

Damn braces. Bless relaxes.

William Blake *Proverbs of Hell*

TUGBOAT

THE T_EX USERS GROUP NEWSLETTER
EDITOR BARBARA BEETON

VOLUME 5, NUMBER 1 • MAY 1984
PROVIDENCE • RHODE ISLAND • U.S.A.

ADDRESSES OF OFFICERS, AUTHORS AND OTHERS

- BEETON, Barbara**
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500
- BERTELSEN, Erik**
Regional EDP Center, Univ of Aarhus (RECAU)
Ny Munkegade
Bygning 540
DK-8000 Aarhus C, Denmark
45 6 128355; Telex: 64 754 recau dk
- BLOCK, Neil**
Hughes Aircraft Co.
Bldg. A1, M/S 3C923
P.O. Box 9339
Long Beach, CA 90810
213-513-4891
- CANZII, G.**
Università degli Studi di Milano
Istituto di Cibernetica
Via Viotti 5
20133 Milano, Italy
23.52.93
- CARNES, Lance**
163 Linden Lane
Mill Valley, CA 94941
415-388-8853
- CHILDS, S. Bert**
Dept of Computer Science
Texas A & M University
College Station, TX 77843
409-845-5470
- CODE, Maria**
Data Processing Services
1371 Sydney Dr
Sunnyvale, CA 94087
- DANIELS, Susan**
Hewlett-Packard, Boise Division
Post Office Box 15
Boise, ID 83707
208-323-2298
- DUPREE, Chuck**
Digital Equipment Corporation
VAX S/W Documentation
110 Spit Brook Road
Nashua, NH 03062
603-881-1295
CDupree@DEC-Marlboro
- FUCHS, David**
Department of Computer Science
Stanford University
Stanford, CA 94305
415-497-1646
- FURUTA, Richard**
Univ of Washington
Computer Science, FR-35
Seattle, WA 98195
206-543-7798
Furuta@Washington
- GENOLINI, F.**
Università degli Studi di Milano
Istituto di Cibernetica
Via Viotti 5
20133 Milano, Italy
23.52.93
- GOUCHER, Raymond E.**
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500 x232
- GROSSO, Paul**
Textset, Inc
P O Box 7993
Ann Arbor, MI 48107
313-761-9732
- GUENTHER, Dean**
Computer Service Center
Washington State University
Computer Sci Bldg, Rm 2144
Pullman, WA 99164
509-335-6611 x233
- HAGEN-WITTBECKER, Alan**
Computing Service Center
Washington State University
Computer Sci Bldg, Rm 2144
Pullman, WA 99164
509-335-6611
- HAUS, Goffredo**
C.I.L.E.A. & Istituto di Cibernetica
viale Pisa, 10
I-20146 Milano, Italy
- KELLER, Arthur**
Computer Science Dept
Stanford University
408C Margaret Jacks Hall
Stanford, CA 94305
415-497-3227
ARK@SAIL
- KELLERMAN, David**
Kellerman & Smith
2343 SE 45th Ave
Portland, OR 97215
503-232-4799
- KNUTH, Donald E.**
Department of Computer Science
Stanford University
Stanford, CA 94305
DEK@SAIL
- LØFSTEDT, Benedikt**
RECAU, Build. 540
Ny Munkegade
8000 Aarhus C, Denmark
06-128355
- LUCARELLA, D.**
Università degli Studi di Milano
Istituto di Cibernetica
Via Viotti 5
20133 Milano, Italy
23.52.93
- MacKAY, Pierre A.**
University of Washington
Department of Computer Science, FR-35
Seattle, WA 98195
206-543-2266
MacKay@Washington
- MALLETT, Rick**
Computing Services
Room 1208 Arts Tower
Carleton University
Ottawa (K1S 5B6), Ontario Can
613-231-7145
- NICHOLS, Monte C.**
Exploratory Chemistry Division
Sandia National Laboratories 8313
Livermore, CA 94550
415-422-2906
- NOOT, Han**
Stichting Mathematisch Centrum
Tweede Boerhaavestraat 49
1091 AL Amsterdam, Netherlands
- 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
- PLASS, Susan**
Polya 203
Center for Information Technology
Stanford University
Stanford, CA 94305
415-497-1322
- RODGERS, David**
Textset, Inc
P O Box 7993
Ann Arbor, MI 48107
313-996-3566
- SMITH, Barry**
Kellerman & Smith
2343 SE 45th Ave
Portland, OR 97215
503-232-4799
- SPIVAK, Michael**
1660 West Alabama, #7
Houston, TX 77006
- STERKEN, Jim**
Textset, Inc
P O Box 7993
Ann Arbor, MI 48107
313-971-3628
- STROMQUIST, Ralph**
MACC
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706
608-262-8821
- THEDFORD, Rilla**
Intergraph Corporation
One Madison Industrial Park
Huntsville, AL 35807
205-772-2000
- TOBIN, Georgia K. M.**
Office of Research
OCLC Online Computer Library Center, Inc
6565 Frantz Rd
Dublin, OH 43017
614-764-6000
- TUTTLE, Joey K.**
I P Sharp Associates
220 California Avenue
Suite 201
Palo Alto, CA 94306
415-327-1700
- WELLAND, Robert**
Department of Mathematics
Northwestern University
2033 Sheridan Road
Evanston, IL 60201
312-864-2898
- WHIDDEN, Samuel B.**
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500
- WHITNEY, Ronald**
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500
- WINTER, Janene**
Computer Service Center
Washington State University
Computer Science Bldg, Rm 2144
Pullman, WA 99164
509-335-6611
- ZABALA, Ignacio**
Centro de Cálculo
Facultades de Ciencias
Universidad de Valencia
Cametera de Ademuz
Valencia, Spain
011-34-6-357-4065
- ZAPP, Hermann**
Seitersweg 35
D-6100 Darmstadt, Fed Rep Germany

TUGboat, the newsletter of the TeX Users Group (TUG), is published twice each year (generally in the Spring and the Fall) for TUG by the American Mathematical Society, P.O. Box 6248, Providence, RI 02940. Annual dues for individual members of TUG include one subscription to TUGboat; fees are detailed on the membership form bound into the back of this issue. Applications for membership in TUG should be addressed to the TeX Users Group, c/o American Mathematical Society, P.O. Box 1571, Annex Station, Providence, RI 02901; applications must be accompanied by payment.

Manuscripts should be submitted to a member of the TUGboat Editorial Committee, whose names and addresses are listed inside the front cover. Articles of general interest, or not covered by any of the topics listed, as well as all items submitted on magnetic tape, should be addressed to Barbara Beeton, American Mathematical Society, P.O. Box 6248, Providence, RI 02940.

Submissions to TUGboat are for the most part reproduced with minimal editing. Any questions regarding the content or accuracy of particular items should be directed to the authors.

OFFICIAL ANNOUNCEMENTS

TUG Meeting/Courses, August 13–24, 1984, Stanford University

A meeting of the T_EX Users Group will be held August 15–17, 1984, at Stanford University. August 13–14 will be occupied by a short course on document design. Joey Tuttle, of I.P. Sharp Associates, Palo Alto, will once again be in charge of compiling the program and of local arrangements. Any suggestions concerning subjects to be covered at the meeting should be conveyed to either Joey Tuttle, (415) 327-1700, or Ray Goucher, (401) 272-9500, ext. 232. August 20–24, Arthur Keller will teach a T_EX for Beginners class, comprising four hours of lectures interspersed with four hours of hands-on experience daily. A preliminary program appears on page 13 and a registration form is being mailed with this issue. *Preregistration deadline*: July 20, 1984.

TUG Membership Dues and Privileges

Memberships and Subscriptions

1984 dues for individual members are as follows:

North America:

- New (first-time) members or subscribers: \$20.
- Membership and subscription renewals: \$30.

Outside North America (includes air mail postage):

- New (first-time) members or subscribers: \$25.
- Membership and subscription renewals: \$35.

Membership privileges include all issues of TUGboat published during the membership (calendar) year. Anyone inquiring about TUG will be sent a complimentary copy of TUGboat Vol. 1, No. 1 (1980), along with a current copy of the membership list, a list of errata to the T_EXbook, and forms for acquiring T_EX82, joining TUG and ordering publications available from TUG.

Issues to domestic addresses are mailed third class bulk, which may take up to six weeks to reach their destinations. If you have not received an issue to which you are entitled, write to TUG at the address given below.

Institutional Membership

1984 Institutional Membership dues for educational organizations are \$200; for non-educational, \$300. Membership privileges include: designating up to 5 persons as individual members and special reduced rates for participation at TUG meetings and T_EX-related courses and for purchase or lease of videotapes. In addition, institutional members are listed in each issue of TUGboat. For further information, call Ray Goucher at (401) 272-9500, ext. 232.

Submitting Items for Publication in TUGboat

The deadline for submitting items for Vol. 5, No. 2 (1984), will be October 1, 1984; the mailing date will be November 8. Contributions on magnetic tape or in camera copy form are encouraged; see the statement of editorial policy, page 3, Vol. 3, No. 1. Editorial addresses are given on the inside front cover. For instructions on preparing magnetic tapes or for transferring items directly to the AMS computer, write or call Barbara Beeton at the address given, (401) 272-9500, ext. 299.

TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call T_EX Users Group, Attention: Ray Goucher, c/o American Mathematical Society, P. O. Box 6248, Providence, RI 02940, (401) 272-9500, ext. 232.

* * * * *

General Delivery

* * * * *

MESSAGE FROM THE PRESIDENT

Pierre MacKay

There is much to celebrate. \TeX "came of age" in December, and the \TeX book is out. Which leads to the first and most urgent of several messages, addressed to those members of TUG who may still be using $\text{P}\TeX$, TeX80 or one of their offspring. The message is, "STOP!" TeX80 is now an interesting piece of technological archaeology; it was a necessary step along the way, but it does not offer anything like the power of genuine \TeX , and it can never improve. Change now, and you will face a slightly painful but rather brief effort of macro conversion, rather like getting a tooth fixed. Be brave. Get it over with. You will be glad you did.

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

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

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

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

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

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

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

* * * * *

\TeX INCUNABULA

by Donald E. Knuth

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

away. The following list is confined to works that were actually published, although my files also include dozens of concert programs, church programs, newsletters, and such things that my wife and I have been putting together ever since T_EX began to be operational.

The first edition of the T_EX manual is already quite rare, although I believe several hundred copies were printed. It was called "Tau Epsilon Chi, a system for technical text," Stanford Computer Science Report STAN-CS-78-675 = Artificial Intelligence Laboratory Memo AIM-317 (September, 1978), 198 pp. The American Mathematical Society published a corrected version of this manual in June, 1979; my wife Jill designed the cover of this edition, of which I believe approximately 1000 copies were sold. If you have a "clean" copy you will be able to distinguish a subtle T_EXture on the cover (quite similar to the example on page 225 of *The T_EXbook*).

Most people learned about the prototype version of T_EX by reading the third edition of the manual, which appeared as part 2 of *T_EX and METAFONT*, co-published by AMS and Digital Press in the latter part of 1979. Approximately 15,000 copies of this book were printed.

The type for all three editions was produced on experimental low-resolution equipment that was not available commercially. The first edition used a Xerox Graphics Printer (XGP) that had been donated to Stanford's Artificial Intelligence Laboratory; the second used a one-of-a-kind "Colorado" printer at Xerox Electro-Optical Systems in Pasadena, California; and the third used a "Penguin" printer at Xerox's Advanced Systems Department in Palo Alto. These machines had variable resolution, which was set to 200 pixels/inch on the XGP and 384 pixels/inch on the others. Dale Green and Leo Guibas were instrumental in getting the latter two editions printed.

The METAFONT manual had a similar printing history: It first came out on the XGP as "METAFONT, a system for alphabet design," Stanford Computer Science Report STAN-CS-79-762 = Artificial Intelligence Laboratory Memo AIM-332, 105 pp.; then it was reprinted on a Penguin, with minor corrections, in the Digital Press book.

Of course, user manuals don't count as real milestones in publishing. I like to think that the first *real* book to be printed with T_EX was a 28-page keepsake that was made for my wife's relatives at Christmastime, 1978. This book included eighteen original linoleum block illustrations, into which we pasted XGP-produced text set in a special 14-point extended variant of the prototype Computer Mod-

ern font. In order to compensate for the XGP's limited resolution, we prepared magnified copy and the printer reduced it to 70%; the effective resolution was therefore about 286 pixels/inch. The title, opening pages, and colophon are illustrated here (reduced another 65% from the published size). About 100 copies were printed, of which roughly 25 were sold and the remaining 75 were given as gifts. A complete library citation for this book would read as follows: "*Lena Bernice: Her Christmas in Wood County, 1895*. By Elizabeth Ann James, with illustrations by Jill Carter Knuth. Columbus, Ohio: Rainshine Press, 1978."

David W. Wall made an unusual application of T_EX and METAFONT in his Ph.D. thesis, "Mechanisms for broadcast and selective broadcast," Stanford Computer Science Report STAN-CS-82-919 = Stanford Computer Systems Laboratory Technical Report No. 190 (June 1980), 120 pp. He considered each illustration to be a "character" in a new "typeface," and he drew these large characters with METAFONT; then he superimposed textual labels using T_EX. This approach would defeat our current METAFONT software if the figures were to be drawn at high resolution, but he got away with it because he was using the XGP. (See the samples attached, which have been reduced to 65% of their original size.) In David's words, "I fear I've opened a Pandora's box; this isn't exactly what METAFONT was designed for. But ain't it purty?"

All this time we didn't have access to a high-resolution phototypesetter, but after several months of work David Fuchs and I successfully built an interface to an Alphatype CRS machine in the winter and spring of 1980. (Most of this effort was directed to complete revision of the microcode inside the CRS, because of the machine's limited font storage.) My notes show that we produced the first decent sample pages on April 1, 1980; and the first page of output that was eventually published was my one-page poem entitled "Disappearances" that appeared on page 264 of *The Mathematical Gardner*, edited by David Klarner (Belmont, California: Wadsworth, 1981). In May I sent off a longer paper, "The Letter S," that was published in *The Mathematical Intelligencer* 2 (Heidelberg: Springer-Verlag, 1980), 114-122.

Most of my time during April, May, June, and July of 1980 was spent making the final revisions to a big book that had been the original impetus for all of my work on T_EX and METAFONT. The volume had already been typeset during 1976 at The Universities Press, Belfast, using Monophoto systems called Cora and Maths, but the results were not

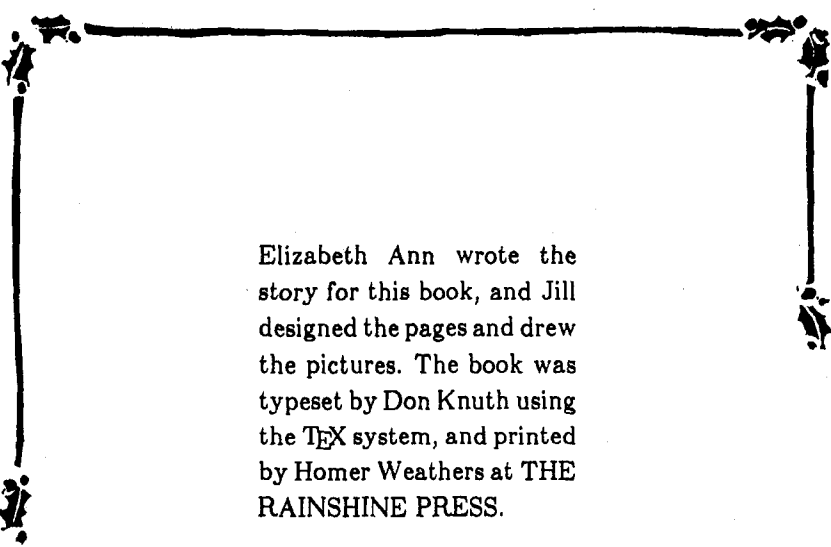


LENA BERNICE

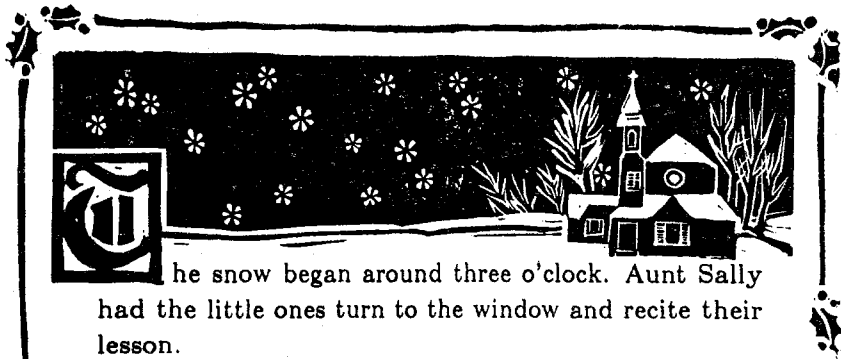
Her Christmas in Wood County, 1895

Lena Bernice was our grandmother.
She told us about her first Christmas
tree. She told us many things while
the snow fell.

Elizabeth Ann and Jill
Christmas 1978



Elizabeth Ann wrote the
story for this book, and Jill
designed the pages and drew
the pictures. The book was
typeset by Don Knuth using
the T_EX system, and printed
by Homer Weathers at THE
RAINSHINE PRESS.



The snow began around three o'clock. Aunt Sally had the little ones turn to the window and recite their lesson.

“See the snow softly fall
over barns and churches tall.”

Gussie was trying to teach Horace at home, so she copied it down. She wanted Horace to be a member of the state legislature, like Ben James who was her uncle and the greatest orator in Wood County.

Lena Bernice thought about little Jimmy Reed in the lesson book and how he wondered if the snow tasted of sugar. She thought about the brave dog, Cæsar, who had protected his mistress during the blizzard in Old Kentucky. She thought about the layer of ice on Rock Pond.

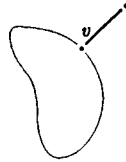
Aunt Sally took up the *Illustrated Geography* and showed The Entire Class a picture of the Alps and of the dear Saint Bernards who saved many a folk from certain death.



Figure 2. The opening pages of *Lena Bernice*.

When a vertex enters FULFILL-OBLIGATION, it looks at its fragment state and picks out the smallest outgoing edge. From here we must consider three cases.

If that edge selected is incident to v , then v sends a message—which includes its fragment state—to the other endpoint, which is not in v 's fragment; when the other endpoint receives it, it merges fragment states, with the result that it now has a larger fragment state than v did.



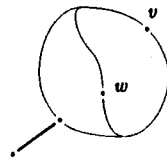
Edge Leaves at v

If the edge is not incident to v but rather to some other vertex w , then v transfers the obligation to w along with its fragment state. If w agrees that this edge is the appropriate one to make into a branch, it sends a message with its fragment state to the other endpoint; since that fragment state includes the state for v , the result is the same as in the previous case—the far endpoint will have a fragment state larger than that of v .



