, left column (1/28/87)

acc_kern: 155, 191, 837, 842, 879, 1125.

Page B546, entry for *c* (4/6/87)

[Add a reference to section 994.]

Page B547, left column (4/7/87)

char: 19, 26–27, 520, 534.

Page B547, left column (6/14/87)

Chinese characters: 134, 585.

Page B547, right column (9/20/87)

co_backup: 366.

Page B548, right column (9/20/87)

cur_order: 366, 447, 448, 454, 462.

Page B548, right column (8/7/87)

cur_s: 593, 616, 619, 629, 640, 642.

Page B551, both columns (12/23/87)

[Remove '372' from *eqtb* and put it into *eq_define*.]

Page B553, entry for *font_base* (6/14/87)

[Insert a reference to section 134.]

Page B554, left column (12/23/87)

[Remove '372' from *frozen_relax*.]

Page B555, right column, new entry (10/25/86)

`Huge page...`, 641.

Page B556, entry for *incr* (1/28/87)

[Add a reference to section 842.]

Page B557, entry for *is_char_node* (1/28/87)

[Delete the reference to section 881.]

Page B557, right column (6/14/87)

Japanese characters: 134, 585.

Page B559, right column (8/13/87)

[Delete the entry for *low_mem_max*.]

Page B560, right column (1/28/87)

max_in_open: 11, 14, 304, 328.

Page B561, left column, line 10 (4/15/87)

169–172, 174, 178, 182, 1249, 1312, 1334.

Page B561, left column (5/1/87)

`Missing font identifier`: 577.

Page B563, left column, line 2 (4/15/87)

136, 145, 149–154, 164, 168–169, 175–176, 182,

Page B563, right column (6/14/87)

oriental characters: 134, 585.

Page B565, left column (8/7/87)

pop: 584–585, 586, 590, 601, 608, 642.

Page B567, left column (12/23/87)

[Insert '372' into *relax*.]

Page B569, right column, in appropriate places (10/12/86)

space: 547, 558, 752, 755, 1042.
space_shrink: 547, 558, 1042.
space_stretch: 547, 558, 1042.

Page B570, third-last line (1/28/87)

786, 795, 809, 819–820, 822, 837, 842–844, 866,

Page B571, right column (10/25/86)

`The following...deleted`, 641, 992, 1121.

Page B571, right column (4/7/87)

text_char: 19, 20, 25, 47.

Page B573, right column (5/1/87)

[Delete the entry for '`Undefined font code`'.]

Page B576, line 2 (1/28/87)

⟨ Add the width of node s to *break_width* and increase t, unless it's discardable 842 ⟩
Used in section 840.

Page B591, line 6 from the bottom (1/28/87)

⟨ Subtract the width of node v from *break_width* 841 ⟩ Used in section 840.

Page C14, top two lines (3/16/87)

The recursive midpoint rule for curve-drawing was discovered in 1959 by Paul de Casteljau, who showed that the curve could be described algebraically by the remarkably simple formula

Page C26, bottom line (7/18/87)

What angle corresponds to the direction North-Northwest?

Page C54, sixth-last to fourth-last lines (10/13/86)

Jonathan H. Quick (a student) used '`a.plus1`' as the name of a variable at the beginning of his program; later he said '`let plus=+`'. How could he refer to the variable '`a.plus1`' after that?

Page C76, line 14 (10/13/86)

$x_4 = w - .01in$ Point 4 should be one-hundredth of an inch inside

Page C103, line 12 (10/12/86)

$ht\# = body\_height\#;\ \ .5[ht\#, -dp\#] = axis\#;$

Page C105, line 13 (10/13/86)

The vertical line just to the right of the italic left parenthesis shows the italic

Page C107, line 13 (10/7/87)

pickup penrazor xscaled *heavyline* rotated $(\text{angle}(z_{32} - z_{31}) + 90)$;

Page C113, lines 20–27 (8/23/86)

The command '**erase fill** c' is an abbreviation for '**cullit**; **unfill** c; **cullit**'; this zeros out the pixel values inside the cyclic path c, and sets other pixel values to 1 if they were positive before erasing took place. (It works because the initial **cullit** makes all the values 0 or 1, then the **unfill** changes the values inside c to 0 or negative. The final **cullit** gets rid of the negative values, so that they won't detract from future filling and drawing.) You can also use '**draw**', '**filldraw**', or '**drawdot**' with '**erase**'; for example, '**erase draw** p' is an abbreviation for '**cullit**; **undraw** p; **cullit**', which uses the currently-picked-up pen as if it were an eraser applied to path p.

Page C124, line 9 (6/17/86)

$$branch_2 = flex((30,570),(10,590),(-1,616))$$

Page C130, 3rd-last line (9/25/86)

Geometry **1** (1986), 123–140]: Given a sequence

Page C144, sixth line of the program (8/23/86)

```
6  y2 = .1h;  top y3 = .4h;
```

Page C148, the line before the illustration (11/27/86)

are polygons with 32 and 40 sides, respectively:

[New illustrations are needed here, since METAFONT version 1.3 improves the accuracy of pen polygons.]

Page C149, 7th line after the illustration (10/24/86)

$(200, y + 100 \pm \alpha)$, where $\alpha = \sqrt{5}/4 \approx 0.559$. If we digitize these outlines and fill the

Page C175, line 23 (1/11/88)

expand into a sequence of tokens. (The language SIMULA67 demonstrated that it is

Page C178, second-last line (8/23/86)

(If $t_3 = t_1$ transum t_2, then z transformed $t_3 = z$ transformed $t_1 + z$ transformed t_2,

Page C198, fifth-last and fourth-last lines (10/13/86)

$$top\ y_2 = \text{round}(top\ \beta).$$

Such operations occur frequently in practice, so plain METAFONT provides convenient

Page C212, lines 9–11 from the bottom (8/23/86)

```
| point ⟨numeric expression⟩ of ⟨path primary⟩
| precontrol ⟨numeric expression⟩ of ⟨path primary⟩
| postcontrol ⟨numeric expression⟩ of ⟨path primary⟩
```

Page C233, lines 13–14 (2/15/87)

one column of white pixels, if the character is $2a$ pixels wide, because the right edge of black pixels is specified here to have the x coordinate $2a - 1$.

Page C247, lines 23–25 (11/27/86)

16.2. '**pencircle** scaled 1.06060' is the diamond but '**pencircle** scaled 1.06061' is the square. (This assumes that *fillin* = 0. If, for example, *fillin* = .1, the change doesn't occur until the diameter is 1.20204.) The next change is at diameter 1.5, which

Page C262, lines 1–4 (7/28/86)

When we come to macros whose use has not yet been explained—for example, somehow `softjoin` and `stop` never made it into Chapters 1 through 27—we shall consider them from a user's viewpoint. But most of the comments that follow are addressed to a potential base-file designer.

Page C266, line 16 (8/17/86)

variables; they have the side effect of changing the variable's value.

Page C276, line 26 (6/23/86)

```
if charic<>0: r((w+charic*hppp,h.o_),(w+charic*hppp,.5h.o_)); fi
```

Page C286, lines 24–26 (10/13/86)

but METAFONT won't let you. And even if this had worked, it wouldn't have solved the problem; it would simply have put **ENDFOR** into the replacement text of **ast**, because expansion is inhibited when the replacement text is being read.

Page C290, line 1 (8/23/86)

2. Fortuitous loops. The 'max' and 'min' macros in Appendix B make use of the fact

Page C298, third-last line (8/23/86)

$$t[u_1,\ldots,u_n] = t[t[u_1,\ldots,u_{n-1}],t[u_2,\ldots,u_n]]$$

Page C304, 14th-last line (2/15/87)

[replace this '`\smallskip`' by a `\smallskip` between lines!]

Page C307, fifth-last line (12/7/86)

adjust_fit(⟨left sidebearing adjustment⟩, ⟨right sidebearing adjustment⟩);

Page C312, line 34 (10/12/86)

```
params[2] = "sans_params";      fontname[2] = "cmssbx10";
```

Page C316, lines 19–21 (8/17/86)

example, '`(some charht values had to be adjusted by as much as 0.12pt)`' means that you had too many different nonzero heights, but METAFONT found a way to reduce the number to at most 15 by changing some of them; none of them had to be

Page C319, line 3 (8/23/86)

specified by saying, e.g.,

Page C321, line 6 (7/28/86)

```
special "identifier " & font_identifier_;
```

Page C331, just below the illustration (7/18/87)

Such a pattern is, of course, rather unlikely to occur in a **gf** file, but **GFtoDVI** would

Page C334, line 2 (6/23/86)

```
currentpicture := currentpicture shifted-(1,1); pix := currentpicture;
```

Page C339, tenth-last line (2/4/87)

```
Jackie K\=aren {\L}au\.ra Mar{\'\i}a N\H{a}ta{\l}{\u\i}e {\O}ctave
```

Page C343, second-last line (8/23/86)

the precise needs of a precise but limited intellectual goal.

Page C346, 2nd line of entry for ';' (1/12/87)

217, 223–224, 263, 312.

Page C348, line 6 (6/17/86)

concatenation, of paths, *70–71*, *123*, 127,

Page C348, just before 'debugging' (3/16/87)

de Casteljau, Paul de Faget, 14.

Page C348, right column (3/16/87)

[The entry for '`define_whole_vertical_blacker_pixels`' should be moved up before the entry for '`define_whole_vertical_pixels`'.]

Page C352, left column (6/1/87)

`*kern`, *97*, *316*, 317.

Page C352, right column (3/8/87)

[The entry for '`lowres`' belongs before the entry for '`lowres_fix`'.]

Page C353, left column (3/8/87)

[The entries for '`mode`' and '⟨mode command⟩' belong before the entry for '`mode_def`'.]

Page C353, entry for `mode_def` (8/17/86)

`mode_def`, 94, 189, *270*, *278–279*.

Page C355, right column (4/15/86)

[The entry for '`rulepen`' belongs before the entry for 'rules'.]

Page C355, right column (8/5/86)

`screenstrokes`, 191, 277.

Page C355, 2nd line of entry for 'semicolons' (1/12/87)

217, 223–224, 263, 312.

Page C356, left column (1/11/88)

SIMULA67 language, 175.

Page C356, full names for the Stanfords (4/10/86)

Stanford, Amasa Leland, 340.
Stanford, Jane Elizabeth Lathrop, 340.

Page C358, right column (2/15/88)

`*yoffset`, 212, 220, 315, 324.

Volume D, in general (7/28/86)

[A number of entries were mistakenly omitted from the mini-indexes on the right-hand pages. Here is a combined list of all the missing items; you can mount it inside the back cover, say, as a secondary mini-index when the first one fails...]

add_or_subtract: **procedure**, §930.
after: **array**, §427.
arg_list: *pointer*, §720.
b: *pixel_color*, §580.
bad_exp: **procedure**, §824.
before: **array**, §427.
begin_name: **procedure**, §770.
bilin1: **procedure**, §968.
binary_mac: **procedure**, §863.
blank_rectangle: **procedure**, §567.
boc_c: *integer*, §1162.
boc_p: *integer*, §1162.
cf: *fraction*, §298.
clockwise: *boolean*, §453.
ct: *fraction*, §298.
cubic_intersection: **procedure**, §556.
cur_pen: *pointer*, §403.
cur_rounding_ptr: 0 .. *max_wiggle*, §427.
cur_spec: *pointer*, §403.
cur_x: *scaled*, §389.
cur_y: *scaled*, §389.
dely: *integer*, §557.
dep_finish: **procedure**, §935.
dep_list = macro, §587.
dimen_head: **array**, §1125.
dx: *integer*, §495.
dy: *integer*, §495.
d1: 0 .. 1, §464.
end_name: **procedure**, §772.
eqtb: **array**, §201.
error_stop_mode = 3, §68.
firm_up_the_line: **procedure**, §682.
get_next: **procedure**, §667.
gf_buf: **array**, §1152.
gf_offset: *integer*, §1152.
gf_ptr: *gf_index*, §1152.
halfword = *min_halfword* .. *max_halfword*, §156.
hash: **array**, §201.
index = macro, §629.
input_ln: **function**, §30.
interaction: 0 .. 3, §68.
j: 0 .. *move_size*, §357.
known_pair: **procedure**, §872.
limit = macro, §629.
m_spread: *integer*, §357.
materialize_pen: **procedure**, §865.
max_allowed: *scaled*, §403.
max_c: **array**, §813.
max_link: **array**, §813.
max_tfm_dimen: *scaled*, §1130.
mem_top = macro, §12.
mem: **array**, §159.
memory_word = **record**, §156.
more_name: **function**, §771.
m1: *integer*, §464.
n: *screen_col*, §580.
n_sin_cos: **procedure**, §145.
name = macro, §629.
negate_dep_list: **procedure**, §904.
new_knot: **function**, §871.
node_to_round: **array**, §427.
n1: *integer*, §464.
octant_dir: **array**, §395.
o1: *small_number*, §453.
o2: *small_number*, §453.
paint_row: **procedure**, §568.
param: **array**, §1096.
param_stack: **array**, §633.
path_length: **function**, §916.
perturbation: *scaled*, §1119.
phi: *angle*, §542.
pool_ptr: *pool_pointer*, §38.
post_head: *pointer*, §843.
pre_head: *pointer*, §843.
print_err = macro, §68.
print_macro_name: **procedure**, §722.
quarterword = 0 .. 255, §156.
recycle_value: **procedure**, §809.
row_transition: *trans_spec*, §579.
scan_text_arg: **procedure**, §730.
scroll_mode = 2, §68.
set_controls: **procedure**, §299.
sf: *fraction*, §298.
show_context: **procedure**, §635.
sorted = macro, §325.
st: *fraction*, §298.
start = macro, §629.
start_sym: *halfword*, §1077.
str_pool: **packed array**, §38.
str_ptr: *str_number*, §38.
str_start: **array**, §38.
take_part: **procedure**, §910.
tfm_changed: *integer*, §1130.
tol: *integer*, §557.
tt: *small_number*, §843.
tx: *scaled*, §954.
txx: *scaled*, §954.
txy: *scaled*, §954.
ty: *scaled*, §954.
tyx: *scaled*, §954.
tyy: *scaled*, §954.
unsorted = macro, §325.
uv: 0 .. *bistack_size*, §557.
xy: 0 .. *bistack_size*, §557.
x1: *scaled*, §542.
x2: *scaled*, §542.
x3: *scaled*, §542.
y1: *scaled*, §542.
y2: *scaled*, §542.
y3: *scaled*, §542.

Volume D, in general (4/6/87)

[The percent signs in all the comments (for example, on pages 7 and 42) are in the wrong font! Change '%' to '%'.]

Page Dvii, line 9 (9/25/86)

Discrete and Computational Geometry **1** (1986), 123–140. *Develops the theory*

Page D2, line 27 (6/17/86)

define *banner* ≡ `'This␣is␣METAFONT,␣Version␣1.3'` { printed when METAFONT starts }

Page D18, line 30 (5/22/86)

str_ptr: *str_number*; { number of the current string being created }

Page D23, second line of mini-index, right column (6/14/87)

pool_name
= "string", §11.

Page D30, lines 33–34 (6/14/87)

to delete a token, and/or if some fatal error occurs while METAFONT is trying to fix a non-fatal one. But such recursion is never more than two levels deep.

Page D63, lines 13–14 (5/5/87)

[These two lines can be eliminated, since the variable *temp_ptr* is no longer used! If you delete them, also remove §158 from the list of sections where global variables are declared (pages D7 and D552), and remove *temp_ptr* from the index on page D540.]