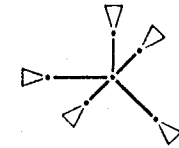
Endpoint Agrees with v

On the other hand, if w looks at its own fragment state and finds a better edge than the one v selected, then w must know about a vertex in the fragment of v and w that v did not know was present. Thus as soon as w merged its fragment state with that of v it had a larger fragment state than v . Its fragment state may even have been larger than v 's beforehand.



Endpoint Disagrees with v

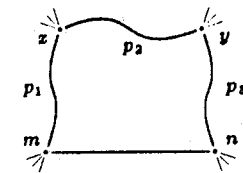
The idea behind this alternate approach is that instead of letting a branch partition the tree into two subtrees, we will let a node partition the tree into n subtrees, where n is the number of neighbors it has. A node sends "yes" across a branch as soon as it knows that there is an original node on this side of the branch, or a "no" as soon as it knows there isn't. We see that the case of an original node is trivial; since it is an original node, it can send "yes" out along all incident branches immediately. Then it can simply wait for answers, keeping or deleting branches according to the messages received across them.



A Node and Its n Subtrees

An added node, in contrast, has a more complicated algorithm than before. If it has n neighbors, it must wait until it has received one "yes" message or $(n - 1)$ "no" messages before it knows anything useful. It can acknowledge any "no" it receives, but can do nothing more. If it receives $(n - 1)$ "no" messages, it knows that the only subtree containing original nodes is the one from which it has not received a message, and so it can send "no" to that subtree, wait for acknowledgment, and then leave.

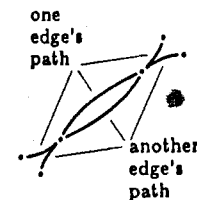
We observe that x precedes y on this path p . For if $x = y$ then this single vertex is on both path q_1 and path q_2 , which are represented by edges without a common endpoint. This violates Lemma 4. Similarly, if x follows y then y is on the portion of path p from m to x and so by Lemma 2 it must also be on the portion of path q_1 from m to x . Thus y is on both q_1 and q_2 , which again contradicts Lemma 4. So x precedes y .



Replacing $\{m, n\}$ with p

Thus we can divide the path p into three pieces: p_1 from m to x , p_2 from x to y , and p_3 from y to n . We have defined x and y in such a manner that the vertices on p_1 only appear on paths incident to m , the vertices on p_3 only appear on paths incident to n , and the vertices on p_2 do not appear either place and hence have not yet appeared at all.

We are trying to show that the graph G_S has no cycles. There is one easy way we might produce a cycle: if a pair of vertices appears on the paths for two of the edges in the tree, and if there are two routes of equal cost between these two vertices, we could pick a different route in each case. The first two lemmas show that nothing this simple will produce a cycle as long as we are using a consistent tie-breaking rule, because we would have to pick the same route both times. As a result, we can only form a cycle from bits and pieces of several paths.



One Way to Make a Cycle

Figure 3. Excerpts from David Wall's thesis.

satisfactory. I received the paper tapes from Belfast and converted them to pseudo- \TeX so that the re-keyboarding would be easier. (It isn't clear that I actually saved any time by this maneuver!) All 700 pages of the book finally fell into place; and the camera-ready copy for *Seminumerical Algorithms: The Art of Computer Programming*, Volume 2, second edition (Reading, Massachusetts: Addison-Wesley, 1981) was completed at 2 am on the morning of Monday, July 29, 1980. On October 22 I had to remake page iv (so that it contained Library of Congress information). A bound copy of the book actually appeared in my hands on January 4, 1981. It seems most appropriate to regard the appearance of this book as the actual birth of \TeX in the world of publishing. My publishers prepared a limited edition of 256 copies, hand bound in leather, to commemorate the occasion. (I believe that only a few people ever purchased these special copies, because computer scientists didn't want to pay for leather binding, while lovers of fine printing didn't cherish the invasion of computers. However, about 10,000 copies of *Seminumerical Algorithms* were sold in its regular binding during 1981.)

Meanwhile other people at Stanford had been getting books ready for publication, using \TeX and the Alphatype in our lab. The first of these to be finished was *KAREL the ROBOT: A Gentle Introduction to the Art of Programming* by Richard E. Pattis (New York: Wiley, 1981), 120 pp. Many of the illustrations in this book were typeset using special symbols **METAFONT**ed for the occasion with David Wall's help. The next book of this kind was *Practical Optimization* by Philip E. Gill, Walter Murray, and Margaret H. Wright (New York: Academic Press, 1981), 417 pp. Speaking of optimization, a paper by Bengt Aspvall and Yossi Shiloach, "A polynomial time algorithm for solving systems of linear inequalities with two variables per inequality," *SIAM J. Computing* **9** (1980), 827-845, was also produced on the Alphatype in our lab that year.

Scott Kim was the first to use our Alphatype together with \TeX to produce copy with non-**METAFONT** typefaces, in *Inversions: A Catalog of Calligraphic Cartwheels* (Peterborough, New Hampshire: Byte Books, 1981), 124 pp. He later **METAFONT**ed some special symbols that are featured in Arthur Keller's *A First Course in Computer Programming using PASCAL* (New York: McGraw-Hill, 1982), 319 pp. (It's interesting to note that the Italian translation of this text, *Programmare in PASCAL* (Bologna: Zanichelli, 1983), 303 pp., was one of the first books to be published from \TeX output in Italy. The Italian translators worked independently of the

American author, and produced the camera-ready copy on a Versatec machine in Milano—unfortunately without Scott's symbols.)

Terry Winograd was one of the first \TeX users and (therefore) one of the first to complain about its original limitations; for example, I added $\backslash\text{def}$ at his request, on November 28, 1978. He had begun writing a book with a system called PUB [Larry Tesler, "PUB, The Document Compiler," Stanford Artificial Intelligence Project Operating Note 70 (March, 1973), 84 pp.], then had converted all the files to BRAVO [Butler W. Lampson, "Bravo Manual," in *Alto User's Handbook*, Xerox Palo Alto Research Center (1978), 32-62], before converting again to \TeX . Winograd contributed macros for indexing to the first issue of *TUGboat*, and his struggles with the early \TeX finally led to the completed book *Language as a Cognitive Process*, Vol. 1: *Syntax* (Reading, Massachusetts: Addison-Wesley, 1983), 654 pp. He used Computer Modern fonts, but substituted Optima for the (awful) sans serifs that I had been using at the time.

Gio Wiederhold modified ACME files that he had used to prepare the first edition of his database book so that he could typeset the second edition with \TeX . He says that it took about six months to do the final formatting (e.g., writing extra copy so that page breaks would occur in desirable places). The resulting volume holds the current record for the longest book to be produced in our lab: *Database Design*, second edition (New York: McGraw-Hill, 1983), 767 pp.

Another faculty colleague, Jeffrey D. Ullman, converted first-edition Troff files to \TeX files, for his book *Principles of Database Systems*, second edition (Rockville, Maryland: Computer Science Press, 1982), 491 pp. Then he used \TeX directly to write *Computational Aspects of VLSI* (Rockville, Maryland: Computer Science Press, 1984), 505 pp. Jeff's 13-year-old son, Peter, helped by using **METAFONT** to create special fonts for the typesetting of VLSI stipple patterns.

My co-author Daniel H. Greene \TeX ed our book *Mathematics for the Analysis of Algorithms* (Boston: Birkhäuser, 1981), 107 pp. It's interesting to compare the first edition to the second (1982, 123 pp.), because the fonts were significantly tuned up during the year that intervened between editions.

The books and articles mentioned so far were all typeset by their authors; this is to be expected in a computer science department. But a few experiments were also undertaken in a more traditional way, where the \TeX composition was done by people who were skilled at keyboard entry but not in-

timately familiar with the subject matter. I think the first such books to be done in our lab were the *Handbook of Artificial Intelligence*, vol. 2, edited by Avron Barr and Edward A. Feigenbaum (Los Altos, California: William Kaufman, 1982), 441 pp.; *Handbook of Artificial Intelligence*, vol. 3, edited by Paul R. Cohen and Edward A. Feigenbaum (Los Altos, California: William Kaufman, 1982), 652 pp.; *Introduction to Arithmetic for Digital Systems Designers* by Shlomo Waser and Michael J. Flynn (New York: Holt, Rinehart and Winston, 1982), 326 pp.; *Introduction to Stochastic Integration* by Kai Lai Chung and Ruth J. Williams (Boston: Birkhäuser, 1983), 204 pp.; *Hands-on Basic: For the IBM Personal Computer*, by Herbert Peckham (New York: McGraw-Hill, 1983), 320 pp.; *Hands-on Basic: For the Apple II*, by Herbert Peckham with Wade Ellis, Jr., and Ed Lodi (New York: McGraw-Hill, 1983), 332 pp.; *Hands-on Basic: For the TRS-80 Color Computer*, by Herbert Peckham with Wade Ellis, Jr., and Ed Lodi (New York: McGraw-Hill, 1983), 354 pp.; *Hands-on Basic: For the Atari 400/800/1200XL*, by Herbert Peckham with Wade Ellis, Jr., and Ed Lodi (New York: McGraw-Hill, 1983), 319 pp.; and *Probability in Social Science* by Samuel Goldberg (Boston: Birkhäuser, 1983), 131 pp. Incidentally, the typesetting of this last book was done by my son John during the summer of 1982, before he had learned anything about computer programming. I helped him with a few `\halign` constructions, but otherwise he worked essentially without supervision. At that time he was about to be a senior in high school; I know of at least three other children in his high school who were typesetting books with \TeX . (These other books have not come out yet.)

It may be of interest to note that the first volume of the *Handbook of Artificial Intelligence* (1981) was done with early Computer Modern fonts on our Alphatype, but the typesetting was by PUB rather than \TeX . In particular, all hyphenation in that book was done by hand.

Members of Stanford's Space, Telecommunications and Radioscience Laboratory began to use \TeX for articles that were typeset on our Alphatype and published in journals and conference proceedings. I believe the first of these were "Photographic observations of earth's airglow from space," by S. B. Mende, P. M. Banks, R. Nobles, O. K. Garriott, and J. Hoffman, *Geophysical Research Letters* **10** (1983), 1108-1111; "Solar wind control of the low-latitude asymmetric magnetic disturbance field," by C. Robert Clauer, Robert L. McPherron, and Craig Searls, *Journal of Geophysical Research* **88** (1983),

2123-2130; "VLF wave injections from the ground," by Robert A. Helliwell, in *Active Experiments in Space* (Paris: European Space Agency, 1983), 3-9; "Electron beam experiments aboard the space shuttle," by P. M. Banks, P. R. Williamson, W. J. Raitt, S. D. Shawhan, and G. Murphy, *ibid.*, 171-175. Dozens more are currently in press.

The software we used to interface between \TeX and the Alphatype CRS was used at three other sites: The American Mathematical Society (Providence, Rhode Island), Kunglig Tekniska Högskolan (Stockholm, Sweden), and Bell Northern Research (Mountain View, California). I have only sketchy information about what books were produced with \TeX at other installations, but I'll give a partial list so that people at those sites might be moved to provide a more correct history.

The first AMS use of \TeX and the Alphatype in my collection is an article entitled "1980 Wiener & Steele Prizes Awarded," *Math. Notices* **27** (1980), 528-533. (Since then an ever-growing percentage of the *Notices* has been \TeX ed.) The SIAM-AMS-MAA Combined Membership List for 1981-1982 was another early production, as was the *AMS Catalog of Publications* for 1981-1982.

The Society first put \TeX 's mathematical abilities to the test in the pre-preliminary edition of Michael Spivak's *The Joy of \TeX* , 134 pp., which was distributed at the AMS meeting in San Francisco (January 1981). There are many instances of \TeX usage in the subsequent *Proceedings* [Volume 85 (1982), pp. 141-488, 567-595, 643-665, 673-674; Volume 86 (1982), pp. 12-14, 19-86, 103-125, 133-142, 148-150, 153-183, 186-188, 253-274, 305-306, 321-327, 363-374, 391, 459-490, 511-524, 574-598, 609-624, 632-637, 641-648, 679-684]. David J. Eck's thesis, "Gauge-natural bundles and generalized gauge theories," was published in *Memoirs of the American Mathematical Society* **33**, number 247 (September 1981), 54 pp.; this memoir includes an interesting preface by Richard Palais, pointing out that David was pleased to be the first guinea pig for $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\TeX}$ when he typeset the thesis.

Before AMS began using the Alphatype, they published several things from photo-reduced Varian output. The indexes to individual issues of *Mathematical Reviews* have been done with \TeX since November 1979 (Volume 58, #5); the *Combined Membership List* for 1980-1981 also came off the Varian.

Several books composed elsewhere have also been typeset with the facilities at AMS headquarters, notably Oregon Software's *PASCAL-2: Version 2.0 for RSX-11* (1981), 186 pp.; *Turtle Geometry* by Harold

Abelson and Andrea A. diSessa (Cambridge, Massachusetts: MIT Press, 1981), 497 pp.; and *History of Ophthalmology* by George Gorin (Wilmington, Delaware: Publish or Perish, 1982), 646 pp.

At Bell Northern, I think T_EX was used mostly (or entirely?) for company-confidential reports. But I have seen several excellent publications from the Swedish Royal Institute of Technology, notably a 46-page monograph on *Non-linear Inverse Problems* by Gerd Ericksson, Report TR17A-NA-8209 (1983), and I would like to know more about their independent experiences with T_EX.

The University of Wisconsin Press sent me a copy of their *1981 Fall Catalog*, which they told me was typeset by T_EX. I don't know if T_EX actually helped to produce any of the books listed in the catalog.

When Computer Modern fonts are not used, it isn't so easy to tell that T_EX was behind the formatting. But I have been assured that the book *Guide to International Commerce Law* by Paul H. Vishny (Colorado Springs, Colorado: Shepard's/McGraw-Hill, 1981), 782 pp., was entirely typeset by T_EX, using an IBM 370/3081 coupled to an APS 5 phototypesetter.

Some books have been published directly from Xerox Dover output (384 dots/inch resolution) that was printed at Stanford. In particular, the original hardcover edition of Joseph Deken's *The Electronic Cottage* (New York: William Morrow, 1982), 334 pp., was produced in this way, because of tight publication deadlines, and so were the books *Arithmetic and Geometry: Papers Dedicated to I. R. Shafarevich on the Occasion of His Sixtieth Birthday*, edited by Michael Artin and John Tate (Boston: Birkhäuser, 1983); vol. 1, 359 pp., vol. 2, 481 pp.

Max Díaz was instrumental in setting up a T_EX installation at the Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas of the Universidad Nacional Autónoma de México (i.e., at IIMAS-UNAM). The first T_EX-produced book to be done entirely in Mexico was *Nonlinear Phenomena*, ed. by Kurt Bernardo Wolf, *Lecture Notes in Physics* 189 (1983), 464 pp. Max's F_ácil T_EX macros were, of course, the basis of this production, which was photoreduced from low-resolution output.

The *AI Magazine* (an official publication of the American Association for Artificial Intelligence) has been typeset with T_EX in our laboratory since volume 3, number 2 (Spring 1982). Actually nobody told me anything about this until June 16, 1983, when I received the following unsolicited letter from the managing editor, Claudia C. Mazzetti: "The production time of the magazine has decreased almost in half because of T_EX. We just want to express

our thanks for creating such a marvelous system!"

Well, by 1983 I was unable to understand why anybody would think the old version of T_EX was easy to use, since I had just spent two years removing hundreds of deficiencies. (All of the work reported above was produced by the old proto-T_EX system.) Furthermore, I'm still not entirely happy with the Computer Modern fonts, although the "Almost Computer Modern" version of July 1983 is much better than the fonts that we were using in 1980. I expect to make further improvements during the next two years, as I complete my research on typography. My goal is to have a new **META-FONT** in 1984 and a new Computer Modern in 1985. Meanwhile, we do have a new, permanent T_EX in 1983, and I'll conclude this list by mentioning the first three publications that have flowed from the new T_EX together with our new APS Micro-5 phototypesetter: *The T_EXbook* (Reading, Massachusetts: Addison-Wesley, 1984), 496 pp., was the first; it was sent to the publisher on October 12. *Coordinated Computing: Tools and Techniques for Distributed Software* by Robert E. Filman and Daniel P. Friedman (New York: McGraw-Hill, 1984), 390 pp., was the second. And my paper "Literate Programming" (15 pp.) was the third; this paper—which discusses WEB—combines Times Roman and Univers type with Computer Modern, and it will be published in volume 27 of *The Computer Journal*.

I like to think that the use of T_EX has not only produced books that are well formatted; it also seems to have helped produce books whose content is significantly better than books that were written in the old way. Part of this change is due simply to the advantages of word processing and computer editing, since changes are so much easier; but part of it is due to the fact that authors are able to choose the notations and formatting that they want, once they are free from the worries of communicating through several levels of other people to whom such notations might be unfamiliar. I believe that a large number of the books listed above show such improvements in scientific exposition; in particular, my own books have been greatly improved because I've been able to control the typesetting. I still rely heavily on the advice of professional editors and book designers, but I can be much more sure of the final quality than ever before, because there now is comparatively little chance that misunderstandings will introduce any errors. This, to me, is the "bottom line" that has made all of my work on T_EX seem worth while.