Page D66, line 6 (5/22/86)

function *get_node*(*s* : *integer*): *pointer*; { variable-size node allocation }

Page D66, lines 31–32 (3/16/86)

controlled growth helps to keep the *mem* usage consecutive when METAFONT is implemented on "virtual memory" systems.

Page D67, lines 7–8 (4/21/87)

if $r = p$ **then if** *rlink*$(p) \neq p$ **then** ⟨ Allocate entire node p and **goto** *found* 171 ⟩;

Page D86, second line of section 128 (2/27/87)

Individual class numbers have no semantic or syntactic significance, except in a few instances

Page D101, line 2 (3/16/86)

like '`x`', or they can combine the structural properties of arrays and records, like '`x20a.b`'. A

Page D102, line 24 (3/16/86)

In other words, variables have a hierarchical structure that includes enough threads running

Page D127, line 10 (5/5/87)

[Variable r can be eliminated, since it is not used in this procedure! If you delete it, also remove 280 from the corresponding index entry on page D536.]

Page D129, line 15 (5/5/87)

[This line can be eliminated, since *sine* and *cosine* are not used in this procedure! If you delete them, also remove 284 from the corresponding index entries on pages D538 and D521.]

Page D142, line 23 (4/24/87)

$(7 - \sqrt{28})/12$; the worst case occurs for polynomials like $B(0, 28 - 4\sqrt{28}, 14 - 5\sqrt{28}, 42; t)$.)

Page D178, third-last line (7/30/86)

The following code maintains the invariant relations $0 \leq x0 < \max(x1, x1 + x2)$, $|x1| < 2^{30}$,

Page D228, line 13 (7/30/86)

while *max_coef* < *fraction_half* **do**

The mini-index at the bottom of the next page should also receive the following new entry:

fraction_half = macro, §105.

Page D228, 10th-last line (5/5/87)

begin *right_type*(*p*) ← *k*;

[Also eliminate '*q*,' seven lines above this, and delete 497 from the index entry for *q* on page D536.]

Page D248, lines 16–21 (11/27/86)

```
alpha ← abs(u); beta ← abs(v);
if alpha < beta then
  begin alpha ← abs(v); beta ← abs(u); end;  { now α = max(|u|,|v|), β = min(|u|,|v|) }
if internal[fillin] ≠ 0 then
  d ← d − take_fraction(internal[fillin], make_fraction(beta + beta, delta));
d ← take_fraction((d + 4) div 8, delta); alpha ← alpha div half_unit;
```

Page D263, line 20 (3/16/86)

instead of *false*, the other routines will simply log the fact that they have been called; they won't

Page D268, line 2 (4/28/87)

Given the number *k* of an open window, the pixels of positive weight in *cur_edges* will be shown

Page D301, line 6 of section 652 (5/5/87)

[This line can be eliminated, since variable *s* is not used in this procedure! If you delete it, also remove 652 from the corresponding index entry on page D537; remove 652 from the index entries for *param_size* and *param_start* on page D534; and remove *param_size* from the mini-index on page D301.]

Page D376, lines 17 and 18 (11/14/86)

[these two mysterious lines should be deleted]

Page D380, line 11 (5/5/87)

[Variables *q* and *r* can be eliminated, since they are not used in this procedure! If you delete them, also remove 862 from the corresponding index entries on page D536.]

Page D429, line 14 (5/5/87)

begin *p* ← *cur_exp*;

[Also eliminate line 12, and delete 985 from the index entry for *vv* on page D543.]

Page D455, line 5 (5/5/87)

[This line can be eliminated, since variable *t* is not used in this procedure! If you delete it, also remove 1059 from the corresponding index entry on page D540; remove 1059 from the index entries for *small_number* and *with_option* on pages D539 and D544; and remove *with_option* from the mini-index on page D455.]

Page D463, line 10 (12/15/86)

```
("Pretend␣that␣you're␣Miss␣Marple:␣Examine␣all␣clues,")
```

Page D465, lines 17–18 (6/14/87)

[Delete these two lines.]

Page D474, 5th-last line (3/16/86)

depths, or italic corrections) are sorted; then the list of sorted values is perturbed, if necessary.

Page D481, line 12 (6/17/86)

print_nl("Font␣metrics␣written␣on␣"); *print*(*metric_file_name*); *print_char*(".");
b_close(*tfm_file*)

The mini-index at the bottom of this page should also receive the following new entry:

print_char: **procedure**, §58.

Page D510, new line to follow line 5 (6/17/86)

This program doesn't bother to close the input files that may still be open.

Page D510, just before the fifth-last line (8/5/86)

internal[*fontmaking*] ← 0; { avoid loop in case of fatal error }

Page D520, right column (6/14/87)

Chinese characters: 1147.

Page D526, left column, lines 1–2 (7/30/86)

fraction_half: 105, 111, 152, 288, 408, 496, 543,
1098, 1128, 1141.

Page D526, left column, lines 6–7 (7/30/86)

478, 497, 499, 503, 530, 540, 547, 549, 599, 603,
612, 615, 815–816, 917, 1169–1170.

Page D528, right column (6/14/87)

Japanese characters: 1147.

Page D530, right column, line 45 (7/30/86)

max: 539, 543.

Page D533, right column (6/14/87)

oriental characters: 1147.

Page D535, right column, line 27 (6/17/86)

1134, 1163–1165, 1182, 1194, 1200, 1205, 1213.

Page D545, left column (10/31/87)

zscaled primitive: 893.
Zabala Salelles, Ignacio Andres: 812.

Page D547, bottom two lines (11/27/86)

[These lines, and the top two on the next page, should move down so that they appear in alphabetical order just before 'Compute test coefficients'.]

Page Exiii, lines 1–2 (7/28/86)

February 11–13, 1984), 49. *An example meta-character of the Devanagari alphabet, worked out "online" with the help of Matthew Carter.*

Page Exiii, line 6 (7/28/86)

and western alphabets work also for Devanagari and Tamil.

Page E12, lines 15 and 19 (7/23/86)

[change '17.32' to '17.28' in both places]

Page E12, third-last line (12/18/86)

[change '41' to '40']

Page E13, lines 3, 4, and 20 (12/18/86)

[change '40' to '41', '48' to '47', '17' to '7']

Page E18, line 20 (7/23/86)

[change '17.32' to '17.28']

Page E18, line 29 (12/9/86)

[change '236' to '212' in the `cmss9` column]

Page E32, second-last line (9/20/87)

after which comes '*math_axis#*; **generate** `mathsy`' (which we won't bother to

Page E170, top illustration (11/2/86)

[There should be no "dish" or depression in the vicinity of point `3r`; the top edge of the character should be straight. This error appears also in the other uses of '*no_dish_serif*' throughout the book, since the illustrations were made before '*no_dish_serif*' was added to the program. See page E180 (twice at the top), E370 (twice), E374 (twice), E376 (twice), E378 (top), E390 (bottom), E398 (top), E402 (top), E406 (top), E453 (twice).]

Page E179, new line to be inserted after line 6 (10/13/86)

if *shaved_stem* < *crisp.breadth*: *shaved_stem* := *crisp.breadth*; **fi**

Page E219, line 29 (6/2/87)

top $y_1 = h$; $x_1 = x_2$; **filldraw stroke** z_{1e} -- $z_{2'e}$; % stem

Page E279, seventh line from the bottom (7/20/86)

that delicious but restrained humor which her readers found so irresistible.

Page E285, bottom line (12/1/87)

Due to Technical Developments (1968)

Page E301, new line to be inserted after line 28 (5/15/87)

if *lower_side* $>$ 1.2*upper_side*: *upper_side* := *lower_side*; **fi**

Page E353, lines 38–39 (8/12/87)

else: **fill** *diag_end*$(6r, 5r, 1, 1, 5l, 6l)$ -- .9$[z_{5l}, z_{6l}]$
..$\{z_5 - z_6\}$.1$[z_{5r}, z_{6r}]$ -- cycle; % middle stem

Page E387, line 13 (8/12/87)

pickup *tiny.nib*; *bulb*$(3, 4, 5)$; % bulb

Page E413, lines 37–38 (8/12/87)

else: **fill** *diag_end*$(6r, 5r, 1, 1, 5l, 6l)$ -- .9$[z_{5l}, z_{6l}]$
..$\{z_5 - z_6\}$.1$[z_{5r}, z_{6r}]$ -- cycle; % middle stem

Page E459, line 24 (8/7/87)

[Delete the '=' sign between '*lft*' and 'x_5'.]

Page E485, line 4 (8/7/87)

[Delete the '=' sign between '*lft*' and 'x_5'.]

Page E550, new line after line 23 (8/15/87)

forsuffixes $ = *notch_cut*, *cap_notch_cut*: **if** $ < 3: $:= 3; **fi endfor**

[To make room for this, combine lines 38 and 39 into a single line.]

Page E554, bottom half of page (12/18/86)

[The letters will change slightly because of the corrections to `cmr17` noted on pages 12 and 13.]

Page E561, line 3 (12/9/86)

[The numerals should be '`0123456789`' (i.e., 2/3 point less tall) because of the correction made to page 18.]

Page E562, line 9 (12/9/86)

[The numerals should be '*0123456789*' (i.e., 2/3 point less tall) because of the correction made to page 18.]

Page E572, entry for *breadth* (10/13/86)

Page E573, entry for `cmcsc10` (8/17/86)

Page E576, tenth-last line (5/15/87)

Changes to the Programs and Fonts

15 February 1988

- **TEX**

Changes subsequent to 'Version 2.0' as published in C&T, Volume B, incorporating changes through 23 December 1987:

```
322. Trivial change to a help message.
@x module 1283
("Pretend that you're Hercule Poirot, examine all clues,")@/
@y
("Pretend that you're Hercule Poirot: Examine all clues,")@/
@z

323. Precaution since |index| can be as high as |max_in_open|. (DRF)
@x module 14
if hash_prime>hash_size then bad:=5;
@y
if hash_prime>hash_size then bad:=5;
if max_in_open>=128 then bad:=6;
@z

324. \kern clobbered at end of pre-break list in discretionary break.
@x module 881
|disc_break:=true|@>;
if not is_char_node(q) then
if (type(q)=math_node)or(type(q)=kern_node) then width(q):=0;
@y
|disc_break:=true|@>
else if (type(q)=math_node)or(type(q)=kern_node) then width(q):=0;
@z

325. width after discretionary didn't take account of discarded nodes.
@x module 830 (slight redefinition)
@!t:quarterword; {replacement count, if |cur_p| is a discretionary node}
@y
@!t:integer; {node count, if |cur_p| is a discretionary node}
@z
@x module 837 (slight rearrangement)
if (break_type=unhyphenated)or(cur_p=null) then
begin s:=cur_p;
while s<>null do
begin if is_char_node(s) then goto done;
case type(s) of
glue_node:@<Subtract glue from |break_width|@>;
penalty_node: do_nothing;
math_node,kern_node: if subtype(s)=acc_kern then goto done
else break_width[1]:=break_width[1]-width(s);
othercases goto done
endcases;@/
s:=link(s);
end;
end
else @<Compute the discretionary |break_width| values@>;
```

```
@y
s:=cur_p;
if break_type>unhyphenated then if cur_p<>null then
@<Compute the discretionary |break_width| values@>;
while s<>null do
begin if is_char_node(s) then goto done;
case type(s) of
glue_node:@<Subtract glue from |break_width|@>;
penalty_node: do_nothing;
math_node,kern_node: if subtype(s)=acc_kern then goto done
else break_width[1]:=break_width[1]-width(s);
othercases goto done
endcases;@/
s:=link(s);
end;
@z
@x module 840 (this is the main change to fix the bug)
begin t:=replace_count(cur_p); s:=cur_p;
while t>0 do
begin decr(t); s:=link(s);
@<Subtract the width of node |s| from |break_width|@>;
end;
s:=post_break(cur_p);
while s<>null do
begin @<Add the width of node |s| to |break_width|@>;
@y
begin t:=replace_count(cur_p); v:=cur_p; s:=post_break(cur_p);
while t>0 do
begin decr(t); v:=link(v);
@<Subtract the width of node |v| from |break_width|@>;
end;
while s<>null do
begin @<Add the width of node |s| to |break_width| and increase |t|,
unless it's discardable@>;
@z
@x module 840, continued
break_width[1]:=break_width[1]+disc_width;
@y
break_width[1]:=break_width[1]+disc_width;
if t=0 then s:=link(v); {more nodes may also be discardable after the break}
@z
```

```
@x module 841 (here we just change |s| to |v|)
@<Subtract the width of node |s|...@>=
if is_char_node(s) then
begin f:=font(s);
break_width[1]:=break_width[1]-char_width(f)(char_info(f)(character(s)));
end
else case type(s) of
ligature_node: begin f:=font(lig_char(s));@/
break_width[1]:=@|break_width[1]-
char_width(f)(char_info(f)(character(lig_char(s))));
end;
hlist_node,vlist_node,rule_node,kern_node:
break_width[1]:=break_width[1]-width(s);
@y
@<Subtract the width of node |v|...@>=
if is_char_node(v) then
begin f:=font(v);
break_width[1]:=break_width[1]-char_width(f)(char_info(f)(character(v)));
end
else case type(v) of
ligature_node: begin f:=font(lig_char(v));@/
break_width[1]:=@|break_width[1]-
char_width(f)(char_info(f)(character(lig_char(v))));
end;
hlist_node,vlist_node,rule_node,kern_node:
break_width[1]:=break_width[1]-width(v);
@z
@x module 842
hlist_node,vlist_node,rule_node,kern_node:
break_width[1]:=break_width[1]+width(s);
othercases confusion("disc2")
@:this can't happen disc2}{\quad disc2@>
endcases
@y
hlist_node,vlist_node,rule_node:break_width[1]:=break_width[1]+width(s);
kern_node: if (t=0)and(subtype(s)<>acc_kern) then t:=-1 {discardable}
else break_width[1]:=break_width[1]+width(s);
othercases confusion("disc2")
@:this can't happen disc2}{\quad disc2@>
endcases;
incr(t)
@z

*** Version 2.1 was released on January 26, 1987.


326. Removal of redundant code (found by Pat Monardo, 5 Feb 87)
@x module 1224 [this change need not be made, since it has no effect]
char_def_code: define(p,char_given,cur_val);
math_char_def_code: define(p,math_given,cur_val);
@y the cases cannot occur, so we simply don't list them
@z
```

```
327. More robustness is needed when debugging (Ronaldo Am\'a, 14 Apr 87)
@x module 174
begin while p>null do
@y maybe mem_min>null (but nodes shouldn't be put in location mem_min exactly)
begin while p>mem_min do
@z
@x module 182
@p procedure show_node_list(@!p:pointer); {prints a node list symbolically}
@y
@p procedure show_node_list(@!p:integer); {prints a node list symbolically}
@z
@x module 182, continued
while p>null do
@y
while p>mem_min do
@z

328. Storage allocation can be more elegant and efficient (4/21/87)
@x module 127
if r=p then if ((rlink(p)<>rover) or (llink(p)<>rover)) then
@y
if r=p then if rlink(p)<>p then
@z

329. Miscalculation of empty-line condition (April 22, 1987)
@x module 360
begin if limit=start then {previous line was empty}
@y
begin if (end_line_char<0)or(end_line_char>127) then incr(limit);
if limit=start then {previous line was empty}
@z

330. A case where we don't have to assume system bookkeeping (April 28, 1987)
@x module 560
done: b_close(tfm_file); read_font_info:=g;
@y [suggested by Jim Sterken]
done: if file_opened then b_close(tfm_file);
read_font_info:=g;
@z [this was not a bug, according to the comment in module 28]

331. jump_out must fix unfinished output (Found by Klaus Gunterman, 3 Aug 87)
@x module 593
doing_leaders:=false; dead_cycles:=0;
@y
doing_leaders:=false; dead_cycles:=0; cur_s:=-1;
@z
@x module 617
cur_s:=-1; ensure_dvi_open;
@y
ensure_dvi_open;
@z
```

```
@x module 640
dvi_out(eop); incr(total_pages);
@y
dvi_out(eop); incr(total_pages); cur_s:=-1;
@z
@x module 642
if total_pages=0 then print_nl("No pages of output.")
@y
while cur_s>-1 do
  begin if cur_s>0 then dvi_out(pop)
  else  begin dvi_out(eop); incr(total_pages);
    end;
  decr(cur_s);
  end;
if total_pages=0 then print_nl("No pages of output.")
@z
```