TUG 1983 FINANCIAL REPORT WITH COMPARISONS

April 24, 1984

	Actual 1982	Actual 1983	Budget 1984	Actuals through 3/84	Estimated thru 12/84
Income:					
Membership/Subscriptions ^{1,2}					
Individual	\$ 11,935	\$ 13,629	\$ 15,800	\$ 10,283	\$ 17,000
Postage	588	1,358	-0-	-0-	-0-
Institutional Membership ^{1,2}					
Educational	200	5,780	5,500	4,200	5,400
Non-educational	200	3,612	5,000	7,200	9,200
Publications					
Back issue sales ³	2,156	5,094	2,500	3,159	5,500
Other publications ⁴	376	634	500	2,318	3,500
Meetings ^{2,5}					
Summer meeting	11,025	15,209	15,000	2,880	25,000
2-day course	27,018	14,187	15,000	3,015	15,000
5-day course	-0-	-0-	-0-	3,850	40,000
Manufacturers reps' fees	300	450	600	-0-	600
Winter meeting	4,500	-0-	6,000	-0-	-0-
Other					
Videotape sales/rental	2,200	4,998	3,000	480	5,000
Advertising & mailing list sales	-0-	325	500	117	600
Royalties (<i>The TEXbook</i>)	-0-	-0-	1,000	-0-	1,000
Total income	\$ 60,498	\$ 65,276	\$ 70,400	\$ 37,502	\$ 127,800
Expenses:					
TUGboat (2 issues)					
Printing	\$ 2,220	\$ 2,620	\$ 2,596	\$ -0-	\$ 2,640
Postage	1,143	934	944	-0-	1,200
Editorial services	3,460	3,772	3,835	-0-	3,900
Clerical services	180	182	177	-0-	180
Computer expenses	2,100	2,220	2,360	-0-	2,400
Meetings ^{2,5}					
Meeting & 2-day course	4,130	7,311	5,900	209	6,600
5-day course	-0-	-0-	-0-	640	18,000
Winter meeting	2,460	-0-	2,360	-0-	-0-
Other					
Other publications ⁴	225	474	354	668	1,200
ANSI meetings ⁶	2,503	1,280	1,652	-0-	1,680
Legal and tax consulting	-0-	-0-	590	240	1,200
Postage, general mailings	1,335	3,605	2,950	1,266	3,600
Printing back issues ³	-0-	7,953	-0-	-0-	-0-
Printing, other	1,212	426	2,360	947	960
Administrative support ⁷	-0-	12,923	11,800	5,292	14,400
Clerical services	3,120	5,708	6,313	1,026	6,600
Subsidies ⁸	-0-	-0-	1,180	-0-	-0-
Video tape duplication	-0-	2,553	1,180	-0-	2,010
Computer expenses	1,455	4,637	1,770	918	3,600
Programming ⁹	-0-	-0-	3,009	3,816	4,800
Miscellaneous ¹⁰	1,688	1,622	2,360	1,258	2,400
Total expenses	\$ 27,231	\$ 58,220	\$ 53,690	\$ 16,280	\$ 77,370
Summary:					
Balance forward	\$ (8,660)	\$ 24,607	\$ 31,663		\$ 31,663
Income (Actual/Budget/Est.)	60,498	65,276	70,400		127,800
Expenses (Actual/Budget/Est.)	(27,231)	(58,220)	(53,690)		(77,370)
Balance	\$ 24,607	\$ 31,663	\$ 48,373		\$ 82,093

TUG 1983 Financial Report

(Continued from preceding page)

Notes:

All 1983 expense figures include an AMS overhead charge of 21.29%. Except as indicated, these remarks apply to the 1983 year. The 1984 budget was prepared in June 1983 for the July TUG meeting.

1. 1983 ended with a total of 835 memberships/subscriptions: U.S. - 617; Canada & Mexico - 52; Other Foreign - 166. (Beginning in 1984, foreign air mail postage is included in membership/subscription fee.) As of April 24, 1984, there were 762 members/subscribers, including 57 Institutional Members: 28 educational; 29 non-educational (listed on the inside cover of this issue).
2. Publicizing TUG and the TUG Meeting/Course was accomplished through a news release to 19 trade publications, several of which are known to have published the notice, in addition to direct mailings to members and former members.
3. 200 copies of the 8 back issues of TUGboat were reprinted; in addition, 1,600 copies of each 1983 issue were printed, with those in excess of 1983 membership requirements (835) being charged to back issue printing. In 1983 300 back issues were sold and to date in 1984 200, which suggests that back issues will be a good source of income in the future. Outside of new-issue overprinting, it should not be necessary to reprint back issues in 1984 and possibly 1985.
4. 75 copies of Max Díaz's "Fácil T_EX" were sold. In January 1984, Arthur Samuel's "First Grade T_EX" was offered for sale; to date over 200 copies have been sold. In May, Hewlett-Packard's "The HP T_EX Macros" will be available through TUG.
5. 84 individuals attended Michael Spivak's "Introductory AMS-T_EX82 Users Course" and 135 members participated in the summer meeting, both conducted at Stanford University, July 11-15, 1983. Representatives from Autologic, Hewlett-Packard, Imagen and Quality Micro Systems gave presentations. At the time the 1984 budget was prepared, only a short course and summer meeting were planned for 1984 and the fees had not been set. The addition of Arthur Keller's week-long course should increase the 1984 income substantially.
6. Lynn Price represented TUG at the ANSI X3J6 meeting, Boulder, Colorado, in August 1983.
7. The American Mathematical Society made available the services of Ray Goucher in 1981 and 1982 at no charge. Beginning in 1983, the A.M.S. required reimbursement for his time. He manages all the administrative details associated with TUG, including daily income and expense accounting; budgeting and treasurer's reports; coordination of all aspects of meeting preparations, accounting, publicity and advertising, in addition to numerous other details.

He was appointed TUG Business Manager at the Stanford Meeting in July.

8. The Steering Committee made this amount available to the Finance Committee to subsidize travel and membership fees for individuals when appropriate.
9. Reprogramming to improve the functioning of the TUG data base.
10. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

* * * * *

**PRELIMINARY PROGRAM
TUG MEETING AND COURSES
AUGUST 1984, STANFORD UNIVERSITY**

The 1984 meeting of the T_EX Users Group will be held at Stanford University, August 15-17, 1984. It will be preceded, on August 13-14, by a short course on document design, and followed by a full-week course, August 20-24, on T_EX for Beginners. Attendance at both courses, particularly the Beginners' course, will be limited. A registration form is being mailed with this issue, and should be returned as soon as possible.

**First Principles of Typographic Design for
Document Production, August 13-14, 1984**

Topics:

Typographic structures in text
Typewriting and typesetting
The typographer's tools
Making text readable
Designing headings: keeping it simple
Laying out the page

The course sets out to establish some basic principles for the typographic design of simple text. The application of these principles to the design of documents, and the implementation of the resulting designs with T_EX, will be discussed.

Richard Southall and Leslie Lamport are the instructors. (Richard Southall, Donald Knuth and Chuck Bigelow presented a course on Metafont programming during the Stanford Spring quarter.)

TUG Summer Meeting, August 15-17, 1984

The principal topics for discussion at the 1984 meeting will be an update on the status of the T_EX82 typesetting system, the WEB system of structured documentation, and the next generation of Metafont.

The business meeting will include elections for the offices of Vice President and Secretary, who will be elected to two-year terms (the terms of the President and Treasurer extend until the 1985 meeting). Nominations for these offices may be made by petition containing the signatures of the nominee and of two other members in good standing; petitions should be sent to the Nominating Committee, T_EX Users Group, c/o American Mathematical Society, P.O. Box 6248, Providence, RI 02940, to be received no later than July 16.

The remainder of the technical program will include both new and updated presentations on:

- Introduction to T_EX and TUG for new users
- T_EX82 and Metafont news
- T_EX82 and WEB user experiences
- Questions and answers on T_EX82
- Site Coordinators' reports
- Birds-of-a-Feather sessions
- Macro Wizards' roundtable
- Output device manufacturers' representatives
- Output devices and drivers

Suggestions for additional topics to be covered at the meeting should be communicated to Joey Tuttle, (415) 327-1700, or to Ray Goucher, (401) 272-9500, ext. 232.

T_EX for Beginners, August 20–24, 1984

Tentative course outline:

Preliminaries

The process from text to typeset copy; Short introduction to the text editor; Short introduction to system commands.

Introduction to T_EX

Overview (review) of process of T_EX; Character set; A sample document from start to finish; Interpreting and correcting errors; Fonts; State changing macros vs. macros with parameters; Ligatures.

Copyediting

Proofreading marks; Typesetting language and concepts.

Sample application: a letter

Using macros; Simple adaptations to existing macros.

Sample application: a report

Designing a document; T_EX code for your design; Writing macros to make it easier.

Macros

Concepts and fundamentals; Examples and applications.

T_EX fundamentals

Dimensions; Boxes and glue; Interpreting and correcting errors; Modes.

Mathematics

In-line math formulas; Introduction to displayed formulas; Interpreting and correcting errors;

Sample application: a math paper

In-line vs. displayed formulas; Shilling vs. built-up fractions; Examples.

Breaking paragraphs into lines

Hyphenation; Penalties; Interpreting and correcting errors.

Breaking lines into pages

Penalties; Insertions and "floating" insertions; Interpreting and correcting errors.

Interpreting and correcting errors

Overview and review.

This course is an intensive introduction to T_EX, suitable for those without any exposure to T_EX and with no prior knowledge of typesetting. All participants will be expected to know how to use at least one computerized text editor (or word processor). This course should be particularly useful for evaluating the capabilities of T_EX, either for organizations investigating T_EX or for T_EX coordinators at sites where T_EX has been newly installed.

The intended format is four hours of lectures during each of the five days, with the remainder of each day occupied by hands-on experience using T_EX.

In order to provide adequate computer access, enrollment will be limited to 60 participants.

* * * * *

Software

* * * * *

HYPHENATION OF ITALIAN WORDS

G. Canzii, F. Genolini, D. Lucarella
Università degli Studi de Milano
Istituto di Cibernetica

This short note deals with the implementation of a procedure to hyphenate Italian words and the integration of such a procedure into the T_EX system.

Since the installation of T_EX we have been working to tailor the system to user needs in order to improve its circulation in the Scientific Community and Publishing Industry. With this aim, we have faced the problem of designing a suitable algorithm to find hyphenation points according to the rules of the Italian language. The procedure is completely general even if its first application has been inside the T_EX system.

The algorithm was influenced by the principles pointed out by Prof. D. Knuth in the description of the original procedure for the English language.

So, our main concern was to provide a fast and short routine capable of identifying the great majority of hyphenation points even if it does not

cover completely all the complex variety of instances for the Italian language. If the routine cannot resolve an ambiguity, the possible break point is bypassed so that it is certain that all the marked points provide a correct hyphenation.

The previous requirement led us to avoid any dictionary look-up to search, given the word, the corresponding break points, not even for exceptional cases. Consequently, a systematic study has been begun in order to identify all hyphenation rules effective for the Italian language. This approach has showed that the attempt to include specific irregular cases in a general set of rules would enlarge excessively the number of rules and the complexity of the algorithm. So, the patterns responsible for non-standard breaks have been identified and grouped in classes. The effect is to minimize the number of entries since every pattern is applicable to all the words belonging to the same irregular class.

In order to reach the hyphenation points, first the algorithm tries to acknowledge a syllable applying one of the standard grammar rules and, then, it tries to match one of the predefined irregular patterns. The procedure processes the word and marks the first break point, hence it recurses without keeping memory of the last break that has no influence on the rest of the word. So the word is scanned sequentially without any backtracking.

To activate the link to our procedure into the \TeX system, it was necessary to modify the original HYPHENATE and JUSTIFICATION procedures. Obviously, in the HYPHENATE procedure the code section devoted to searching break points for English words has been removed and our code substituted. The interfaces have not been modified so that the storing of the current word does not change and the hyphenation points are marked in the same way. In the JUSTIFICATION procedure, the pointers to word chars have been modified in order to analyze the entire word (HYPHENATIONWORD) including prefix and suffix. Just like in the original HYPHENATE procedure the only words that are analyzed consist of alphabetical strings without blanks or special punctuation symbols.

The algorithm has been running inside \TeX on an IBM 3083 under the MVS Operating System since November 83 and it works very well. In order to verify its functionality under a stressing condition various tests have been submitted with constraints of small values for the HSIZE parameter.

As a second step of this activity our aim is to implement the capability to select dynamically the appropriate hyphenation procedure. It would be preferable for the end user to have a switch com-

mand that drives \TeX to analyze the words in the text with either English or Italian rules.

Acknowledgements are due to A. Pilenga of the Cybernetics Institute for her support in integrating the procedure into \TeX and to B. Zonta of the National Research Council for her kind suggestions in classifying the hyphenation rules for the Italian language.

This activity was sponsored by the *Communication and Programming Project*, a cooperation between Honeywell Information Systems Italia and the University of Milan, Institute of Cybernetics.

* * * * *

HYPHENATION EXCEPTION LOG

Editor's note: This is the list requested by Don Knuth in his "Note on Hyphenation", TUGboat Volume 4, No. 2, page 64. It contains the words that have come to the Editor's attention, either through \TeX hax or by other means. As expected, there are more instances of missed than of incorrect hyphens. The " \TeX " column gives the result from the `\showhyphens{...}` facility of plain \TeX ; the second column contains versions which are suitable for insertion in a `\hyphenation{...}` list. Only the singular is shown for nouns; the plural should also be specified in a `\hyphenation{...}` list if it appears in your document.

\TeX	
ap-pendix	ap-pen-dix
cartwheel	cart-wheel
database	data-base
Di-jk-stra	Dijk-stra
in-fras-truc-ture	in-fra-struc-ture
manuscript	man-u-script
mi-cro-fiche	mi-cro-fiche
paramil-i-tary	para-mil-i-tary
postam-ble	post-am-ble
pream-ble	pre-am-ble
sub-scriber	sub-scrib-er
waveg-uide	wave-guide

* * * * *
Output Devices
 * * * * *

OUTPUT DEVICES AND COMPUTERS

Table I: Proof-Quality Devices															
	Diablo 630	Epson MX-80	Facit 4542	Fla.Data OSP	GE3000	HP2680	Imagen Imprint 10	Laser-grafix	Perq/Canon	Qume Sprint 5	Symbolics LGP-1	Varian	Versatec	Xerox Dover	Xerox 9700
Amdahl (MTS)							U. British Columbia								Univ. Michigan
Apollo					COS Info.		OCLC						OCLC		COS Info.
Ethernet						Stanford	Imagen, Inc.							Stanford	
DEC10 *							Vanderbilt*	Talaris*					Vanderbilt*		
DEC10							Stanford (Sail)	Talaris					G A Technology		Univ. Delaware
DEC20				Math Reviews			SRI; Columbia	Talaris				AMS		CMU	
DG MV8000								Texas A&M							
HP1000		JDJ Wordware													
HP3000	TeXeT					TeXeT				TeXeT					
IBM(MVS)															CIT
IBM(VM)							SLAC						SLAC		
Prime								Texas A&M					Livermore		
Siemens BS2000									GMD Bonn						
Sun				Textset			Sun Inc.								Textset
VAX (Unix)							UC Irvine	Talaris			UC* Santa Cruz		Cal. Tech.*; Univ. Wash.	Stanford	
VAX (VMS)		Inter-graph	INFN CNAF*			Inter-graph †	Argonne*	Texas A&M			Calma*	Sci. Appl.*	Inter-graph †		

Notes:

* Still running TeX80

† Graphics supported

Most of the interfaces listed here are not on the standard distribution tape. Some of them are considered proprietary. Information regarding these interfaces should be obtained directly from the sites listed.

Output device data is being maintained by Rilla Thedford. Anyone desiring more information or relaying new information can now send it to her on the Arpanet:

Rilla_Thedford%UMich-MTS@MIT

Table II: Typesetters						
	Alphatype CRS	APS-5/Micro-5	Compugraphic 8400	Compugraphic 8600	Harris 7500	Linotron 202
Amdahl (MVS)*		Washington St U*		Washington St U*		
Apollo		COS Info.				
CDC Cyber *				RECAU*		
DEC 20	AMS	Textset				Adapt, Inc.*
HP3000			Univ. of Sheffield			
IBM 370 *		Info. Handling*				
Sun		Textset				
Univac 1100 *				Univ. of Wisconsin*		
VAX (UNIX)					SARA	
VAX (VMS)		Intergraph †				

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, as usual, has been prepared (using \TeX 82 for pages 4-11 and \TeX 80 otherwise) on the DEC 2060 and Alphatype CRS at the American Mathematical Society.

- Autologic APS Micro-5: Description of Metafont Generic Font file format and Gray fonts for Metafont proofs, p. 31; DEC 2060.
- Compugraphic 8600: Dean Guenther et al., \TeX at Washington State University, p. 24; Amdahl V/8 (MVS).
- Compugraphic 8600: Goffredo Haus, *How to Tame Your Phototypesetter by \TeX* , p. 17; Univac 1100/80.
- Harris 7500: Han Noot, *DVI-Code to the Harris 7500*, p. 18; \TeX on CDC Cyber 175-750, typesetter driven by VAX 11/780 (Unix) and/or PDP 11/45.
- HP 2688A Laser Printer (300 dpi): Susan Daniels, *The HP \TeX Macros*, p. 49; HP 3000.
- Imagen Imprint-10 (240 dpi): G.M.K. Tobin, *The OCLC Roman Family of Fonts*, p. 36; Apollo.

HOW TO TAME YOUR PHOTOTYPESETTER BY \TeX

Goffredo Haus

C.I.L.E.A. & Istituto di Cibernetica & Te.Co.Graf.

viale Pisa, 10

I-20146 Milano (Italy)

I have implemented \TeX -1100 from MACC on a UNIVAC 1100/80 main-frame computer at C.I.L.E.A. (Segrate, Milano).

I have used a Compugraphic 8600 phototypesetter as a quality output device and various low-cost graphic devices to edit documents.

I have read the CG8600 User Manual and I have found a lot of features which seem to be very useful to produce my documents but, unfortunately, \TeX is not able to control them.

So, I have successfully tried to deceive \TeX : I have built (by means of the FONTPRE program from MACC) a TFM file in which ASCII codes correspond to particular CG8600 commands to be treated as zerowidth, zerodepth, zeroheight characters by \TeX .

The CG8600 driver program SETTEX (from MACC) recognizes the font code and produces a command string suitable for the phototypesetter according to the character code.

For example, I can write white-on-black or *I CAN GET SMALLCAPSITALIC*
FROM SMALLCAPS FONT.

This simple trick, allows me to completely control CG8600 directly from the \TeX source of my document by means of \TeX processor and SETTEX driver program using my special TFM-commands file.

DVI-CODE TO THE HARRIS 7500

Han Noot

Introduction

In 1981 it was decided that at SARA (Stichting Academisch Rekencentrum Amsterdam) \TeX would be installed on a Cyber 175-750 under NOS/BE 552. SARA is a computing center which provides its service to Amsterdam's two universities and to Stichting Mathematisch Centrum (the author's institute) among others. \TeX 's DVI-code was to be processed at SMC, where a Harris 7500 phototypesetter, driven by a VAX 11/780 and a PDP 11/45 are available. (On both computers the UNIX operating system runs.) The filter which converts DVI-code to Harris 7500 code as well as the various programs for the creation and maintenance of font metric files would be implemented and run under UNIX at SMC.

At SARA, one obtained a Cyber version of \TeX from Erik Bertelsen, RECAU, Aarhus, Denmark. We at SMC obtained a \TeX version for the VAX under UNIX from Robert Morris, University of Massachusetts, Boston, USA.

The Harris 7500 phototypesetter

The Harris 7500 phototypesetter is a CRT machine with typefonts stored in digital form on an internal system disc. It produces output on photographic paper or film. Its setting speed highly depends on character size and the number and size of moves between characters; the estimated speed is between 200 and 300 characters/second when the typesetter is driven through a 9600 Baud serial interface. Characters can be explicitly positioned with an accuracy of 0.05 points in both the horizontal and vertical directions.

When used in so-called 'slave mode', the Harris 7500 accepts a set of commands which is an almost ideal target language for the translation of DVI-code. Its main features that interest us here are:

- Horizontal- and vertical move commands which perform displacements relative to the current position in the film plane. They come in two flavors: utilizing absolute machine units (0.05 points) or relative units (1/72 of an 'em' at the current point size).
- Horizontal and vertical position commands relative to the origin. (A user defined point in the left margin.)