332. \hangindent=1pt$$\halign{...\cr\noalign{\hrule}}$$ problem (19 Aug 87)

```
@x module 805
q:=link(head);
@y
q:=link(head); s:=head;
@z
@x module 805, continued
  q:=link(q);
@y
  s:=q; q:=link(q);
@z
@x module 806
if is_running(depth(q)) then depth(q):=depth(p);
@y
if is_running(depth(q)) then depth(q):=depth(p);
if o<>0 then
  begin r:=link(q); link(q):=null; q:=hpack(q,natural);
  shift_amount(q):=o; link(q):=r; link(s):=q;
  end;
@z
```

333. \hskip 0pt plus 1fil\ifdim problem (found by Alan Guth, 20 Aug 87)

```
@x module 366
@!cvl_backup,@!radix_backup:small_number; {to save |cur_val_level| and |radix|}
@y
@!cvl_backup,@!radix_backup,@!co_backup:small_number;
  {to save |cur_val_level|, etc.}
@z
@x
backup_backup:=link(backup_head);
@y
co_backup:=cur_order; backup_backup:=link(backup_head);
@z
@x
link(backup_head):=backup_backup;
@y
cur_order:=co_backup; link(backup_head):=backup_backup;
@z
```

```
334. leaders too sensitive near exact multiples (M. F. Bridgland, 9 Nov 87)
@x module 626
  begin edge:=cur_h+rule_wd; lx:=0;
@y
  begin rule_wd:=rule_wd+10; {compensate for floating-point rounding}
  edge:=cur_h+rule_wd; lx:=0;
@z
@x ibid
  cur_h:=edge; goto next_p;
@y
  cur_h:=edge-10; goto next_p;
@z
@x module 635
  begin edge:=cur_v+rule_ht; lx:=0;
@y
  begin rule_ht:=rule_ht+10; {compensate for floating-point rounding}
  edge:=cur_v+rule_ht; lx:=0;
@z
@x ibid
  cur_v:=edge; goto next_p;
@y
  cur_v:=edge-10; goto next_p;
@z

335. Glitch in fixed-point multiplication of negatives (W.G. Sullivan, 17Nov87)
@x module 572
begin alpha:=16*z; beta:=16;
while z>=@'40000000 do
  begin z:=z div 2; beta:=beta div 2;
  end;
@y
begin alpha:=16;
while z>=@'40000000 do
  begin z:=z div 2; alpha:=alpha+alpha;
  end;
beta:=256 div alpha; alpha:=alpha*z;
@z

336. If there are no \patterns and some \lccode is 1 (Breitenlohner, 12Dec87)
@x module 952
trie_link(0):=0; trie_char(0):=0; trie_op(0):=0;
@y
trie_link(0):=0; trie_char(0):=0; trie_op(0):=min_quarterword;
@z

337. \csname might encounter undefined_cs in a group (Chris Thompson, 23Dec87)
@x module 372
  begin eqtb[cur_cs]:=eqtb[frozen_relax];
@y
  begin eq_define(cur_cs,relax,256);
@z

338. (I sincerely hope that there won't be any more)
```

- METAFONT

Changes subsequent to 'Version 1.0' as published in C&T, Volume D, incorporating changes through 5 May 1987:

```
533. Inconsistent punctuation in user messages (found by Karl Berry, June 86)
@x module 1134
print_nl("Font metrics written on "); print(metric_file_name);
@y
print_nl("Font metrics written on "); print(metric_file_name); print_char(".");
@z (that was installed in version 1.1)

534. Possible arithmetic overflows (found by Klaus Guntermann, July 86)
@x module 496
while max_coef<fraction_one do
@y
while max_coef<fraction_half do
@z (that was installed in version 1.2)

535. Possible loop in nonstopmode (found by Chris Thompson, July 86)
@x module 1206
    @<Finish the \.{TFM} file@>;
@y
    internal[fontmaking]:=0; {avoid loop in case of fatal error}
    @<Finish the \.{TFM} file@>;
@z (that too was installed in version 1.2)

536. Double rounding error should be avoided in make_ellipse (JDH, 22 Nov 86)
@x module 533
d:=take_fraction(d,delta);
alpha:=abs(u); beta:=abs(v);
if alpha<beta then
  begin delta:=alpha; alpha:=beta; beta:=delta;
  end; {now $\alpha=\max(\vert u\vert,\vert v\vert)$,
      $\beta=\min(\vert u\vert,\vert v\vert)$}
if internal[fillin]<>0 then d:=d-take_fraction(internal[fillin],beta+beta);
d:=(d+4) div 8; alpha:=alpha div half_unit;
@y
alpha:=abs(u); beta:=abs(v);
if alpha<beta then
  begin alpha:=abs(v); beta:=abs(u);
  end; {now $\alpha=\max(\vert u\vert,\vert v\vert)$,
      $\beta=\min(\vert u\vert,\vert v\vert)$}
if internal[fillin]<>0 then
  d:=d-take_fraction(internal[fillin],make_fraction(beta+beta,delta));
d:=take_fraction((d+4) div 8,delta); alpha:=alpha div half_unit;
@z (That was the reason for version 1.3)
```

```
537. Trivial change to a help message (version number is still 1.3).
@x module 1086
  ("Pretend that you're Miss Marple, examine all clues,")@/
@y
  ("Pretend that you're Miss Marple: Examine all clues,")@/
@z

538. Storage allocation can be more elegant and efficient (4/21/87)
@x module 169
if r=p then if ((rlink(p)<>rover) or (llink(p)<>rover)) then
@y
if r=p then if rlink(p)<>p then
@z

539. Unused variables can be eliminated. (Found by John Sauter, 5/5/87)
@x module 158
@<Glob...@>=
@!temp_ptr:pointer; {a pointer variable for occasional emergency use}

@y (I used it only in my change file!)
@z
@x module 280
@!r,@!s,@!t:pointer; {registers for list traversal}
@y
@!s,@!t:pointer; {registers for list traversal}
@z
@x module 284
@!sine,@!cosine:fraction; {trig functions of various angles}
@y (the declarations in module 281 are correct, but in 284 they're superfluous)
@z
@x module 497
var @!q,@!ww:pointer; {for list manipulation}
@!du,@!dv:scaled; {for slope calculation}
@!t0,@!t1,@!t2:integer; {test coefficients}
@!t:fraction; {place where the derivative passes a critical slope}
@!s:fraction; {slope or reciprocal slope}
@!v:integer; {intermediate value for updating |x0..y2|}
begin loop
  begin q:=link(p); right_type(p):=k;
@y
var @!ww:pointer; {for list manipulation}
@!du,@!dv:scaled; {for slope calculation}
@!t0,@!t1,@!t2:integer; {test coefficients}
@!t:fraction; {place where the derivative passes a critical slope}
@!s:fraction; {slope or reciprocal slope}
@!v:integer; {intermediate value for updating |x0..y2|}
begin loop
  begin right_type(p):=k;
@z
```

```
@x module 652
@!s:0..param_size; {value of |param_start| on the current level}
@y who knows why that line was there?
@z
@x module 862
var @!p,@!q,@!r:pointer; {for list manipulation}
@y
var @!p:pointer; {for list manipulation}
@z
@x module 985
@!vv:scaled; {initial value of |v|}
@!q:pointer; {successor of |p|}
begin vv:=v; p:=cur_exp;@/
@y
@!q:pointer; {successor of |p|}
begin p:=cur_exp;@/
@z
@x module 1059
@!t:small_number; {variant of |with_option|}
@y
@z
```