- Characters can be typeset with- or without automatic updating of the writing beam position.
- Two character size definition commands. The first fixes the overall pointsize of the characters (in units of 0.1 points), the second can override the pointsize in the horizontal direction as determined by the first. As a result, horizontal- and vertical pointsize can be set independently.
- A rule command for hardware generated rules.
- Slanting hardware to produce slanted versions of non-slanted characters. There are three possibilities: rotation through 9, 12 and 15 degrees. (This feature only gives typographically good results with sans-serif characters.)
- Characters are grouped into fonts of at most 128 characters each.

A naive approach

For every \TeX font we need two different font information files: the familiar \TeX font metric file and a file to be used by the DVI-code translator. This last file contains information on the correspondence between \TeX characters and Harris characters and the width of every character. (Width information is included because the code translator must be able to update the horizontal position after typesetting a character.)

We have designed a symbolic human readable font description from which both font information files can be generated by program. These programs use a data base of height-, depth-, and width values for all the characters that are available on our typesetter. (We obtained this information through the help of the Harris corporation.)

The symbolic font description consists of:

- a header section,
- the character mapping section,
- ligature specifications,
- kern specifications,
- lists of characters of ascending size,
- extensible character definitions,
- a font parameter part.

The header section just contains the font name, design size, character coding scheme and whether the font is a 'NOS- \TeX ' font or a 'UNIX- \TeX ' font (see below).

The character mapping section contains one line for every character in the \TeX font. Such a line has the form:

Nint Fname Cint Sdig Ifrac Pint Hint
in which N, F, C, S, I, P and H are fixed key-letters.
The meaning is the following:

Noct: We are talking about the character with octal number *oct* ($0 \leq oct \leq 177$) from the \TeX font being defined here. (We use octal numbers because they are used in the font examples in appendix F of the \TeX manual too.)

Fname: This character is represented by a character taken from the Harris font indicated by *name*.

Cdec: We use character number *dec* ($0 \leq dec \leq 127$) from this Harris font. (That we have a decimal number here is for easy correspondence with Harris documentation.)

Sdig: The character must be set slanted by 9, 12 or 15 degrees (*dig* = 0, 1 or 2).

Ifrac: The italic correction is *frac* * *character-width*, where *frac* is a decimal fraction.

Pint: The typesetter must set this character at pointsize *int* (specified in units of 0.1 points).

Hint: The horizontal pointsize must be *int*, instead of the value specified by the P-field.

The ligature specification consists of one line of the form:

Noct1 Noct2 Loct3

for each ligature, indicating that \TeX chars *oct1* and *oct2* are to be combined to ligature *oct3*.

Kern specifications are of the form:

Noct1 Noct2 Wfrac

indicating that the kerning value for \TeX characters *oct1* and *oct2* is decimal fraction *frac* of the width of char *oct1*.

Next come lists of characters of increasing size (e.g. a list of left braces). There is one list per line.

Extensible characters are specified by lines of the form:

Noct1 Toct2 Moct3 Boct4 Eoct5

saying that \TeX character *oct1* is extensible, having as top part character *oct2*, as middle part character *oct3*, *oct4* as bottom part and *oct5* as extension component.

This symbolic \TeX -font description together with the data base of Harris character dimensions was thought to contain all the information needed to generate \TeX font metric files by program, but a few somewhat unexpected problems cropped up. They required an extension of the character mapping definition. We deal with them in the next section. Meanwhile we mention that while generating font metric files, our program performs various consistency checks. For instance, it is checked whether

a character is not both declared to be the first of a ligature pair and the first of a kern pair. (By the way, why does \TeX not allow such -albeit theoretical-possibilities?)

The device font files for the DVI-code translator are generated using only the header- and character mapping parts of the symbolic font specification. They contain a machine readable form of the N, F, C, S, P and H fields of the character mapping lines. Furthermore, they contain the width of the characters, both in \TeX 's rsu's and in typesetter units. These twofold width values are used to eliminate rounding errors. As soon as the theoretical (correct) position on a page, measured in rsu's differs from the physical (rounded) position measured in typesetter units (0.01 point) by more than one typesetter unit, the typesetter is instructed to make a compensating move.

A complication with device font files is that we have to generate different files depending on whether we use \TeX running under NOS or \TeX under UNIX. This is so, because our 'NOS- \TeX ' and 'UNIX- \TeX ' use different rsu's, namely a 2^{-16} point rsu respectively a 2^{-20} meter rsu. Hence our DVI-code translator unfortunately has to be aware of which type of DVI-code it is translating. This is moreover so because the two \TeX versions produce slightly different postamble formats in their DVI-code.

Complications

The font description scheme discussed so far would work fine, if not for a number of practical complications, which we will discuss now.

In the first place, Harris characters are generated by digitizing hand drawn art work. The result is that even in a straightforward typefont like Times Roman, not all capitals are of exactly the same height nor do they have equal depth. The same applies to lower case characters: 'p' does not always have its height exactly equal to that of 'r' or its depth equal to that of 'g'. As a consequence, font metric files cannot be generated by directly using character dimensions from the data base (after some unit conversions). That way we would exceed by far the limit of sixteen different depth- and height- values per font as allowed by \TeX . We considered automatic rounding to sixteen different values but had to discard this for the following reason: In the extension font, extension components should have an (almost) exact depth, while the height and depth of for instance the '+' operator may be wrong by quite a lot. A symbol like ' Σ ' is an intermediate case. So rounding must be done under explicit human control.

To make that possible, we introduced into the character mapping lines an optional A-field (A comes from as) which can take one of the following forms:

Aarg, *Adarg*, *Aharg*, or *Ahargdarg*

in which *arg* is either an octal number or a period. If a line reads:

N157 ... A141

it means: take the height and depth of character 157 equal to the height and depth of character 141 from this same \TeX font. *Ah*i*d*j** means: take the height equal to that of character *i* and the depth equal to that of character *j*. *Adj* means: give the character its own height but take its depth equal to that of character *j*. Finally, a period instead of a character number (e.g. A. or Ah. etc.) means: take the character dimension indicated by the period equal to zero.

The next complication arose from the fact that in \TeX 's extension font, components of extensible characters must have a height of zero while the corresponding Harris characters intersect the base line.

* Furthermore, symbols with limits (like Σ) only come out right when they have a height of zero too. As a remedy, we introduced an optional U (**up**) field in the character mapping lines. It is more general than is strictly needed, but can be turned to good use in other circumstances. This U field comes in four forms:

Ud, *Uu*, *Udfrac* or *Uufrac*

meaning the following: *Ud* pushes the Harris character representing the \TeX character so much down, that its top touches the base line. *Udfrac* pushes the character down by an amount equal to *frac* * (*character-height* + *character-depth*), in which *frac* is a decimal fraction. *Uu* and *Uufrac* work like *Ud* and *Udfrac*, but now in the upward direction.

When all this was done, two problems remained: there were (just visible) gaps between occurrences of some extensible character components and between objects like root signs and rules connected to them. The problem with extension characters seems entirely due to the fact that there apparently are minor discrepancies between character height and -depth as given in the typesetter documentation and their real physical dimensions. We introduced an E (epsilon) field for that. It tells the font metric file generating program to take the height and depth of a character a fraction 'epsilon' (in our case 0.02) smaller than

* By the way, we wonder if this fact might not be mentioned explicitly in appendix F of the \TeX manual where the extension font is described. We had to deduce it from \TeX 's behavior when big brackets turned out far too tall. Only later on, a short reference to this phenomena, contained in an appendix of the METAFONT manual, was brought to our attention.

the values contained in the character dimension data base. This produces just enough overlap between extensible character components to make the extensible character come out fine.

The problem with roots (and horizontal braces) is due to the fact that there is sometimes a bit of white space at the left and right of certain typesetter characters where we do not want it. Solution: the R- (reduce white space) field. It has the form:

Rlfrac1rfrac2

(The order of the *lfrac*- and *rfrac* parts may be reversed and one of them may be omitted.) The meaning is: reduce the white space to the left of a character by *frac1* * *character-width* and to the left by *frac2* * *character-width*. (*frac1* and *frac2* are signed decimal fractions.) This is done by having the typesetter perform small horizontal moves before and after setting the character. It solves the problem with roots and furthermore it can be used more generally to make differently spaced fonts out of existing ones.

To sum up: apart from the fields discussed in the previous section, a character mapping line may contain:

- an A field to specify that character dimensions must be taken equal to those of other characters,
- an U field to specify vertical displacement of the character,
- a R field to specify changes in the amount of white space surrounding the character,
- an E field to specify that the character height and -depth must be taken somewhat smaller than the physical values.

A character mapping line **must** contain a N, F and C field; it **may** contain S, I, P, H, U, R, E or A fields. When font metric files and device font files are generated from the symbolic font description, the order of processing of the various fields of a character-mapping line is crucial. It is the order in the summary of the fields just given. First the character height, -depth and -width are obtained from the data base. Height and depth are updated according to the ratio of the character pointsize (P-field) and the pointsize for which the data base values are applicable. The same is done for the width, using the horizontal pointsize if a H field is there, otherwise using the P-field. Next, a width value may be affected by a R-field; height- and depth values are modified by U-, A- and E- field values, in that order. Finally all values are converted to proper units and inserted in the device font- and font metric files.

The rationale for this processing order is the following: First the precise dimensions of the physical character as a result from pointsize specification,

up/down moves etc. are calculated. Only thereafter, deliberate errors may be introduced by E- and A-field values.

The specification of an symbolic font should proceed in corresponding stages. First, the font is specified without R-, E- and A-fields. After inspection of a human-readable version of a font metric file (probably still containing too many height or depth values), A-fields are introduced as needed. Thereafter, on evidence from the typeset output, R- and E fields can be added. E-fields may only be added to character specifications **not** containing A-fields or to **all** characters which have the same height and depth value (taking into account their A-fields). If this restriction is violated, extra height- and depth values may again appear.

Finally, it may be useful to sum up which information from character mapping lines is reflected in font metric files and which in device font files. First the font metric file: The F and C fields together with the character data base determine initial character dimensions. These are then acted upon by eventual P, H, U, R, E and A field values, so all these fields may influence the contents of a font metric file. Furthermore, the I- (italic correction) field is used in this file too. On the other hand, only the F, C, P, H, I, U and R fields determine the contents of the device font file. F, P, H and I specify the state in which the typesetter must be while typesetting the character, the U and R fields describe moves to be performed in the film plane before and after setting the character.

Miscellaneous topics

To assist in the task of creating and maintaining symbolic font description files, Gertjan Vinkesteijn has implemented a kind of special purpose editor. This program interactively asks for the values of the various font description fields, checks the response for correct format, range in which values may lie, etc.

The translator program, which generates typesetter code from DVI code can be called with a number of optional arguments which are used to specify:

- whether 'NOS-DVI'- or 'UNIX-DVI' code must be processed,
- which pages from the document must be typeset (all of them, only individual ones, ranges of pages) and in which order,
- whether some special symbol must be typeset in the margin in order to demarcate pages (to assist in cutting them later on) and if so which one,

- whether all pages must be set to the length of the largest one or to their individual length (which saves some quite expensive photographic material but which may annoy a printer).

As already stated, all programs are implemented under UNIX in the programming language C. There is no fundamental reason for this however. Everything could have been equally well coded in say PASCAL, but we quite heavily use the UNIX system call which makes possible to move freely forward or backward to any position in a file.

Conclusions

The claim made by some that DVI-code can be used to generate code for almost any reasonable output device has not been falsified by our experience. On the contrary, it was quite straightforward to transform DVI-code to code for the Harris 7500. On the other hand, there is quite a lot of 'device dependence' (METAFONT dependence?) hidden in the limitations on font metric files and especially in the peculiarities of T_EX's extension font. In particular, the 'zero-height' requirement for certain characters considerably expands the amount of code to be generated for our typesetter. To typeset **one** extension component **seven** bytes must be sent across the typesetter-interface. The first three generate a downward move (one opcode, two operand bytes), the fourth byte is the actual character and next come three bytes to move up again. (Some of these moves could be optimized away, but only at the cost of complicating the device font files with otherwise unnecessary height- and depth- values of characters.) We wonder whether the reason (unknown to us) for the zero-height requirement is compelling enough to justify the complications it introduces in driving output devices and generating font metric files. Furthermore, could there not be two types of font metric files: one of the current compacted kind and one allowing 128 different values for every character dimension? In spite of its cost in memory, we certainly would have used the latter type for the extension font. Meanwhile we are curious to know, whether there are others who have encountered problems analogous to the ones described here.

To end with a note of optimism, when everything finally worked, we indeed got typeset output which looked quite attractive!

* * * * *
 Site Reports
 * * * * *

NEWS FROM THE T_EX PROJECT

David Fuchs

Just a short note this time. T_EX version 1.0 has been out for over half a year, and it's doing quite well. We'll have a 1.1 available 'soon' that fixes the dozen bugs found since then. Overall, the problems have been obscure enough that I wouldn't even suggest that anyone bother to bring up 1.1, except that we'll be sending L^AT_EX along, which you're sure to want.

The T_EXbook has been out almost as long, and is about to go into a second printing with corrections. Accompanying this TUGboat is a list of errata found before the second printing, and another of errata found after the correction pages for the second printing were sent to the publisher (oh, well). While some of the errors in the first printing were embarrassing, fortunately none of them were disastrous. Prof. Knuth is still offering a reward to the first finder of any error in the T_EXbook, as well as any bug in the code.

The biggest roadblock for T_EX continues to be the availability of fast, inexpensive, high quality printers. The gap is closing though: Imagen has a new model for \$10,000 that prints 8 pages per minute at a very, very pretty 300 dots/inch (it looks much nicer than 2700 output, for instance). This machine is actually shipping, and I'm told that QMS will have a competing model out soon. Let's hope that that will drive the price down. At the other end of the spectrum, John Johnson has been showing off some quite respectable output from a Toshiba printer (180 dots/inch, one half page per minute, under \$2K). Perhaps by the time you read this, these prices will be lower

For those of you maintaining DVI-to-printer programs based on DVIt_ye, please note that we've made two changes to DVIt_ye that should result in better character spacing in special situations. Negative kerns are now rounded correctly, and accents on long words should come out better. Look for DVIt_ye version 2.4 or later on the T_EX 1.1 tape.

The TeXhax mailing list continues to expand to more and more sites. We can now reach BITNET hosts as well as ARPANET, CSNET and UUCP. The main address for add/delete requests is TEXHAX@SU-SCORE.ARPA. One big flurry of recent messages concerned where the point (0,0) goes on

an output page when reading a DVI file. Putting it at the upper left corner of the paper can't be right, because that ends up leaving no top or left margin for the text. On the other hand, nowhere in our documentation does it mention that anyone should expect an inch of space at the top and left to be provided automatically. I would urge everyone to adopt the convention used at Stanford: all DVI-reading programs allow the user to specify an extra top or left margin, but these values default to 1 inch if not explicitly specified.

Another major topic of discussion was how to specify various special graphic effects, and in general how the T_EX community can standardize on \special commands. Perhaps the best way to resolve these issues will be to hold a 'Special Interest' meeting at the coming TUG meeting.

The big activity at Stanford these days revolves around the new Metafont. The program itself is under development on our DEC10 by Prof. Knuth. Although not complete, it's up and running, and is also being used on our Sun machines by the students in the Metafont course currently being taught by Profs. Knuth, Southall and Bigelow. This course is broadcast on the Stanford Instructional TV Network, through which it is picked up by HP in Boise, among others (it will also be available on video tape). We sent the current version of the new Metafont to HP-Boise, and they report that they got it running on their 9836 machines within a few days. I offer this as evidence for the claim that 'change files' for MF.WEB should be nearly the same as the corresponding change files for TeX.WEB. So, everyone should have a pretty easy time getting Metafont going on their machines.

Don't get too excited yet, though. Metafont will undoubtedly be changing rapidly during the next few months, as the course progresses and everyone gains experience with the new system. Even as it stands now, not all of the features that are planned have been coded. So I don't expect to see a Metafont 'version 0' available for distribution before the TUG meeting this summer. A new version of Computer Modern in the new Metafont language will take quite some time, but that shouldn't stop anyone out there from trying their hand at using version 0.

New Metafont differs from the old in every way but name. It's written in WEB for portability; it's much the same size as the new T_EX, and should run on pretty much the same class of machines as T_EX. We've done away entirely with the old raster array. This means that the new Metafont should have much less trouble with very-high resolution fonts (both in speed and character size limitations).

The new Metafont's output also works differently than the old. It's more akin to the way \TeX uses DVI files: the new MF generates 'Generic Font' or GF files for output. A GF file specifies the black and white dots that make up all the characters of a particular font at a given resolution. As with DVI format, GF files are meant to be temporary; the idea is that after you run Metafont to get a GF file, you must then run another program to transform the GF file into a format appropriate for your local printing hardware. For instance, you may have to run a GFtoPXL program if your spooling software looks for PXL-format font files. The old Metafont had to have special code added for each new device that had to be able to interface to \TeX , and even then each time we added a format there was always some new device that needed vertical scan direction rather than horizontal, or horizontal run-lengths, or 16-bit packing rather than 32, etc. The new scheme unbundles Metafont from such considerations, making it less likely that bugs will crop up in different installations, while still allowing each installation to tailor their font storage formats to their particular output devices.

Elsewhere in this TUGboat (page 31), I have extracted the section of MF.WEB that describes GF format, for those of you who are curious or want to start planning ahead. Part of the first distribution of Metafont will be an ancillary program, GFtype, that serves similar purposes as DVItypE: GFtype is meant to be an unambiguous description of GF format, and also to serve as a basis for future GF-to-whatever format conversion programs. GF format is based on horizontal run-length information, which happened to be the most convenient way to output from Metafont's internal data structures. GFtype demonstrates how to create full bitmaps of each character, from which it should be clear that any other format can be generated without much fuss.

Another program that comes along with the new Metafont is GFtoDVI. This program creates proof sheets for fonts under development with Metafont. These proof sheets are similar to the ones you may have seen that were created with the old Metafont: there are vertical and horizontal lines showing the x-height, cap-height, reference point, etc.; dots show where the critical points are that were used to draw the character; the character itself appears in a half-tone grey image, at many times normal size. Anyway, GFtoDVI assumes that there is a very special font, called GRAY, available to output the proof sheets. A GRAY font is device-specific, and this TUGboat also contains a description, on page 35, of the requirements it must meet.

* * * * *

CDC \TeX AT RECAU

Benedict Løfstedt
RECAU, Aarhus Universitet

RECAU's implementation of \TeX 80 on the CDC Cyber under the operating system NOS1 has been used increasingly since the system was released to users in 1982. Courses are held for 30 to 40 users per term and the total of active \TeX users is estimated to be approximately 200. Among other things the system now contains drivers for a Compugraphic MCS8600 phototypesetter, for a NEC spinwriter and for plotters via DISSPLA. In connection with RECAU's change to NOS2 on the CDC Cyber 825 the system has been brought up to date so that it can also be used under this operating system.

The process of installing \TeX 82 under NOS2 has been started. Based on \TeX 82, version 0.9999, are the programs TANGLE, PLtoTF, TFtoPL, DVItypE and \TeX adapted to Pascal3 and NOS2 so that \TeX may carry out a "TRIP" (cf. A test file for \TeX , in \TeX ware).

Proper drivers for \TeX 82 have not been developed yet, and fonts for the type of printer that RECAU uses also need to be developed (RECAU has no raster-based printing device and so cannot use Stanford's font library).

In the implementation CDC 6/12 ASCII-representation is used, and a DEC10 file name of the format: "area.name.ext" is translated to a NOS file name consisting of the characters in "name" followed by the characters in "ext" (not exceeding 7 characters, the rest are truncated). This is a very coarse adaptation to the NOS2 file system, and the system cannot be used by ordinary users in this form.

Another difficulty is the size of the programs: In the TRIP job INITEX requires 320.000₈ CM words; with the original table sizes from Stanford, INITEX cannot be loaded on our machine with 370.000₈ CM words available. A division of the \TeX program into capsules (as has been done with \TeX 80, which can run in down to 140.000₈ CM words) will be necessary before the system can be released to the users.

The adaptation of the abovementioned programs from the \TeX 82 system is described in the form of change-files (cf. The WEB System of Structured Documentation) which is constructed by means of UPDATE. Installation jobs look like standard NOS2 installation jobs. Other CDC installations interested in RECAU's \TeX 82 version may contact the centre.

* * * * *

TeX at Washington State University

Dean Guenther
 Alan Hagen-Wittbecker
 Janene Winter
 Washington State University
 Computing Service Center

Late in April, we began a TeX80 to TeX82 conversion. With the help of David Fuchs, we were able to bring up TeX in both our IBM/CMS and MVS environments in 2 days (whew!) That still left some things like creating more font tables, but we expect by the time this article is published we should be fairly well converted to TeX82.

We have been using TeX80 here at WSU for production work for the last three years on an Amdahl V/8 under MVS. Our proofing was limited to a line printer or an IBM 6670, for *very* rough drafts. The output was typeset on either a Compugraphics 8600 on campus or downloaded to an APS-5 located in Washington's State Printers Office in Olympia, 350 miles away.

The WSU Computing Service Center is a Washington state data processing service center, that provides computing to many city, state, and federal users throughout the state of Washington, and a few outside the state. Any of these users can use TeX, and many are. The Center also supports these users with training and consulting. The main TeXnicians involved are Dean Guenther, Alan Hagen-Wittbecker, and Janene Winter.

One project we've completed within the Center is the *Chief Joseph Dam Cultural Report*. The *Chief Joseph* report describes five years worth of archaeological finds along the Columbia River behind the Chief Joseph Dam. This report, 160 pages long, is authored jointly by the U. S. Army Corp of Engineers and the University of Washington Department of Archaeology. Two PASCAL/VS programs were developed to convert WORDSTAR codes into TeX codes before the document could be formatted.

Another project we've completed is *The Nature and Practice of Biological Control of Plant Pathogens* by R. James Cook and Kenneth F. Baker (ISBN 0-89054-053-5). *Plant Pathogens* is a 550 page book which explains how biological control works in the soil, in crop residue, on the surface of the living plant, and inside the plant. Special macros were created to the specifications of the authors and publisher.

Our heaviest user of TeX is the Washington State

Printer's office, under the TeXnical supervision of Mike Cole and Tom Zamora. The State Printer's office has produced numerous manuals, brochures, and books using TeX. They have, for example, produced two manuals for the Washington State Department of Game—one on the mountain goats of Washington (≈ 100 pages) and the other on mountain sheep (also ≈ 100 pages). Other manuals include a Washington State Department of Transportation bridge specifications manual (750 pages), a real estate manual (300 pages), the liquor board annual report (250 pages), the Washington State SCAN (telephone) directory (170 pages), the Attorney General's manual for contracts, and many others.

The true pioneer of TeX at WSU is the Humanities Research Center (HRC) under the direction of Professor Tom Faulkner. At HRC, Professor Faulkner has been busy on many TeX projects for scholarly publishing. The *Cauda Pavonis* is a biannual publication of material on all aspects of alchemy and Hermeticism. Another biannual publication, *Windrow*, is a vehicle for publishing short stories and poems written by the students and alumni of WSU. *ESQ* is a journal of the American Renaissance devoted to the study of nineteenth-century American literature, religion, philosophical and historical writings emanating from New England, of which Ralph Waldo Emerson is a principal figure.

The project which has been the main thrust of Professor Faulkner's TeX work is a critical edition of Robert Burton's seventeenth-century *The Anatomy of Melancholy*, spearheaded by Professor Nicolas Kiessling. The *Anatomy* was published five times during Burton's lifetime, with a sixth edition published posthumously in 1651. At least five scholars within the last century have devoted their entire lives to creating a critical edition of the *Anatomy*. All have died before they finished.

Professors Faulkner and Kiessling are the first to use a computer in collating the six editions to create a critical edition. (The grand wizard of TeX lent Professor Faulkner some help on the collation problem.) The collation is nearly completed, and the final typesetting for the critical edition is due to begin later this year, or early in 1985. For further information on the collation algorithm used for the Burton project, refer to *Computers and the Humanities* 15 (1981) 163–182.

The Oxford University Press is publishing this critical edition of the *Anatomy*, which is the first time in the history of the Press that they have allowed the typesetting for one of their publications to be

done outside Oxford. The T_EX-formatted *Anatomy* preliminary sample, a partial copy of which follows this article, was sent to Oxford in 1981. Their chief typographer closely scrutinized the sample and in a letter requested two pages of modifications to the format. These changes included modifying the running headline, changing the rule size, decreasing the paragraph indent and paragraph skip, and modify-

ing the kern for capitals “TO” in the chapter title — all of which were accommodated in less than an hour simply by massaging a few of the macros and fonts. The chief typographer also added, “The hyphenation and justification shown on the specimen are very good.” And the sample was done using the **old** version. T_EX82 will look even better.

DEMOCRITUS JUNIOR TO THE READER

Gentle Reader, I presume thou wilt be very inquisitive to know what
 Anticke or Personate Actor this is, that so insolently intrudes upon
 this common Theater, to the worlds view, arrogating another mans name,
 whence hee is, why he doth it, and what he hath to say? Although, as ^ahe
 said, *Primum si noluerō, non respondebo, quis coactus est?* I am a
 free man borne, and may chuse whether I will tell, who can compell me?
 If I be urged I will as readily reply as that *Ægyptian* in ^b*Plutarch*, when a
 curious fellow would needs know what he had in his Basket, *Quum vides
 velatam, quid inquiris in rem absconditam?* It was therefore covered,
 because he should not know what was in it. Seeke not after that which
 is hid, if the contents please thee, ^c*and be for thy use, suppose the
 Man in the Moone, or whom thou wilt to be the Author;* I would not
 willingly be knowne. Yet in some sort to give thee satisfaction, which
 is more then I need, I will shew a reason, both of this usurped Name,
 Title, and Subject. And first of the name of *Democritus*; lest any man
 by reason of it, should be deceived, expecting a Pasquill, a Satyre, some
 ridiculous Treatise (as I my selfe should have done) some prodigious
 Tenent, or Paradox of the Earths motion, of infinite Worlds *in infinito
 vacuo, ex fortuitâ atomorum collisione*, in an infinit wast, so caused by
 an accidentall collision of Motes in the Sunne, all which *Democritus* held,
Epicurus and their Master *Leucippus* of old maintained, and are lately
 revived by *Copernicus*, *Brunus*, and some others. Besides it hath beene
 alwaies an ordinary custome, as ^d*Gellius* observes, *For later Writers and
 impostors, to broach many absurd and insolent fictions, under the name
 of so noble a Philosopher as Democritus, to get themselves credit, and
 by that means the more to be respected*, as artificers usually doe, *Novo
 qui marmoris ascribunt Praxitem suo.* ’Tis not so with me.

^e*Non hic Centauros, non Gorgonas, Harpyasque
 Invenies, hominem pagina nostra sapit.*

^a*Seneca in ludo in mortem Claudii Cæsaris.* [Apocol. 1.]

^b*Lib. de curiositate.* [Mor. § 516E.]

^c*Modò hæc tibi usui sint, quemvis authorem fingito.* Wecker. [Med. syntaxes, ‘Pio
 lectore’.]

^d*Lib. 10. cap. 12. Multa à malè feriat in Democriti nomine commenta data, nobilitatis,
 autoritatisque ejus perfugio utentibus.* [NA 10. 12. 8-9.]

^e*Martialis lib. 10. epig. 4.* [v. 9-10.]

DATA GENERAL SITE REPORT

Bart Childs
Texas A & M University
Department of Computer Science

TEX 1.0 is running on our MV/8000. We have exported it to several sites and are refining the necessary documentation for easy installation.

Our current output is to the QMS Lasergrafix 1200 printer. This is connected as the AOS/VS LQP (letter quality printer) and is a queued device. The DVI translator I wrote by adapting DVItypewriter will soon be contributed for distribution with the TEX distribution tapes. In its current form, DVI_QMS creates a raster image of output that is using fonts which are not downloaded. We generally keep about 16 fonts downloaded. It is written in WEB.

A modification of an earlier Tangled DVI_QMS.PAS is being used on several VAXes. This version does some font caching.

We expect to have our DG driving a Versatec V-80 (no downloaded fonts) by the time you are reading this.

Mark Piwonka, DG-Austin, has a form of DVI_QMS which is driving the G-500 graphics terminal for previewing output.

We are logging each process which uses DVI_QMS and capturing the user_id; document; date and time; and for each font the name, size and count of characters. We intend to use this to create a "best" environment for each installation to "sysgen" their DVI_QMS-like procedure with regard to selecting font caching procedures or queueing documents based on a preferred set of downloaded fonts.

* * * * *

PRIME SITE REPORT

Bart Childs
Texas A & M University
Department of Computer Science

Our progress on this installation has been rather slow. The delays have been due to problems in getting the Prime 750 in the Engineering Experiment Station upgraded to Rev.19.+ of PRIMOS. The communications packages causing the delay have apparently been converted. We expect quick success because Riley Rainey (the person who will do most of it) did a lot of the work on the Data General installation.

The output will be on a QMS Lasergrafix 1200.

* * * * *

UNIX TEX SITE REPORT

Richard Furuta[†]
Department of Computer Science
University of Washington

Since our last report in these pages, the long awaited "final" TEX version 1.0 was released and the accompanying *TEXbook* has become readily available. Possibly as a result, we have noticed a marked increase in the number of requests for the Unix-TEX distribution tape. As of this date of writing, we have directly distributed Unix-TEX to 107 sites (with some receiving more than one version). Over 80% of these sites received Version 1.0 of TEX and most of the remaining sites received the preceding version numbered 0.9999. We are certain that the software has also been redistributed by these primary sites to a significant number of additional sites. Consequently, the Unix-TEX user community must be reaching a quite reasonable size.

Incidentally, we would like to urge all sites running TEXs older than TEX Version 1.0 to get a new copy. The Unix-TEX distribution runs on Berkeley Unix, either 4.1 or 4.2 bsd, on the VAX. To get it, send me a copy of your 4.1 or 4.2 bsd source license (the license from Berkeley, *not* any of the AT&T licenses) and a check for \$50 made out to the University of Washington (no purchase orders, please). Further details were printed in the last two *TUGboats*, so I won't repeat them here—if you don't have a copy, write to me for an information sheet.

Unfortunately, we don't yet have TEX for distribution here for other types of Unix (e.g., System III or System IV) or for computers other than the VAX. However, Textset, Inc., of Ann Arbor, Michigan, has announced a Sun version of TEX (for the Sun running 4.2 bsd). A description of this port follows the Site Report. Contact Textset directly for further information on this product.

Summary of Recent Changes in the Unix-TEX distribution

As of early April, the current version of TEX is Version 1.0. We expect that Stanford will release Version 1.1 soon and it will be included on our tape at that time.

For the benefit of those of you who already have Unix-TEX tapes, I'd like to summarize the major changes we've made to the distribution since September 1983.

[†]This work is funded, in part, by grants from Bell-Northern Research and Northern Telecom, Inc., to the University of Washington.

The largest change was the release of \TeX , Version 1.0, on December 10, 1983. At that time, Howard Trickey also provided new versions of Tangle and Weave that replaced the `pc` input statements with C language routines and we all marveled at the resulting speedup. Consequently, we performed the same transformations to the Symbolics laser printer device driver as well as corrected some bugs. Due to the efforts of Mike Urban of TRW and Ralph Campbell of the University of California at Berkeley, the distribution included a new version of the Imagen output device driver incorporating landscape page printing. We also began including a statement of our guidelines governing the copying of the distribution tape.

We received a number of significant contributions to the distribution during this period. In January 1984, John Hershberger of Stanford provided a \TeX -mode for Gosling's `emacs`. In February, we got Mike Urban's writeup entitled *A Guide to \TeX for the Troff User*. In March, we added a program, written by Mark Senn of Purdue and modified by James Schaad of the University of Washington, that allows one to see DVI files on a BBN BitGraph terminal. If any of you have similar programs for different devices, I'd like to include them in the distribution.

In early March, an experimental version of Leslie Lamport's \LaTeX macro package was included on the Unix- \TeX tape. The distribution still doesn't support this package completely (for example, some fonts are missing) but we expect to have everything put together by the time that the general \LaTeX releases become available. We urge all \TeX sites also to provide \LaTeX .

The most recent change, implemented by Howard Trickey and added to the distribution in late March, was to generalize the mechanism by which Unix- \TeX finds the files it needs to operate. Previously, default directory names were compiled into the system. Now, the values of environment variables are used as a directory path. If the \TeX user doesn't set one of the environment variables, a standard path is used. Unix- \TeX uses environment variables `TEXINPUTS`, `TEXFONTS`, `TEXFORMATS`, and `TEXPOOL` to show where to look for input (and `\read`) files, fonts, formats, and the `tex.pool` file. New versions of `DVItyp` and of the Symbolics laser printer device driver using `TEXFONTS` were also added.

Two Bugs

During this period, Howard found two bugs affecting \TeX compilations. The first one only affects 4.1 sites using our modified `pc` with the `-O` switch (`optimize`) to compile \TeX (in other words, it

is apparently *not* present in the `pc` distributed with 4.2). The problem is present in version 0.9999 and may be present in earlier versions. A simple test to see if your \TeX has the problem is:

```
\tracingall
\edef\A{\noexpand\B}
\show\A
```

A good \TeX will say:

```
> \A=macro:
->\B
```

A bad one will say:

```
> \A=macro:
->\notexpanded:
```

The quick fix is to not use the `-O` option when compiling \TeX (and this change has been included on tapes written after December 1983).

The second bug affects 4.2 sites. On 4.2 sites, Unix- \TeX uses the standardly distributed `pc`. An undocumented option to `pxp` (coincidentally also named `-O`) is used to replace the default arm of the case statement with something that `pc` can compile. The bug is that `pxp` incorrectly eliminates parentheses when handling expressions with unary minuses such as $-(a + b + c)$ which becomes $-a + b + c$. The quick fix is to use `pxp -O -f` which causes expressions to be fully parenthesized (and this change was made to the distribution on March 29, 1984). The actual fix is to correct the code in `/usr/src/ucb/pascal/pxp/rval.c`. The current code reads:

```
case T_MINUS:
  ppop(r[0] == T_PLUS ? "+" : "-");
  a1 = r[2];
  rvalue(r[2], prec(a1) > prec(r) || full);
  break;
```

and the corrected code reads:

```
case T_MINUS:
  ppop(r[0] == T_PLUS ? "+" : "-");
  a1 = r[2];
  rvalue(r[2], prec(a1) <= prec(r) || full);
  break;
```

(changing the second parameter to `rvalue` to cause parenthesizing when precedences are \leq rather than the current code which parenthesizes when precedences are $>$).

Notes and Comments

Talaris Systems Inc., of La Jolla, California, informs us that they now have a version of their QMS output device driver for Unix. Apparently this is a port of their DEC-10 driver and is based on `DVItyp`.

In addition to the Sun \TeX port, Textset, Inc., tells us that they are working on a DVI previewer for the Sun Workstation (displaying \TeX output

on the Sun's screen) that will eventually also allow editing of the \TeX source file. They also have available a output device driver for the Autologic APS-5 series phototypesetters. Textset can also configure this output device driver for Mergenthaler and Compugraphic phototypesetters.

You should contact Talaris and Textset directly for further information about their products.

Finally, let me post a call for volunteers. An increasing demand exists for ports and new implementations of support software for \TeX . If you create a output device driver for a new device or a DVI previewer for a different bit-mapped screen, please consider sending it to us for inclusion on the Unix- \TeX distribution tape.

* * * * *

\TeX 1.0 on SUN WORKSTATIONS

D. L. Rodgers, J. J. Sterken, and P. Grosso
Textset, Inc.

\TeX 1.0 has been ported to a Sun Workstation (Berkeley Unix 4.2) and compiled using Oregon Software's Pascal-II. This system is being used by Textset, Inc., for production typesetting on an APS-5 phototypesetter.

The WEB change file required to tailor \TeX for the Sun Workstation/Oregon Software Pascal-II environment is being distributed on an "as is" cost-recovery basis charge to TUG members—no support or installation assistance is included (the estimated charge is \leq \$200 and much of this is due to the cost of Sun streamer tapes at \$50 each). A fully supported \TeX (1.0) system in the form of object files for Sun or Sun-II Workstations is marketed as a product by Textset, Inc. The change file will be available on quarter- and half-inch magnetic tape.

\TeX 1.0 Performance on Sun Workstations

For purposes of comparison, \TeX on the Sun is roughly 1/25th as fast as \TeX on an Amdahl 5860. That's comparing CPU times though, and while we can get all of the Sun, we get only part of the timeshared Amdahl 5860. Comparing elapsed time on the Amdahl 5860 to Sun CPU time (which is practically equivalent to elapsed time) yields the ratio of 5-to-1.

Here are some timings (all on the Sun):

- Time to Tangle Tangle: 1:10 (1 minute, 10 seconds).
- Time to Tangle \TeX : 8:31.
- Time to compile \TeX : 50 minutes.

- Time to initialize \TeX and get the .fmt file loaded: 10 sec.
- Time/page while doing simple pages—single column, 6.5 by 9.0 inch layout, fairly complicated macro processing going on: 7.5 sec/page.
- Time/page while doing complex pages—8 by 10 inches, dense seven point type, fairly complicated macros: 45.0 sec/page.