540. (I sincerely hope that there won't be any more)

- **Computer Modern fonts**

  Changes subsequent to 'Version 2' as published in C&T, Volume E, incorporating changes through 15 August 1987:

```
@x in GREEKU
numeric shaved_stem; shaved_stem=hround .9[vair,.85cap_stem];
@y
numeric shaved_stem; shaved_stem=hround .9[vair,.85cap_stem];
if shaved_stem<crisp.breadth: shaved_stem:=crisp.breadth; fi
@z

@x in CMSS9 [this affects the TFM file to a small extent!]
fig_height#:=236/36pt#; % height of numerals
@y
fig_height#:=212/36pt#; % height of numerals
@z

@x in CMSSI9 [this affects the TFM file in ten hts and eleven italcorrs]
fig_height#:=236/36pt#; % height of numerals
@y
fig_height#:=212/36pt#; % height of numerals
@z

@x in CMR17 [no change to TFM file]
curve#:=41/36pt#; % lowercase curve breadth
@y
curve#:=40/36pt#; % lowercase curve breadth
@z
@x
cap_stem#:=40/36pt#; % uppercase stem breadth
cap_curve#:=48/36pt#; % uppercase curve breadth
@y
cap_stem#:=41/36pt#; % uppercase stem breadth
cap_curve#:=47/36pt#; % uppercase curve breadth
@z
@x
serif_drop#:=17/36pt#; % vertical drop of sloped serifs
@y
serif_drop#:=7/36pt#; % vertical drop of sloped serifs
@z

@x in ROMAND [this fixes the 'disappearing hairline' in some lowres 8's]
 lower_side=hround(.5[hair,stem]+stem_corr);
@y
 lower_side=hround(.5[hair,stem]+stem_corr);
 if lower_side>1.2upper_side: upper_side:=lower_side; fi
@z
% Note: the SAME change should also be made in files ITALD and OLDDIG

@x in ITALL [this fixes italic ell especially at low resolutions]
top y1=h; x1=x2; filldraw stroke z1e--z2e; % stem
@y
top y1=h; x1=x2; filldraw stroke z1e--z2'e; % stem
@z
```

```
@x in SYM, the plus-or-minus character
x1=x2=.5w; lft x3=lft=x5=hround u-eps; x4=x6=w-x3;
@y
x1=x2=.5w; lft x3=lft x5=hround u-eps; x4=x6=w-x3;
@z actually the code worked but it was "infelicitous"

@x in SYMBOL, the minus-or-plus character
x1=x2=.5w; lft x3=lft=x5=hround u-eps; x4=x6=w-x3;
@y
x1=x2=.5w; lft x3=lft x5=hround u-eps; x4=x6=w-x3;
@z actually the code worked but it was "infelicitous"


@x in ROMANU, letter J [fixes a bug if dish=0 and crisp<tiny and serifs]
 bulb(3,4,5); % bulb
@y
 pickup tiny.nib; bulb(3,4,5); % bulb
@z

@x in ROMANL, letter w [makes notch_cut more useful]
 else: fill diag_end(6r,5r,1,1,5l,6l)--.5[z5l,z6l]
   --.5[z5r,z6r]--cycle;% middle stem
@y
 else: fill diag_end(6r,5r,1,1,5l,6l)--.9[z5l,z6l]
   ..{z5-z6}.1[z5r,z6r]--cycle; % middle stem
@z the same change applies also to letter W in ROMANU

@x in CMBASE, makes lowres types (especially TT) look better
 define_blacker_pixels(notch_cut,cap_notch_cut);
@y
 define_blacker_pixels(notch_cut,cap_notch_cut);
 forsuffixes $=notch_cut,cap_notch_cut: if $<3: $:=3; fi endfor
@z
```

(I sincerely hope there won't be any more!)

Corrections to earlier editions

Corrections and changes to the AMS/Digital Press edition of the TEX and METAFONT manual (December 1979) and to the TEX78 and METAFONT79 programs are described in the booklet *TEX and METAFONT: Errata and Changes* dated September 1983 (originally distributed with TUGboat Volume 4, No. 2). This document also contains a comparison of TEX78 (formerly known as TEX80) with TEX82.

Errata to editions of *The TEXbook* published prior to 1986 are described in *The TEXbook: Errata and Changes* dated February 1986 (originally distributed with TUGboat Volume 7, No. 1).

Both of these documents are available from TUG; for information on how to obtain copies, write to TUG at the address on the front cover.

The TEX Tapes
Knuth's Course Available on Video

During Stanford's Spring Quarter, 1987, Donald Knuth presented a special course, TEX: The Program: A case study in software design. This course consisted of nineteen 75-minute lectures, which were videotaped by the Stanford Instructional Television Network (SITN) and transmitted to remote locations.

The implementation of TEX was discussed as an example of the design and documentation of a medium-size software system. Also discussed was the —WEB— system of structured documentation. With knowledge of the TEX and Pascal languages as a prerequisite, enough information about the innards of TEX was presented for students to learn to make extensions to the system.

TUG has permission to rent these tapes to TUG members, although SITN has stipulated that TUG may not rent them to profit-making organizations (who should contact SITN directly at Stanford University, Stanford, CA 94305).

The videotapes are available in VHS format (with sufficient demand, they may be made available also in Beta format). Due to anticipated scheduling requirements, the *normal* rental period will be six weeks. The rental fee for the 19 videotapes for that duration will be $800. (A longer rental period may be arranged and the rental fee will be prorated.) This includes all costs except shipping/handling/insurance charges for their return to TUG. Current TUG Institutional Members are entitled to a 20% discount.

Professor Knuth's book, *TEX: The Program*, the text for the course, may also be ordered from TUG. Since that book was meant to serve primarily as a reference, not as a text, Professor Knuth created a supplementary collection of exercises and problems applicable to the course. A copy of this problem set will accompany the tapes; it may be retained after the rental period. The problem set will also be made available separately by TUG.

To reserve these videotapes for rental, or to obtain additional information, please contact the TUG office.

TEX Users Group Membership List — 1987–88

As of 15 February 1988

This list includes the names of all persons who were members of TUG for 1987 or whose 1988 application forms were received by 15 February 1988. All institutional members are listed. Total membership: 136 institutional members and 3,244 individuals affiliated with more than 1,350 colleges and universities, commercial publishers, government agencies, and other organizations throughout the world having need for an advanced composition system. Additions to this list during 1988 will be published as supplements, included in the Summer and Fall issues of TUGboat.

The following information is included for each listing of an individual member, where it has been provided:

Name and mailing address

Telephone number

Network address

Title and organizational affiliation, when that is not obvious from the mailing address

Computer and typesetting equipment available to the member, or type of equipment on which his organization wishes to (or has) installed TEX

Uses to which TEX may be put, or a general indication of why the member is interested in TEX

CONTENTS

Recipients of this list are encouraged to use it to identify others with similar interests, and, as TUG members, to keep their own listings up-to-date in order for the list to remain as useful as possible. New or changed information may be submitted on the membership renewal form bound into the back of a recent issue of TUGboat. Comments on ways in which the content and presentation of the membership list can be improved are welcome.

This list is intended for the private use of TUG members; it is not to be used as a source of names to be included in mailing lists or for other purposes not approved by TUG. Additional copies are available from TUG. Mailing lists of current TUG membership are available for purchase. For more information, contact Ray Goucher, TUG Executive Director.