For documents being \TeX ed for the APS-5, it takes about a minute for \TeX to get going, and then anywhere from 5–60 seconds per page. Page timings correlate closely with character density. We expect typical math books with average character density to take 10–15 seconds/page.

Program Size

The Pascal-II version of standard \TeX , compiled with optimization enabled and with no runtime checking, takes 159K bytes for the code, 1K for data/constants, and 326K for the runtime stack/storage. For a larger version (Big \TeX) with limits bumped, 554K is required for runtime storage.

So, the grand total is 486K for \TeX and 714K for Big \TeX . The \TeX grand total just sneaks in under 500K so it is conceivable that \TeX might fit in a half-megabyte—though it is unlikely that programs get anywhere near 500K on a half-megabyte machine. However, there is no problem on the Sun—we have a megabyte of main memory (1000K) and an extra 16 megabytes of virtual memory. With just one user doing \TeX processing, no paging is observed.

As an aside: the IBM Pascal/VS compiler produces less compact code than the Oregon Software Pascal-II compiler. The code for \TeX requires 270K on the Amdahl 5860. The runtime stack/storage sizes are comparable.

We're mildly surprised that the runtime stack/storage takes 326K for standard 1.0 \TeX . The MEM array at size 30000 takes only about 120K—we didn't realize that much extra storage was needed and are checking further.

Comments on Pascal-II

The Pascal-II compiler is a very good piece of software, but Oregon Software is not finished working on it. Barry Smith and David Kellerman completed putting up the basic compiler—a job started by David Billstrom. Now Billstrom is back working on it—he seems very good too, he just didn't have the time last fall. Billstrom did most of the preliminary coding for the Unix/68000 version.

Here's what works:

- The basic compiler without runtime error checking. That's exactly what we needed

for our production \TeX system. The port of \TeX to the Sun was made easier because Oregon Software put in all of the Pascal extensions that should have been a part of standard Pascal. For the most part all we did was strip some stuff out of the Pascal/VS change files, and add a little Unix related stuff.

- A few hooks to Unix ... enough, for example, to be able to set things up so \TeX can be invoked via `tex docfile`.

And what doesn't work:

- Run time error checking (subscript out-of-bounds, etc.)
- The Pascal debugger
- The Pascal execution profiler

And what would be nice:

- They don't have any reset/rewrite options corresponding to the `/I` used in \TeX to suppress the initial get when opening a file. That wasn't hard to get around though.
- Can't use `writeln` with packed file of char, `writeln` can only be used with text.
- Can't say `type eight_bits = packed 0..255` as was possible in Pascal/VS.
- With just `type eight_bits = 0..255`, Pascal-II uses 2 bytes instead of of one. It'll use one byte though if you say:

```
type eight_bits = 0..255;
var string: packed array[0..100]
  \qqquad of eight_bits;
```

This turns out not to be a big problem in \TeX because most of the big arrays are defined with `packed`. We didn't worry about throwing in `packed` for all the other `eight_bits` or `ascii` array definitions, so we're using a little more storage than we really need.

- Can't write single bytes to files easily. `packed` file of `eight_bits` doesn't work because `packed` is ignored in that case, so it writes each byte to the file as two bytes, padded with a zero byte on the left. We've changed \TeX to always read TFM's and write DVIs one 4-byte word at a time.
- More hooks to Unix. It currently isn't possible/easy to associate a file with `stdin`, `stdout`, or `stderr`. For the time being, I'm just using `/dev/tty` for `|term_in|` and `|term_out|`—but that doesn't allow redirection of the \TeX output away from the terminal to a file.

We've talked to Oregon Software about these problems. They claim they will work on them over the next few months.

Bottlenecks

We've got the slow Interlogic disk controller on our Sun, but that doesn't seem to be much of a problem since \TeX is CPU bound. For most \TeX runs the CPU time used is 90–95% of the actual elapsed time. And we can't hear the disk doing much seeking.

CPU time is the main bottleneck. We've still not received the Sun-II upgrade (free)—that's supposed to add about 20% to CPU speed.

Actually our main bottleneck is going to be getting data into and out of the Sun given that we don't have a 9-track tape drive, and that we aren't hooked to an Ethernet. The only recourse is the phone—and for binary data like DVI-files, our file transfer program goes at a rate of 80 chars/second over a 1200 baud connection. For a document with a 1.8 million byte DVI-file, that'll take about 6.5 hours! We've got to do something about that.

* * * * *

VAX/VMS SITE REPORT

Monte C. Nichols
Sandia National Laboratory
Livermore, California

There have been a number of new developments for VAX/VMS since the last issue of TUGboat. First, we have Carlos Felippa (Lockheed) to thank for contributing a VERSATEC driver to the VMS \TeX tape distributed by Maria Code (see page 69). Significant contributions have been made to Felippa's driver by S. Marsh (Sachs/Freeman Assoc.) with additional help from I. Haber (NRL), P. Poggio (LLNL) and P. Leary (SNLL). Thanks to all of you for your efforts which have allowed us to get \TeX 82 out on at least one device for VAX/VMS. Also during this period, David Fuchs has upgraded the VAX/VMS \TeX 82 to version 1.0 of \TeX . We owe many thanks to David for seeing that the tape distributed by M. Code has been kept up to date. By the time this issue reaches you, the distribution tape as distributed through M. Code is expected to contain a VMS compatible version of both $\mathcal{L}\TeX$ and $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$. The most recent version of the Stanford VMS tape is available from M. Code. Use the form on page 69 and be sure to specify that you want the Backup format.

As if that isn't enough good news, be sure to see the article by Kellerman and Smith in this issue regarding the \TeX 82 package that Barry Smith and David Kellerman (both formerly at Oregon

Software) are offering. It is impressive that they plan to offer phone support for their products and, if contacted by enough interested VERSATEC users, would be willing to consider providing a true spooler for the VERSATEC.

We are planning to have a VMS/TeX session as part of the upcoming TUG meeting at Stanford. If you have suggestions on subjects that should be discussed that are VMS specific, be sure to contact me.

* * * * *

NOTICE TO VAX/VMS USERS

David Kellerman
Barry Smith

Early this year, we left Oregon Software, forming a partnership to continue our work with TeX and WEB. We are continuing to handle the distribution of TeX software for VAX/VMS systems (Oregon Software will no longer distribute TeX).

Our new address:

Kellerman and Smith
2343 SE 45th Avenue
Portland, Oregon 97215
(503) 232-4799

The following software is currently available:

- **TeX 1.0 (TeX82).** TeX itself is compatible with the version put together by David Fuchs, and offers some additional performance and friendly features (the PLAIN format is really preloaded, for example). The system interface to VMS is cleaner, especially for batch processing. For those who commonly use canned macros, we've packaged INITEX so that you can easily create saved images with new preloaded formats. Included on one 2400' tape are the TeXware programs, the WEB system, all program sources and executable images, and the new AM fonts at twenty-one different magnifications. (We include a copy of The TeXbook.)
- **VERTEX.** This is an all-new WEB language Versatec driver program for model 1200 and V-80 printers. VERTEX uses the VMS Command Language Definition facility and has a bewildering number of options. It can print TeX78 or TeX 1.0 DVI files, with the old or new PXL font files, in either landscape or portrait orientation. Like TeX, you can easily preload your set of common PXL images at your site. VERTEX is provided in executable image format on a 600' tape.

- **IMPRINT.** This is a print spooler that, in various versions, will drive the Imagen printers (IMPRINT-10, 8/300, 5/840, 60/240). IMPRINT uses the VAX/VMS print queueing facilities and is completely compatible with the standard PRINT command. It prints files in Printer, Daisy, Tektronix, Impress, and DVI formats without intermediate processing. Even more than VERTEX, IMPRINT has a ridiculous array of options, as well as several layers of site and user-dependent defaults to simplify commands. IMPRINT is provided in executable image format on a 600' tape, and is also available directly from the Imagen Corporation.

All of the above software includes a user's guide, system manager's installation guide, 90-day unconditional warranty, telephone support for the same period, and domestic shipping via UPS 2nd-day air. International orders will be billed for air-freight costs, and must include a written statement that the software will not be re-exported.

Prices? TeX is \$200 (US), VERTEX is \$400, a package with both TeX and VERTEX is \$500, and IMPRINT (IMPRINT-10, 8/300) is \$1,200 (\$900 for educational institutions). We will be offering support, maintenance, and update services in the near future.

Our current projects are a true spooler for the Versatec, and a VAX/VMS spooling interface to the Compugraphics 8400/8600 photo-typesetters. We're open to other requests.

* * * * *

Fonts

* * * * *

Editor's note: The two documents on the following pages are extracted from a work-in-progress—the new Metafont—and are thus subject to change. Nonetheless, they give the flavor of the new approach to device-independent font definition, and should be useful for (as David Fuchs has put it in his report on page 22) 'planning ahead'.

METAFONT GENERIC FONT FILE FORMAT

1. **Generic font file format.** The most important output produced by a typical run of **METAFONT** is the “generic font” (GF) file that specifies the bit patterns of the characters that have been drawn. The term *generic* indicates that this file format doesn’t match the conventions of any name-brand manufacturer; but it is easy to convert GF files to the special format required by almost all digital phototypesetting equipment. There’s a strong analogy between the DVI files written by **T_EX** and the GF files written by **METAFONT**; and, in fact, the file formats have a lot in common.

A GF file is a stream of 8-bit bytes that may be regarded as a series of commands in a machine-like language. The first byte of each command is the operation code, and this code is followed by zero or more bytes that provide parameters to the command. The parameters themselves may consist of several consecutive bytes; for example, the ‘*boc*’ (beginning of character) command has seven parameters, each of which is four bytes long. Parameters are usually regarded as nonnegative integers; but four-byte-long parameters can be either positive or negative, hence they range in value from -2^{31} to $2^{31} - 1$. As in TFM files, numbers that occupy more than one byte position appear in BigEndian order, and negative numbers appear in two’s complement notation.

A GF file consists of a “preamble,” followed by a sequence of one or more “characters,” followed by a “postamble.” The preamble is simply a *pre* command, with its parameters that introduce the file; this must come first. Each “character” consists of a *boc* command, followed by any number of other commands that specify the “black” pixels of a character, followed by an *eoc* command. The characters appear in the order that **METAFONT** generated them. If we ignore no-op commands (which are allowed between any two commands in the file), each *eoc* command is immediately followed by a *boc* command, or by a *post* command; in the latter case, there are no more characters in the file, and the remaining bytes form the postamble. Further details about the postamble will be explained later.

Some parameters in GF commands are “pointers.” These are four-byte quantities that give the location number of some other byte in the file; the first byte is number 0, then comes number 1, and so on.

2. The GF format is intended to be both compact and easily interpreted by a machine. Compactness is achieved by making most of the information relative instead of absolute. When a GF-reading program reads the commands for a character, it keeps track of several quantities: (a) the current row number, *y*; (b) the current column number, *x*; and (c) the current starting-column number, *z*. These are 32-bit signed integers, although most actual font formats produced from GF files will need to curtail this vast range because of practical limitations. (**METAFONT** output will never allow $|x|$, $|y|$, or $|z|$ to exceed 4095, but the GF format tries to be more general.)

How do GF’s row and column numbers correspond to the conventions of **T_EX** and **METAFONT**? Well, the “reference point” of a character, in **T_EX**’s view, is considered to be at the lower left corner of the pixel in row 0 and column 0. This point is the intersection of the baseline with the left edge of the type; it corresponds to location (0,0) in **METAFONT** programs. Thus the pixel in row 0 and column 0 is **METAFONT**’s unit square, comprising the region of the plane whose coordinates both lie between 0 and 1. Negative values of *y* correspond to rows of pixels *below* the baseline.

Besides *x*, *y*, and *z*, there’s also a fourth aspect of the current state, namely the *paint_switch*, which is always either *black* or *white*. Each *paint* command advances *x* by a specified amount *d*, and blackens the intervening pixels if *paint_switch* = *black*; then the *paint_switch* changes its state. GF’s commands are designed so that *x* will never decrease within a row, and *y* will never increase within a character; hence there is no way to whiten a pixel that has been blackened.

3. Here is a list of all the commands that may appear in a GF file. Each command is specified by its symbolic name (e.g., *boc*), its opcode byte (e.g., 67), and its parameters (if any). The parameters are followed by a bracketed number telling how many bytes they occupy; for example, ‘*d*[2]’ means that parameter *d* is two bytes long.

paint_0 0. This is a *paint* command with *d* = 0; it does nothing but change the *paint_switch* from *black* to *white* or vice versa.

- paint_1* through *paint_63* (opcodes 1 to 63). These are *paint* commands with $d = 1$ to 63, defined as follows:
 If *paint_switch* = *black*, blacken d pixels of the current row y , in columns x through $x + d - 1$ inclusive.
 Then, in any case, complement the *paint_switch* and advance x by d .
- paint1* 64 $d[1]$. This is a *paint* command with a specified value of d ; **METAFONT** uses it to paint when $64 \leq d < 256$.
- paint2* 65 $d[2]$. Same as *paint1*, but d can be as high as 65535.
- paint3* 66 $d[3]$. Same as *paint1*, but d can be as high as $2^{24} - 1$. **METAFONT** never needs this command, and it is hard to imagine anybody making practical use of it; surely a more compact encoding will be desirable when characters can be this large. But the command is there, anyway, just in case.
- boc* 67 $c[4]$ $p[4]$ $min_x[4]$ $max_x[4]$ $min_y[4]$ $max_y[4]$ $z[4]$. Beginning of a character: Here c is the character code, and p points to the previous *boc* command (if any) for characters having this code number modulo 256. (The pointer p is -1 if there was no prior character with an equivalent code.) All x -coordinates of black pixels in the character that follows will be $\geq min_x$ and $\leq max_x$; all y -coordinates of black pixels will be $\geq min_y$ and $\leq max_y$. Finally, z is the leftmost potentially black column in row max_y ; it satisfies $min_x \leq z \leq max_x$. When a GF-reading program sees a *boc*, it can use min_x , max_x , min_y , and max_y to initialize the bounds of an array. Then it sets $y \leftarrow max_y$, *paint_switch* $\leftarrow black$, and initializes its x and z registers to the stated value of z .
- eoc* 68. End of character: All pixels blackened so far constitute the pattern for this character. In particular, a completely blank character might have *eoc* immediately following *boc*.
- skip1* 69 $m[1]$. Decrease y by $m + 1$, set $x \leftarrow z$, and set *paint_switch* $\leftarrow black$. This is a way to produce m all-white rows.
- skip2* 70 $m[2]$. Same as *skip1*, but m can be as large as 65535.
- skip3* 71 $m[3]$. Same as *skip1*, but m can be as large as $2^{24} - 1$. **METAFONT** obviously never needs this command.
- new_row* 72 $u[4]$. Decrease y by 1 and set $z \leftarrow z + u$; then set $x \leftarrow z$ and *paint_switch* $\leftarrow black$. (It's a general way to finish one row and begin another.)
- left_z_83* through *left_z_1* (opcodes 73 to 155). Same as *new_row*, with $u = -83$ through -1 , respectively.
- right_z_0* 156. Same as *skip1* with $m = 0$ or *new_row* with $u = 0$.
- right_z_1* through *right_z_83* (opcodes 157 to 239). Same as *new_row*, with $u = +1$ through $+83$, respectively.
METAFONT generates a *new_row* command only when $|u| > 83$.
- nop* 240. No operation, do nothing. Any number of *nop*'s may occur between GF commands, but a *nop* cannot be inserted between a command and its parameters or between two parameters.
- xxx1* 241 $k[1]$ $x[k]$. This command is undefined in general; it functions as a $(k + 2)$ -byte *nop* unless special GF-reading programs are being used. **METAFONT** generates *xxx* commands when encountering a **special** string; this occurs in the GF file only between characters, after the preamble, and before the postamble. However, *xxx* commands can appear anywhere. It is recommended that x be a string having the form of a keyword followed by possible parameters relevant to that keyword.
- xxx2* 242 $k[2]$ $x[k]$. Like *xxx1*, but $0 \leq k < 65536$.
- xxx3* 243 $k[3]$ $x[k]$. Like *xxx1*, but $0 \leq k < 2^{24}$. **METAFONT** uses this when sending a **special** string whose length exceeds 255.
- xxx4* 244 $k[4]$ $x[k]$. Like *xxx1*, but k can be ridiculously large; k mustn't be negative.
- yyy* 245 $n[4]$. This command is undefined in general; it functions as a 5-byte *nop* unless special GF-reading programs are being used. **METAFONT** puts scaled numbers into *yyy*'s, as a result of **numspecial** commands; the intent is to provide numeric parameters to *xxx* commands that immediately precede.
- char_loc* 246 $c[1]$ $v[4]$ $w[4]$ $p[4]$. This command will appear only in the postamble, which will be explained shortly.
- pre* 247 $i[1]$ $k[1]$ $x[k]$. Beginning of the preamble; this must come at the very beginning of the file. Parameter i is an identifying number for GF format, currently 129. The other information is merely commentary; it is not given special interpretation like *xxx* commands are. (Note that *xxx* commands may immediately follow the preamble, before the first *boc*.)

post 248. Beginning of the postamble, see below.

post_post 249. Ending of the postamble, see below.

Commands 250–255 are undefined at the present time.

define *gf_id.byte* = 129 { identifies the kind of GF files described here }

4. The last character in a GF file is followed by '*post*'; this command introduces the postamble, which summarizes important facts that **METAFONT** has accumulated. The postamble has the form

```

post p[4] ds[4] cs[4] hppp[4] vppp[4] min_x[4] max_x[4] min_y[4] max_y[4]
< character locators >
post_post q[4] i[1] 223's[≥4]

```

Here *p* is a pointer to the byte following the final *eoc* in the file (or to the byte following the preamble, if there are no characters); it can be used to locate the beginning of *xxx* commands that might have preceded the postamble. The *ds* and *cs* parameters give the design size and check sum, respectively, which are exactly the values put into the header of the TFM file that **METAFONT** produces (or would produce) on this run. Parameters *hppp* and *vppp* are the ratios of pixels per point, horizontally and vertically, expressed as *scaled* integers (i.e., multiplied by 2^{16}); they can be used to correlate the font with specific device resolutions, magnifications, and "at sizes." Then come *min_x*, *max_x*, *min_y*, and *max_y*, which bound the values that *x* and *y* assume in all of the characters of this GF file.

5. Character locators are introduced by *char_loc* commands, which contain a character residue *c*, a character device width *v*, a character width *w*, and a pointer *p* to the beginning of that character. (If two or more characters have the same code *c* modulo 256, only the last will be indicated; the others can be located by following backpointers. Characters whose codes differ by a multiple of 256 are assumed to share the same font metric information, hence the TFM file contains only residues of character codes modulo 256. This convention is intended for oriental languages, when there are many character shapes but few distinct widths.)

The character device width *v* is the value of **METAFONT**'s **chardw** parameter, rounded to the nearest integer, i.e., the number of pixels that the font designer wishes the character to occupy when it is typeset within a word.

The character width *w* duplicates the information in the TFM file; it is a *fix_word* value relative to the design size, and it should be independent of magnification.

The backpointer *p* points to the character's *boc*, or to the first of a sequence of consecutive *nop* or *xxx* or *yyy* commands that immediately precede the *boc*, if such commands exist; such "special" commands essentially belong to the characters, while the special commands after the final character belong to the postamble (i.e., to the font as a whole). This convention about *p* applies also to the backpointers in *boc* commands, even though it wasn't explained in the description of *boc*.

6. The last part of the postamble, following the *post_post* byte that signifies the end of the character locators, contains *q*, a pointer to the *post* command that started the postamble. An identification byte, *i*, comes next; this currently equals 129, as in the preamble.

The *i* byte is followed by four or more bytes that are all equal to the decimal number 223 (i.e., '397 in octal). **METAFONT** puts out four to seven of these trailing bytes, until the total length of the file is a multiple of four bytes, since this works out best on machines that pack four bytes per word; but any number of 223's is allowed, as long as there are at least four of them. In effect, 223 is a sort of signature that is added at the very end.

This curious way to finish off a GF file makes it feasible for GF-reading programs to find the postamble first, on most computers, even though **METAFONT** wants to write the postamble last. Most operating systems permit random access to individual words or bytes of a file, so the GF reader can start at the end and skip backwards over the 223's until finding the identification byte. Then it can back up four bytes, read *q*, and move to byte *q* of the file. This byte should, of course, contain the value 248 (*post*); now the postamble can be read, so the GF reader can discover all the information needed for individual characters.

Unfortunately, however, standard PASCAL does not include the ability to access a random position in a file, or even to determine the length of a file. Almost all systems nowadays provide the necessary capabilities, so GF format has been designed to work most efficiently with modern operating systems. But if GF files have to be processed under the restrictions of standard PASCAL, one can simply read them from front to back. This will be adequate for most applications. However, the postamble-first approach would facilitate a program that merges two GF files, replacing data from one that is overridden by corresponding data in the other.

GRAY FONTS FOR METAFONT PROOFS

Gray fonts for METAFONT proofs.

(Preliminary draft: May 2, 1984)

The GFtoDVI program converts a GF file into a DVI file that, when printed, gives a hardcopy proof of the characters. The proof diagrams can be regarded as an array of rectangles, where each rectangle is either blank or filled with a special symbol that we shall call x . A blank rectangle represents a white pixel, while x represents a black pixel. Additional labels and reference lines are often superimposed on this array of rectangles; hence it is usually best to choose a symbol x that has a somewhat gray appearance, although any symbol can actually be used.

In order to construct such proofs, GFtoDVI needs to work with a special type of font known as a “gray font”; it’s possible to obtain a wide variety of different sorts of proofs by using different sorts of gray fonts. The purpose of this memo is to explain exactly what gray fonts are supposed to contain.

The simplest gray font contains only two characters, namely x and a another symbol that is used for dots that identify key points. If proofs with relatively large pixels are desired, a two-character gray font is all that’s needed. However, if the pixel size is to be relatively small, practical considerations make a two-character font too inefficient, since it requires the typesetting of tens of thousands of tiny little characters; printing device drivers rarely work very well when they are presented with data that is so different from ordinary text. Therefore a gray font with small pixels usually has a number of characters that replicate x in such a way that comparatively few characters actually need to be typeset.

Since many printing devices are not able to cope with arbitrarily large or complex characters, it is not possible for a single gray font to work well on all machines. In fact, x must have a width that is an even multiple of the printing device’s unit of horizontal position, since rounding the positions of grey characters would otherwise produce unsightly streaks on proof output. Thus, there is no way to make the gray font as device independent as the rest of the system, in the sense that we would expect approximately identical output on machines with different resolution. Fortunately, proof sheets are rarely considered to be final documents; hence GFtoDVI is set up to provide results that adapt suitably to local conditions.

This understood, we can now take a look at what GFtoDVI expects to see in a gray font. The character x always appears in position 1. It must have positive height h and positive width w ; its depth and italic correction are ignored.

Positions 2–120 of a gray font are reserved for special combinations of x ’s and blanks, stacked on top of each other. None of these character codes need be present in the font; but if they are, the slots should be occupied by characters of width w that have certain configurations of x ’s and blanks, prescribed for each character position. For example, position 3 of the font should either contain no character at all, or it should contain a character consisting of two x ’s one above the other; one of these x ’s should appear immediately above the baseline, and the other should appear immediately below.

It will be convenient to use a horizontal notation like ‘XOXXO’ to stand for a vertical stack of x ’s and blanks. The convention will be that the stack is built from bottom to top, and the topmost rectangle should sit on the baseline. Thus, ‘XOXXO’ stands actually for a character of depth $4h$ that looks like this:

```

blank ← baseline
  x
  x
blank
  x

```

We use a horizontal notation instead of a vertical one because column vectors take too much space, and because the horizontal notation corresponds to binary numbers in a convenient way.

Positions 1–63 of a gray font are reserved for the patterns X, XO, XX, XOO, XOX, . . . , XXXXXX, just as in the normal binary notation of the numbers 1–63. Positions 64–70 are reserved for the special patterns XOOOOO, XXOOOO, . . . , XXXXXO, XXXXXX of length seven; positions 71–78 are, similarly, reserved for the length-eight patterns XOOOOOO through XXXXXXX. The length-nine patterns XOOOOOOO through XXXXXXXX are assigned to positions 79–87, the length-ten patterns to positions 88–97, the length-eleven patterns to positions 98–108, and the length-twelve patterns to positions 109–120.

Position 0 of a gray font is reserved for the “dot” character, which should have positive height h' and positive width w' . When GFtoDVI wants to put a dot at some place (x, y) on the figure, it positions the dot character so that its reference point is at (x, y) . The dot will be considered to occupy a rectangle

$(x + \delta, y + \epsilon)$ for $-w' \leq \delta \leq w'$ and $-h' \leq \epsilon \leq h'$; the rectangular box for a label will butt up against the rectangle enclosing the dot.

All other character positions of a gray font (namely, positions 121–255) are unreserved, in the sense that they have no predefined meaning. But GFtoDVI may access them via the “character list” feature of TFM files, starting with any of the characters in positions 1–120. In such a case each succeeding character in a list should be equivalent to two of its predecessors, horizontally adjacent to each other. For example, in a character list like

53, 121, 122, 123

character 121 will stand for two 53's, character 122 for two 121's (i.e., four 53's), and character 123 for two 122's (i.e., eight 53's). Since position 53 contains the pattern XXOXOX, character 123 in this example would have height h , depth $5h$, and width $8w$, and it would stand for the pattern

XXXXXXXX

XXXXXXXX

XXXXXXXX

XXXXXXXX

Such a pattern is, of course, rather unlikely to occur in a GF file, but GFtoDVI would be able to use it if it were present. Designers of gray fonts should provide characters only for patterns that they think will occur often enough to make the doubling worthwhile. For example, the character in position 120 (XXXXXXXXXXXX), or whatever is the tallest stack of x 's present in the font, is a natural candidate for repeated doubling.

Here's how GFtoDVI decides what characters of the gray font will be used, given a configuration of black and white pixels: If there are no black pixels, stop. Otherwise look at the top row that contains at least one black pixel, and the eleven rows that follow. For each such column, find the largest k such that $1 \leq k \leq 120$ and the gray font contains character k and the pattern assigned to position k appears in the given column. Typeset character k (unless no such character exists) and erase the corresponding black pixels; use doubled characters, if they are present in the gray font, if two or more consecutive equal characters need to be typeset. Repeat the same process on the remaining configuration, until all the black pixels have been erased.

If all characters in positions 1–120 are present, this process is guaranteed to take care of at least six rows each time; and it usually takes care of twelve, since all patterns that contain at most one “run” of x 's are present.

Fonts have optional parameters, as described in Appendix F of *The T_EXbook*, and some of these are important in gray fonts. The slant parameter s , if nonzero, will cause GFtoDVI to skew its output; in this case the character x will presumably be a parallelogram with a corresponding slant, rather than the usual rectangle. METAFONT's coordinate (x, y) will appear in physical position $(xw + yhs, yh)$ on the proofsheets.

Parameter number 8 of a gray font specifies the thickness of rules that go on the proofs. If this parameter is zero, T_EX's default rule thickness (0.4 pt) will be used.

The other parameters of a gray font are ignored by GFtoDVI, but it is conventional to set the space parameter to w and the xheight parameter to h .

For best results the designer of a gray font should choose h and w so that the user's DVI-to-hardcopy software will not make any rounding errors. Furthermore, the dot should be an even number $2m$ of pixels in diameter, and the rule thickness should work out to an even number $2n$ of pixels; then the dots and rules will be centered on the correct positions, in case of integer coordinates. Gray fonts are almost always intended for particular output devices, even though ‘DVI’ stands for ‘device independent’; we use DVI files for METAFONT proofs chiefly because software to print DVI files is already in place.

Editor's note: The following article by Georgia Tobin was set using T_EX82.9999 on Apollo microcomputers, and printed on an Imprint-10 laser printer at 240 dots per inch. Ms. Tobin hopes that it is clear to TUGboat readers that this is a “font-in-progress”. Her article “Computer Calligraphy”, which appeared in vol. 4, no. 1, described the design and construction of a Copperplate script font.

At the deepest level, we could also fiddle with the subroutine definitions in *cmbase.mf* — and of course that would essentially amount to the creation of a new family of fonts.

— DONALD E. KNUTH, *The Computer Modern Family of Typefaces* (1980)

. . . All hell broke loose.

— JOHN MILTON, *Paradise Lost*, i, 917 (1667)

The OCLC Roman Family of Fonts

Georgia K.M. Tobin
OCLC Online Computer Library Center, Inc.

"A complete font design is a complex system . . ." As near as I can tell, everything in Don Knuth's marvelous book, *The Computer Modern Family of Typefaces*, is true; but there is no one statement whose truth I will so readily avow as the preceding sentence. My experience with the METAFONT design of OCLC ROMAN has shown me that a thorough understanding of what is entailed in the complex system that is called a family of fonts is essential to successful METAFONT design. In this report, I will try to show how the complexities of the OCLC ROMAN family of fonts have been incorporated into METAFONT file structure, and to describe the files which fit into that file structure. My intention is to show how one would create a family of fonts and, by implication at least, to offer supporting evidence for the power and subtlety of METAFONT as a graphic design tool.

The creation of workable versions of the fonts outlined was accomplished between March and October of 1983 by a font design group consisting of myself; the fine-tuning of these workable versions is an ongoing process. I had the pleasure of briefly discussing my designs with Dr. Knuth, but the basis of my design technique was gleaned from perusal of *The Computer Modern Family of Typefaces*, and application of the ideas contained therein to the peculiarities of OCLC ROMAN. That, combined with experience gained from my earlier METAFONT design work, combined with an inborn desire to run a tidy operation, gave rise to the OCLC ROMAN file structure as it now stands.

My OCLC ROMAN family of fonts is intended to capture the flavor of Stanley Morison's *Times Roman* in the METAFONT idiom. *Times Roman* is one of the most popular, most readable, and most read typefaces ever designed; it is also, in my opinion, one of the most beautiful. The OCLC ROMAN family of fonts includes such standard *Times Roman* fonts as text, italic, bold, titling, and extended titling. It also includes

a complete symbols font, an italic font suitable for typesetting mathematics, and an extensible font (i.e., one whose characters can grow as required by the formula being set) consistent with Dr. Knuth's Computer Modern fonts. Furthermore, there are fonts suitable for typesetting text in Cyrillic or in Greek.

The first point I want to stress is the breadth of the concept of a METAFONT family of fonts, a system about which the following statements are true:

The fonts which make up a family of fonts are clearly distinguishable from one another, yet all share certain common traits which identify them as members of that family of fonts.

The characters which make up a font are clearly distinguishable from one another, yet all share certain common traits which identify them as members of that font.

The structural components which make up the characters in a font are clearly distinguishable from one another, yet all share certain common traits which permit them to be integrated into a distinguishable character of a distinguishable font of a distinguishable family of fonts.

The single, particular character thus created must be capable of being simultaneously the *specific* representation of a particular character in a particular font of a particular family of fonts, and a *general* template for that particular character at any point size and at any resolution.

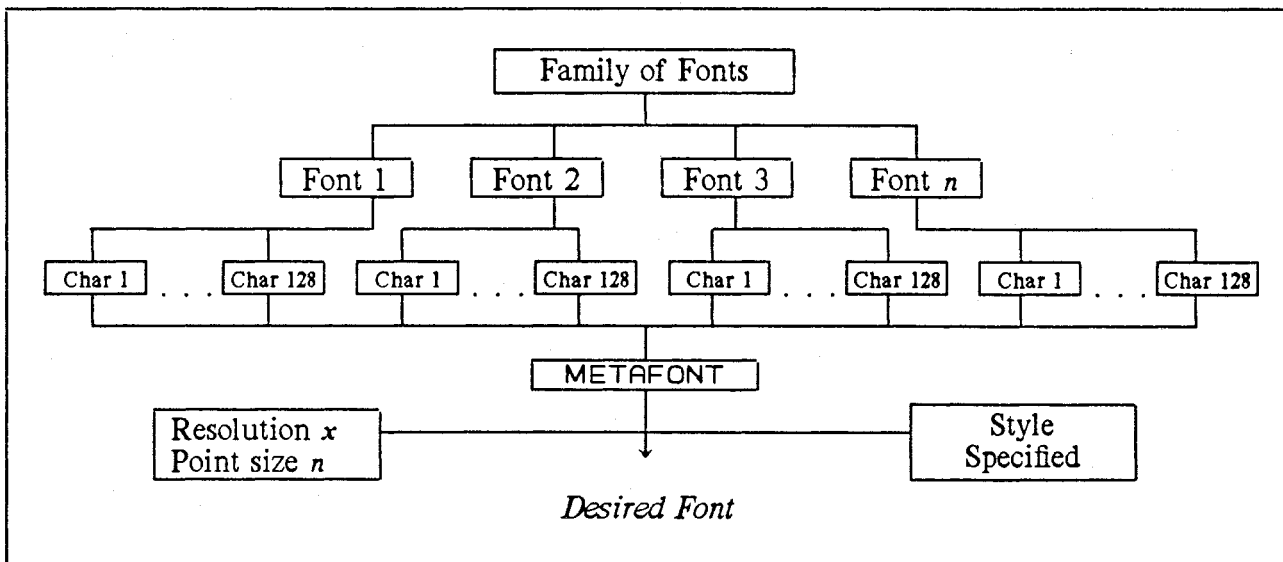


Figure 1. The Hierarchy in a Family of Fonts

My experience with OCLC ROMAN has indicated that this rather tall order may be filled by:

1. Attention to the hierarchical relationship of characters to fonts to a family of fonts in the METAFONT file structure;
2. Attention to the niceties of the particular level in the hierarchy with which a file is concerned; that is, a "font" level file should take care of "font" level detail, not "family" level detail or "character" level detail.

The file structure of OCLC ROMAN corresponds to this hierarchy. *Base.mf* takes care of details at the "family of fonts" level. It contains the definitions and subroutines used by each and every character in each and every font of the OCLC ROMAN family; it also contains subroutines available to each and every character, though these subroutines may not be needed to depict a given form.

Four files take care of details at the "font" level. The first of these is the name of the style of OCLC ROMAN to be constructed, e.g. *text.mf*, *italic.mf*, *bold.mf*, etc. This file assigns values to a number of variables, including: the heights of upper- and lower-case letters; the various pens used in drawing the font; the amount of intercharacter spacing in the font; and the amount of slant in the font. This *font.mf* file calls the appropriate *font_setwidths.mf* file, which allows the designer to assign values for the width of the main body of the character and of the right and left sidebearings for each character in a particular font. *Font.ligatures.mf* provides METAFONT with some critical information on the proper kerning of the font and what ligatures (if any) are used. Equally important, it gives the font's \TeX info. This consists of at least seven parameters which control such critical items as interword spacing, shrink and stretch, and slant per point. The fourth file, *font.switch.mf*, merely specifies the characters that will make up the font.

At the "character" level of detail, we have the character routines for the font. Each of these is a short program (usually twenty to fifty lines of METAFONT code) which describes a particular character in a general way; that is, it describes the character in terms such that METAFONT may draw it at any point size and for any resolution.

We need to consider with some care how these files interact. I shall attempt to do this by starting at the "top," at the family of fonts level of detail, and working my way downward to the individual character level, showing where we obtain the information we require along the way. Let us therefore take a long, hard look at *base.mf*.

The first small bit of *base.mf* contains the METAFONT code which takes care of various house-keeping tasks. These values hold for the entire OCLC ROMAN family. *Mode* is a value assigned by the designer, which must be shared by each font in a family of fonts, because it sets up the values of two factors used throughout METAFONT's computations: *pixelshoriz* and *pixelsvert*. These represent, respectively, the number of pixels oriented horizontally and the number of pixels oriented vertically, and they control the resolution of the output which is produced. The next big chunk of code in *base.mf* is a subroutine called *romanbegin.mf*. This code pertains to the *font* level. It provides METAFONT with such crucial information as what point size of font is to be designed, what sort of grid the font will be designed upon, and what sorts of pens will be used. My insistence upon "sorts of" in the final two clauses of the preceding sentence is not some stylistic penchant for vagueness, but an accurate representation of what *base.mf* does. For though, as I mentioned, *base.mf* is the largest and most complex OCLC ROMAN file read by METAFONT, without additional instructions from the designer it will draw absolutely nothing. (Its saving grace, of course, is that, with the proper additional instructions, it is quite a prolific draftsman.)

As an example, let us consider the way in which the grid upon which OCLC ROMAN characters are designed is defined by *base.mf*. The code for the two routines that accomplish this is shown in Figure 2. We draw six horizontal lines of uniform length at 1) the lowest point which characters with non-rounded descenders reach ($-d$); 2) the baseline upon which all characters sit (0); 3) the greatest height which lower-case characters without ascenders reach (m); 4) the greatest height which nonrounded upper-case letters reach (h); 5) the greatest height which rounded upper-case letters reach (*topp*, i.e. $h + vo$); and 6) the lowest point which characters with rounded descenders reach (*bott*, i.e. $-d - vo$). Only the y value of the baseline is nonnegotiable; all the other y values mentioned in the preceding sentence vary from font to font, and are passed to this routine from whichever *font.mf* file the designer has specified. Each of these six lines starts at $x = 0$ and ends at $x = r$, where r is the width of an individual character which varies from character to character and is passed to this routine from whichever *font_setwidths.mf* file the designer has specified. The grid is completed by drawing two vertical lines from the highest point to the lowest point, one at $x = 0$ and one at $x = r$.