Distributed with TUGboat Volume 9 (1988), No. 1. Published by

TEX Users Group
P. O. Box 9506
Providence, R.I. 02940-9506, U.S.A.

Addresses of TUG Steering Committee Members and Members of Other TUG Committees

Site Coordinators

CDC Cyber

FOX, Jim
Academic Computing Center HG-45
University of Washington
3737 Brooklyn Ave NE
Seattle, WA 98105
206-543-4320
fox@uwavm.acs.washington.edu
Bitnet: fox7632@uwacdc

DG MV

CHILDS, S. Bart
Dept of Computer Science
Texas A & M University
College Station, TX 77843-3112
409-845-5470
Bitnet: Bart@TAMLSR
President; Finance Committee

IBM MVS

PLATT, Craig R.
Dept of Math & Astronomy
Machray Hall
Univ of Manitoba
Winnipeg R3T 2N2, Manitoba Can
204-474-9832
Bitnet: platt@uofmcc
CSnet: platt@uofm.cc.cdn

IBM VM/CMS

GUENTHER, Dean
Computer Service Center
Washington State University
Computer Science Building, Room 2144
Pullman WA 99164-1220
509-335-0411
Bitnet: Guenther@WSUVM1
Annual Meeting Program Coordinator

Prime 50 Series

CRAWFORD, John M.
Computing Services Center
College of Business
Ohio State University
1775 College Road
Columbus, OH 43210
614-292-1741
Crawford-J@Ohio-State.edu
Bitnet: TS0135@OHSTVMA
International Coordinator

"small" systems

CARNES, Lance
163 Linden Lane
Mill Valley, CA 94941
415-388-8853

UNIX

MacKAY, Pierre A.
Northwest Computer Support Group
University of Washington
Mail Stop DW-10
Seattle, WA 98195
206-543-6259
MacKay@June.CS.Washington.Edu

VAX (VMS)

SMITH, Barry
Kellerman & Smith
534 SW Third Avenue
Portland, OR 97204
503-222-4234; TLX 9102404397
Usenet: tektronix!reed!barry

Officers

CHILDS, S. Bart
President; Finance Committee;
DG MV Site Coordinator

THEDFORD, Rilla
Intergraph Corporation
MS HQ1006
One Madison Industrial Park
Huntsville, AL 35807
205-772-2440
Vice President; Finance Committee

HOENIG, Alan
17 Bay Avenue
Huntington, NY 11743
516-385-0736
Secretary; Finance Committee

NESS, David
TV Guide
Radnor, PA 19088
215-293-8860
Treasurer; Finance Committee

Other Steering Committee Members

BECK, Lawrence A.
Grumman Data Systems
R & D, MS D12-237
Woodbury, NY 11797
516-682-8478
X3V1.8 Coordinator

BEETON, Barbara
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500 x299
BNB@Seed.AMS.com
BNB@XX.LCS.MIT.edu
Editor, TUGboat

DYER, Allen R.
2922 Wyman Parkway
Baltimore, MD 21211
301-243-0008 or 243-7283

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

FURUTA, Richard
Department of Computer Science
University of Maryland
College Park, MD 20742
301-454-1461
furuta@mimsy.umd.edu
Laser-Lovers moderator;
Finance Committee

GOUCHER, Raymond
TEX Users Group
P.O. Box 9506
Providence, RI 02940-9506
401-272-9500 x232
reg@Seed.AMS.com
Executive Director

HENDERSON, Doug
Division of Library Automation
Univ of California, Berkeley
186 University Hall
Berkeley, CA 94720
415-642-9485
Bitnet: dlatex@ucbcmsa
Metafont Coordinator

ION, Patrick D.
Mathematical Reviews
416 Fourth Street
P.O. Box 8604
Ann Arbor, MI 48107
313-996-5273
ion@Seed.AMS.com

KNUTH, Donald E.
Department of Computer Science
Stanford University
Stanford, CA 94305
DEK@Sail.Stanford.edu

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

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

WHIDDEN, Samuel B.
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500
sbw@seed.ams.com
Finance Committee

ZAPF, Hermann
Seitersweg 35
D-6100 Darmstadt
Fed Rep Germany

Members of Other Committees

Annual Meeting Program Committee

GUENTHER, Dean R.
Bitnet: GUENTHER@WSUVM1
IBM VM/CMS Site
Coordinator; Annual Meeting
Program Coordinator

Committee on Corporate Relationships

FERGUSON, Michael J.
INRS - Télécommunications
Univ du Québec
3 Place du Commerce
Verdun (H3E 1H6), Québec Canada
514-765-7834
CSnet: mike%tel.inrs.cdn@ubc.csnet

SMITH, Barry

WHIDDEN, Samuel B.

Nominating Committee

FERGUSON, Michael J.

ION, Patrick D.

JACKSON, Calvin W., Jr.
1749 Micheltorena St
Los Angeles, CA 90026
818-356-6245 or 213-660-3257

Output Device Standards Committee

McGAFFEY, Robert W.
Martin Marietta Energy Systems, Inc.
Building 9104-2
P.O. Box Y
Oak Ridge, TN 37831
615-574-0618
McGaffey%ORN.MFEnet@nmfecc.arpa

1988 Scholarship Committee

HSIA, Doris T.
2630 Sierra Vista Ct
San Jose, CA 95116
415-723-8117

McPARTLAND-CONN, Marie
8 Burlington Rd
Billerica, MA 01821
617-466-4220

WITTBECKER, Alan
TEX Users Group
P.O. Box 9506
Providence, RI 02940
aew@Seed.AMS.com

TUGboat Editorial Committee

BEETON, Barbara
Editor

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

EPPSTEIN, Maureen
Administrative Publications
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-725-1717
as.mve@Forsythe.Stanford.Edu
Associate Editor for Applications

HOENIG, Alan
Associate Co-Editor, Typesetting on
Personal Computers

JÜRGENSEN, Helmut
Dept of Computer Science
Univ of Western Ontario
London N6A 5B7, Ontario, Canada
519-661-3560
Bitnet: A505@UWOCC1
UUCP: helmut@deepthot
Associate Editor for Software

MANN, Laurie D.
Stratus Computer
55 Fairbanks Blvd
Marlboro, MA 01752
617-460-2610
uucp: harvard!anvil!es!Mann
Associate Editor for Training issues

PFEFFER, Mitch
Suite 90
148 Harbor View South
Lawrence, NY 11559
516-239-4110
Associate Co-Editor, Typesetting on
Personal Computers

TOBIN, Georgia K.M.
The Metafoundry
OCLC Inc., MC 485
6565 Frantz Road
Dublin, OH 43017
614-764-6087
Associate Editor for Fonts

Addresses and telephone numbers of individuals serving in more than one capacity are listed only once. Unless indicated otherwise, network address are for the Internet.

TUG Institutional Members

Addison-Wesley Publishing Company
Reading, Massachusetts

The Aerospace Corporation
El Segundo, California

American Mathematical Society
Providence, Rhode Island

ArborText, Inc.
Ann Arbor, Michigan

ASCII Corporation
Tokyo, Japan

Aston University
Birmingham, England

Brookhaven National Laboratory
Upton, New York

California Institute of Technology
Pasadena, California

Calvin College
Grand Rapids, Michigan

Centre Inter-Régional de Calcul Électronique, CNRS
Orsay, France

City University of New York
New York, New York

College of St. Thomas
Computing Center
St. Paul, Minnesota

College of William & Mary
Department of Computer Science
Williamsburg, Virginia

Columbia University
Center for Computing Activities
New York, New York

COS Information
Montreal, P. Q., Canada

Data General Corporation
Westboro, Massachusetts

DECUS, L&T Special Interest Group
Marlboro, Massachusetts

Department of National Defence
Ottawa, Ontario, Canada

Digital Equipment Corporation
Nashua, New Hampshire

dit Company, Ltd.
Tokyo, Japan

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

General Motors Research Laboratories
Warren, Michigan

Geophysical Company of Norway A/S
Stavanger, Norway

Grinnell College
Computer Services
Grinnell, Iowa

Grumman Corporation
Bethpage, New York

GTE Laboratories
Waltham, Massachusetts

Hart Information Systems
Austin, Texas

Hartford Graduate Center
Hartford, Connecticut

Harvard University
Computer Services
Cambridge, Massachusetts

Hewlett-Packard Co.
Boise, Idaho

Hobart & William Smith Colleges
Geneva, New York

Humboldt State University
Arcata, California

Hutchinson Community College
Hutchinson, Kansas

IBM Corporation
Scientific Center
Palo Alto, California

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

Louisiana State University
Baton Rouge, Louisiana

Marquette University
Department of Mathematics, Statistics
and Computer Science
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

National Center for Atmospheric Research
Boulder, Colorado

National Institutes of Health
Bethesda, Maryland

National Research Council Canada
Computation Centre
Ottawa, Ontario, Canada

New Jersey Institute of Technology
Newark, New Jersey

New York University
Academic Computing Facility
New York, New York

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 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
Department of Computer Science
Stony Brook, New York

Stratus Computer, Inc.
Marlboro, Massachusetts

Syracuse University
Syracuse, New York

Talaris Systems, Inc.
San Diego, California

Texas A & M University
Computing Services Center
College Station, Texas

Texas A & M University
Department of Computer Science
College Station, Texas

TRW, Inc.
Redondo Beach, California

Tufts University
Medford, Massachusetts

TV Guide
Radnor, Pennsylvania

TYX Corporation
Reston, Virginia

UNI.C
Danmarks EDB-Center
Aarhus, Denmark

University College
Cork, Ireland

University of Alabama
Tuscaloosa, Alabama

University of British Columbia
Computing Centre
Vancouver, British Columbia, Canada

University of British Columbia
Mathematics Department
Vancouver, British Columbia, Canada

University of Calgary
Calgary, Alberta, Canada

University of California, Berkeley
Academic Computing Services
Berkeley, California

University of California, Berkeley
Computer Science Division
Berkeley, California

University of California, Irvine
Department of Mathematics
Irvine, California

University of California, Irvine
Information & Computer Science
Irvine, 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 Crete
Institute of Computer Science
Research Center
Heraklio, Crete, Greece

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 Vermont
Burlington, Vermont

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

Vereinigte Aluminium-Werke AG
Bonn, Federal Republic of Germany

Villanova University
Villanova, Pennsylvania

Vrije Universiteit
Amsterdam, The Netherlands

Washington State University
Pullman, Washington

Widener University
Computing Services
Chester, Pennsylvania

John Wiley & Sons, Incorporated
New York, New York

Worcester Polytechnic Institute
Worcester, Massachusetts

Yale University
Department of Computer Science
New Haven, Connecticut

(138 institutional members as of 28 Feb 1988)

TEX Consulting and Production Services

North America

ALDINE PRESS
12625 La Cresta Drive, Los Altos Hills, CA 94022; (415) 948-2149
Composition services for mathematical and technical books.

AMERICAN MATHEMATICAL SOCIETY
P. O. Box 6248, Providence, RI 02940; (401) 272-9500, ext. 224
Typesetting from DVI files on an Autologic APS Micro-5. Times Roman and Computer Modern fonts. Composition services for mathematical and technical books and journal production.

ARBORTEXT, Inc.
535 W. William, Suite 300, Ann Arbor, MI 48103; (313) 996-3566
Typesetting from DVI files on an Autologic APS-5. Computer Modern and standard Autologic fonts. TEX installation and applications support. TEX-related software products.

ARCHETYPE PUBLISHING, Inc., Lori McWilliam Pickert
P. O. Box 6567, Champaign, IL 61821; (217) 359-8178
Experienced in producing and editing technical journals with TEX; complete book production from manuscript to camera-ready copy; TEX macro writing including complete macro packages; consulting.

HOENIG, Alan
17 Bay Avenue, Huntington, NY 11743; (516) 385-0736
TEX typesetting services including complete book production; macro writing; individual and group TEX instruction.

KUMAR, Romesh
1028 Emerald, Naperville, IL 60540; (312) 972-4342
Beginners and intermediate group/individual instruction in TEX. Development of TEX macros for specific purposes. Using TEX with FORTRAN for custom-tailored software. Flexible hours, including evenings and weekends.

OGAWA, Arthur
920 Addison, Palo Alto, CA 94301; (415) 323-9624
Experienced in book production, macro packages, programming and consultation. Complete book production from computer-readable copy to camera-ready copy.

RICHERT, Norman
1614 Loch Lake Drive, El Lago, TX 77586; (713) 326-2583
TEX macro consulting.

TEXNOLOGY, Inc., Amy Hendrickson
57 Longwood Ave., Brookline, MA 02146; (617) 738-8029.
TEX macro writing (author of MacroTEX); custom macros written to meet publisher's or designer's specifications; instruction.

Outside North America

BAZARGAN, Kaveh
Optics Section, Blackett Laboratory, Imperial College of Science and Technology, London SW7 2BZ, U.K.; (01) 589 5111 ext. 6841
Instruction in TEX, for beginner and intermediate levels. Custom macros, including complex tables.

TEXPERT SYSTEMS Ltd., (Malcolm Clark and Cathy Booth)
5, Northernhay Square, Exeter, EX4 3ES, Devon, U.K.; (0392) 76091
The Complete TEX Service: consultation, all levels of document production, printing, customised style sheets (macros) to simplify text input, comprehensive training courses (will travel).

VARDAS
88, North Rd., St. Andrews, Bristol, BS6 5AJ, U.K.; (0272) 428709
TEX consultation, instruction, macro writing, book production, consulting.

Information about this service can be obtained from Ray Goucher, TEX Users Group, P. O. Box 9506, Providence, RI 02940, (401) 272-9500, ext. 232.

MONTRÉAL

The Meeting.
The 1988 TEX Users Group Annual Meeting will focus on TEX in a production environment. Papers and sessions will address a number of the aspects, approaches, problems, and solutions associated with using and teaching TEX and TEX-related programs. Nontechnical subjects and human factors will be considered, also (see the announcement on page 87 for a list of topics). Courses to be offered in conjunction with the meeting include Beginning TEX, Intensive Beginning/Intermediate TEX, METAFONT, Macro Writing, Output Routines, and PostScript.

The Host.
The core of McGill University, with almost 30,000 students, is located in downtown Montréal, but the University has other facilities, research sites, and museums throughout the city and Canada. The green central campus is a prime downtown picnicking spot. Tours of the campus can be arranged in French or English. The annual meeting will be held in the Bronfman Building on Sherbrooke Street; courses will be held elsewhere on campus.

The City.
An island city of almost 2 million, Montréal is the second largest French-speaking city outside of France; English is widely spoken, and Italian, Greek, and Chinese are common. Expect the possibility of hot weather, and remember one possibility of escape: a wonderful Métro connects an underground city. There are stores, restaurants, bakeries, gardens, museums, and galleries for every taste.

The Feast.
TUG will host a cash-bar cocktail reception in the David M. Stewart Museum, a great opportunity to make or renew professional acquaintances or to visit the maritime and military exhibits. The Québecois dinner will be served in Le Festin du Governeur, a restaurant located in the old fort on Ile Ste-Hélène. This 17th century feast, which includes drinks, soup, quiche, beef brochette, vegetables, wine, dessert, and beverages, is accompanied by a revelry of performers—an historical adventure in eating and entertainment.

The Memories.
(To be filled in by each of you—)

August 22–24, 1988

The TEX Users Group, P.O. Box 9506, Providence, Rhode Island 02940, USA
Call for reservations or information: (401) 272–9500, ext. 232