The important point to notice here is not the mechanics of constructing the OCLC ROMAN grid, but the way in which a METAFONT subroutine provides for both the underlying coherence of a family of fonts and the unique qualities of a particular font and character. That is, the grid always extends from $h + vo$ to $-d - vo$, but the particular values of h , d , and vo vary according to design considerations.

Thus far, we have discussed that portion of *base.mf* which pertains to each and every character in a given font. All of the remaining subroutines in *base.mf* provide rote ways to draw certain common

features of individual characters. These move down a level in the hierarchy from the first part of *base.mf*, but provide the same flexibility at that level. That is, they allow for both the individuality of a font and the similarity of a family of fonts.

```

subroutine box:
new offset; offset= 0;
no drawtrace; no proofmode;
new topp, bott, leftt, rightt;
topp=h+vo; bott=-d-vo;
x1 = x3 = x5 = x7 = x9 = x11 = x13 = leftt;
x2 = x4 = x6 = x8 = x10 = x12 = x14 = rightt;
y1 = y2 = -d;
cpen;
l draw 1 . . 2
y3 = y4 = 0;
draw 3 . . 4;
y5 = y6 = m;
draw 5 . . 6;
y7 = y8 = h;
draw 7 . . 8;
y9 = y10 = topp;
draw 9 . . 10;
y13 = y14 = bott;
draw 13 . . 14;
trxy 0;
if italcrr > 0:
x19 = x20 = rightt + italcrr · pixelshoriz;
y19 = topp; y20 = 0;
fi;
trxy pixelsvert · pixelshoriz · slant;
call unitlines

subroutine unitlines:
y1 = topp; y2 = bott; cpen;
new x1, x2; x1 = x2 = 0; draw 1 . . 2; % left
new x1, x2; x1 = x2 = r; draw 1 . . 2; %right

```

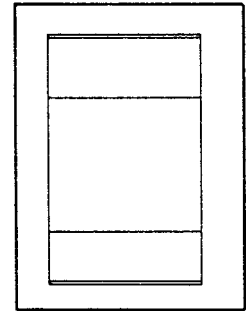


Figure 2. "Font Level" subroutine from *base.mf* and the grid it draws

Let me backtrack for just a bit to enlarge on what I mean. One of the first things that we notice when contemplating the design of a Times Roman-like font is that we will need to draw rather a lot of serifs. In fact, we will need (among others) strictly *horizontally* oriented serifs which extend to either the right or the left of the letter's stem, or to *both* the right and the left of the stem, and strictly *vertically* oriented serifs, which extend either upwards or downwards from the bar, or both upwards and downwards from the bar. We will also need several different sorts of sloped serifs. By describing ways to draw these various serifs in subroutines in *base.mf*, we have a way to both preserve the inherent "Times Roman-ness" of each and to allow a particular serif to look right for a given style of a given letter at a given point size. To better understand how this is done, let us consider *symmhserif*, the subroutine for a horizontally oriented serif which extends to both the right and the left of the letter's stem at either the top or the bottom of a stem.

Whatever routine calls *symmhserif* must pass it four arguments. *Midedge* is the point exactly half way between the leftmost and rightmost points on the serif at its topmost (for a top-of-stem serif) or bottommost (for a bottom-of-stem serif) point. *Joinstem* is the point in the middle of the stem as high as the point at which the serif joins the stem. If $y_{joinstem}$ is less than $y_{midedge}$, *symmhserif* knows that it is dealing with a top-of-stem serif; if $y_{joinstem}$ is greater than $y_{midedge}$, it is dealing with a bottom-of-stem serif. *Stempen* is the name of the horizontal pen with which the stem to which this serif is to be connected is drawn. *Serifwd* is the width of the serif.

Fortified with this knowledge, *symmhserif*'s plodding brain proceeds in the following manner: (See Figure 3 to follow along.)

I will draw the portion of the serif which extends to the left of the stem first. I will do that in the following way. I will define a point 1 on the leftmost edge of the stem and as high as the point *midedge*. I will define a point 2 on the leftmost edge of the stem and as high as the point *joinstem*. I will define a point 3 which lies to the left of the middle of the stem by a distance equal to one half the total width of the serif and as high as the point *midedge*. I will define a point 4 which is the same distance to the left of the middle of the stem as 3 but which is either a minimal pen height *above* the point *midedge* (if I'm drawing a bottom-of-stem serif) or a minimal pen height *below* the point *midedge* (if I'm drawing a top-of-stem serif). I will define a point 5 which lies one half of the way from point 1 to point 3 and is as high as point 3. I will define a point 6 which lies on a concave curve between points 4 and 2; the arc of this curve I know from the value of *brangle*, which I got from this particular font's *font.mf* file.

Now, I am ready to draw. I will use a circular pen one pixel in diameter. I fill in the entire area bounded by the curve from 4 to 2 and the segment from 3 to 1. That takes care of the serif to the left side of the stem.

Now, I use a horizontal pen of size *stempen* to extend the stem to the bottom of the serif.

Now, I am ready to draw the portion of the serif which extends to the right of the stem. I will do that in the following way. I will define a point 7 on the rightmost edge of the stem and as high as the point *midedge*. I will define a point 8 on the rightmost edge of the stem and as high as the point *joinstem*. I will define a point 9 which lies to the right of the middle of the stem by a distance equal to one half the total width of the serif and as high as the point *midedge*. I will define a point 10 which is the same distance to the right of the middle of the stem as 9 but which is either a minimal pen height *above* the point *midedge* (if I'm drawing a bottom-of-stem serif) or a minimal pen height *below* the point *midedge* (if I'm drawing a top-of-stem serif). I will define a point 11 which lies one half of the way from point 7 to point 9 and is as high as point 9. I will define a point 12 which lies on a concave curve between points 10 and 8; the arc of this curve I know from the value of *brangle*, which I got from this particular font's *font.mf* file.

Now, I am ready to draw that side of the serif. I will use a circular pen one pixel in diameter. I fill in the entire area bounded by the curve from 10 to 8 and the segment from 9 to 7. That takes care of the serif to the right side of the stem, and I'm all done.

It is clear enough that having the subroutine *symmhserif* will spare us the task of cranking out all that code every time we want a serif; but it does more, too. Values which are defined in a font's *style.mf* file are used in *symmhserif*, both explicitly (e.g. *brangle*) and implicitly (e.g. *serifwd* is calculated from *standardserif*). This tends to make any serif look as though it were drawn by the same hand; and that lends underlying stylistic coherence to the entire family of fonts. That coherence is really the whole point of *base*.


```

subroutine symmhserif (index midedge)
    (index joinstem)
    (index stempen)
    (var serifwd)
% This draws a symmetrical horizontal serif
cpen;
no proofmode;
new w99; w99 = (wstempen);
call checkpen(99);
rt99 x1 =good99(xmidedge);
rt99 x2 =good99(xjoinstem);
y1 = ymidedge; y2 = yjoinstem;
x3 = x4 = xmidedge - (.5 * r * serifwd);
y3 = ymidedge;
if ymidedge > yjoinstem :
top101 y4 = y3;
else:
bot101 y4 = y3;
fi;
x5 = ½[x3, x1];
y5 = y3;
x6 = 1/brangle[x4, x2];
y6 = 1/brangle[y2, y4];
cpen;
1 ddraw 3 . . 5 . . 1,
4{ x2 - x4, 0} . . 6{ x2 - x4, y2 - y4} . . 2{ 0, y2 - y4} ;
hpen; hpenht 1;
wstempen draw midedge . . joinstem;
lft99 x7 =good99(xmidedge);
y7 = ymidedge;
lft99 x8 =good99(xjoinstem);
y8 = yjoinstem;
x9 = x10 = xmidedge + (.5 * r * serifwd);
y9 = ymidedge;
y10 = y4;
y11 = ½[x9, x7];
y11 = y9;
x12 = 1/brangle[x10, x8]; y12 = 1/brangle[y8, y10];
cpen; 1 ddraw 9 . . 11 . . 7,
10{ x8 - x10, 0} . . 12{ x8 - x10, y8 - y10} . . 8{ 0, y8 - y10} ;

```

Figure 3. A "Character Level" subroutine from *base.mf*

The deepest level in our family of fonts hierarchy is the character level. At this level, the designer uses information from the other files at higher levels described above to compose a description of a character which at once distinguishes it from other characters in the font, enables it to fit in stylistically with other characters in the font, and, in true METAFONT form, makes it adaptable to other point sizes and resolutions.

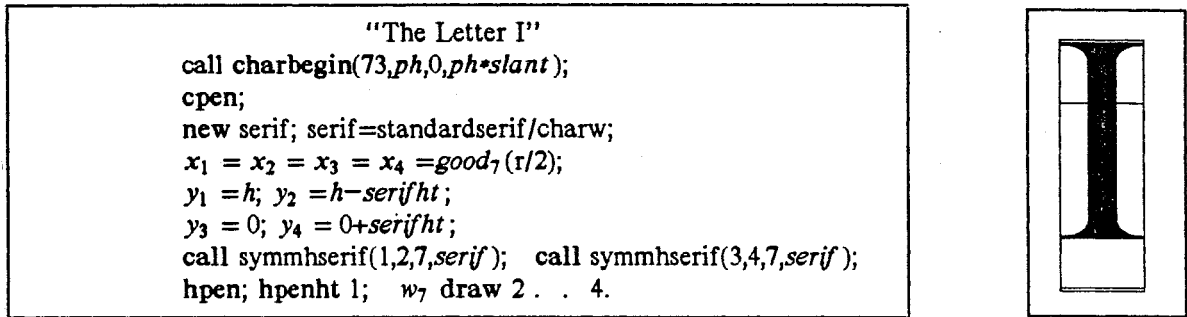


Figure 4. A Character Subroutine, and the Character It Draws.

Consider the routine for a capital "I". This is quite a simple routine, and yet even here, there is a lot going on. *Standardserif* is a value defined in each font's style.mf file; *charw* is a value calculated in *base.mf* using information culled from the *style.ligature* file and the *style.mf* file. By setting the quotient of these two values equal to the width of the serif passed to *symmhserif* not only in the routine for the letter I but for all letters, the designer is assured that the width of the serifs will be uniform throughout the alphabet. Likewise, the pen w_7 is defined in *base.mf* using values obtained from the *style.mf* file; and we know with confidence that the pen will be the same for all letters that share the same definition of w_7 , that is, all letters in a given font.

The upshot of all this interdependence among the various files which make up the font family OCLC ROMAN is this: by carefully designing a basic collection of character forms and carefully setting up the subroutines which support those letter forms, we can exploit the inherent flexibility of our system to produce a limitless number of variations on the typographic theme we have chosen to produce a rich and varied but stylistically coherent family of fonts through skillful manipulation of our files. All of which sounds stirring enough; but we must eschew such generalities for a good long look at how this translates into the nitty-gritty of real fonts.

We first produce a standard text font:

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	θ	Λ	Ξ	Π	Σ	Τ	"0x
'01x	ϕ	ψ	Ω	ff	fi	fl	ffi	fff	
'02x	ı	ı	`	'	˘	-	-	˚	"1x
'03x	˙	β	æ	œ	o	Æ	Œ	Ø	
'04x	ˆ	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	i	=	ı	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	["]	^	˙	
'14x	'	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	"	~	..	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 5. OCLC ROMAN text 10 point.

By establishing another *style.mf* file which sets *slant* to 0.20 rather than 0, but using those same characters, we get:

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	θ	Λ	Ξ	Π	Σ	Τ	"0x
'01x	Φ	Ψ	Ω	ff	fi	fl	ffi	fff	
'02x	ι	ϰ	·	˘	˙	˚	˛	◦	"1x
'03x	⸣	β	æ	œ	ο	Æ	Œ	Ø	
'04x	ˉ	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	i	=	ι	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	["]	^	.	
'14x	'	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	~	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 6. OCLC ROMAN slanted text 10 point.

We need to do a bit more work to obtain a true italic, which looks like this:

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	θ	Λ	Ξ	Π	Σ	Τ	"0x
'01x	Φ	Ψ	Ω	ff	fi	fl	ffi	fff	
'02x	ι	ϰ	·	˘	˙	˚	˛	◦	"1x
'03x	⸣	β	æ	œ	ο	Æ	Œ	Ø	
'04x	ˉ	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	i	=	ι	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	["]	^	.	
'14x	'	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	~	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 7. OCLC ROMAN Italic 10 point.

All we needed to do this was to create a *style.mf* file which establishes a *slant* value of 0.20 and defines a set of slightly thinner pens than those used for text. We were able to use the same character forms by and large, except for the lower case letters. We also needed to make some changes in the character widths. However, the bit of extra work pays off; for by running these character routines with a *style.mf* file identical to the one used for italic except that it sets *slant* to 0, we get an unslanted italic font.

Of course, not all the various fonts we want are produced with so few changes from standard text. Boldface, for example is *not* simply text written with broader pens; there are fundamental differences in several of the character forms, and the proportions in bold lettering are *not* merely the proportions of text scaled up. Happily, though, our file structure enables us to describe all these differences in the *style.mf*, *style.ligatures.mf*, and *style.setwidths.mf* files, and to use the subroutines in *base.mf*.

This same state of affairs is what helps to make some fonts, which require almost a complete new set of character forms, to nevertheless have an OCLC ROMAN look to them. Consider, for example, the following:

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Г	Δ	Θ	Λ	Э	Π	Σ	Т	"0x
'01x	Ф	Ψ	Ω	Ю	Ж	Ь	Ъ	Э	
'02x	№	«	`	'	˘	-	-	˙	"1x
'03x	,	»	И	Ю	Ж	Ь	Ъ	Э	
'04x	˘	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	i	=	¿	?	
'10x	Й	А	Б	Ц	Д	Е	Ф	Г	"4x
'11x	Х	И	Я	К	Л	М	Н	О	
'12x	П	Ч	Р	С	Т	У	В	Щ	"5x
'13x	Ш	Ы	З	["]	^	˙	
'14x	‘	а	б	ц	д	е	ф	г	"6x
'15x	х	и	я	к	л	м	н	о	
'16x	п	ч	р	с	т	у	в	щ	"7x
'17x	ш	ы	з	-	-	"	~	..	
	"8	"9	"A	"B	"C	"D	"E	"F	---

Figure 8. OCLC ROMAN compatible Cyrillic 10 point.

Despite their superficial dissimilarity, 'I' and 'И' are more nearly akin than 'I' and 'I'. (The second *I* in the preceding sentence is from Dr. Knuth's Computer Modern Roman alphabet.) The first pair share pens; they share serif routines; they share *h*-heights. The second pair share none of those things, and strike us as similar because they are, coincidentally, representations of the same English character.

The font family resemblance becomes more difficult to descry in non-text fonts like the math symbols font or the extensible symbols font.

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	()	[]	L	J	Γ	∏	"0x
'01x	{	}	<	>			/	\	
'02x	()	()	[]	L	J	"1x
'03x	Γ	∏	{	}	<	>	/	\	
'04x	()	[]	L	J	Γ	∏	"2x
'05x	{	}	<	>	/	\	/	\	
'06x			Γ	∏	L	J	·	·	"3x
'07x	f	g	l	j	{	}	·	·	
'10x			·	·	<	>	□	□	"4x
'11x	f	g	o	o	⊕	⊕	⊗	⊗	
'12x	Σ	Π	f	U	∩	∪	∧	∨	"5x
'13x	Σ	Π	f	U	∩	∪	∧	∨	
'14x	□	□	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ	"6x
'15x	[]	L	J	Γ	∏	{	}	
'16x	√	√	√	√	√		Γ		"7x
'17x	↑	↓	⌒	⌒	⌒	⌒	↑	↓	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 9. OCLC ROMAN compatible Extensible Symbols 10 point.

Nevertheless, it is underlying family resemblance among the 600-odd routines which compose the OCLC ROMAN family which enable apparently dissimilar fonts to work together harmoniously. Witness the following:

OCLC ROMAN has a font which corresponds to every font in the file plain.tex used by T_EX82. In fact, we at OCLC are using a version of plain.tex with the font families redefined to OCLC ROMAN.

The OCLC ROMAN family of fonts encompasses a complete range of digital typefaces suitable for a broad spectrum of uses from regular text to scientific and mathematical material.

Text
Italic
 Medium Bold
Medium Bold Italic
 Bold
Bold Italic
 Extra Bold
Extra Bold Italic
 TITLING
 HEAVY TITLING
 EXTENDED TITLING
Slanted Text
 Unslanted Italic
 Monospace

The complete Roman alphabet is available in text, medium bold, bold, and extra bold weights, with corresponding italics. Roman small caps ("titling" faces) are available in text and bold weights, as well as in an extended version. Specialty versions of the Roman alphabet include slanted text, unslanted italic and a monospaced face.

Пролетарии
 всех стран
 соединяйтесь!

Non-Roman alphabets include Greek, Cyrillic for modern Russian, Azeri, and a "Slavic" character set containing the letters used in Bulgarian, Byelorussian, Macedonian, Serbian and Ukrainian; the latter two are available in text weight in both standard and slanted versions.

The sum of $\begin{Bmatrix} 1+2 \\ 2+2 \\ 3+2 \end{Bmatrix}$ is $\begin{Bmatrix} 3 \\ 4 \\ 5 \end{Bmatrix}$.

$$\pi(n) = \sum_{m=2}^n \left[\left(\sum_{k=1}^{m-1} \lfloor (m/k) / \lceil m/k \rceil \rfloor \right)^{-1} \right].$$

$$\frac{f(x + \Delta x) - f(x)}{\Delta x} \dots f'(x) \quad \text{as } \Delta x \dots 0.$$

OCLC ROMAN's scientific and mathematical fonts include math italic in text and bold weights, a font of common mathematical signs and symbols, including an upper case script alphabet, and a font which produces symbols of arbitrary size.

All of the OCLC ROMAN typefaces are available in sizes from 4 to 96 point and for use on devices with 100, 240, 300 or 480 dpi.

You can go from  to z.

* * * * *

"small" T_EX

* * * * *

Send submissions to:

Lance Carnes
163 Linden Lane
Mill Valley, CA 94941
(415) 388-8853

Last issue (Vol. 4, No. 2, September 1983) this column carried an appeal for interest in and funding for a PC-T_EX. I received several replies and, while the interest is high, the level of funding is not. I was able to justify the purchase of an IBM PC for business use, and will pursue porting T_EX to it as time allows.

The ideal system would run on the PC (with sufficient speed, I hope, to compile a page or two per minute), and print a "proof" copy on a graphics printer (such as an Epson MX-80). This would provide a convenient, low-cost work station for T_EX document preparation. Since many companies, schools and individuals already own systems,

there would no additional expense for hardware. If higher-quality output is required, documents could be transferred from the microcomputers to systems that can compile and print T_EX files on high-resolution devices. Let's hope it will work, since this capability could vastly increase the popularity and use of T_EX.

I know of at least one experiment running T_EX on a PC-XT with the IBM370 emulation. Alan Spragens of SLAC and David Fuchs downloaded a VM T_EX to the PC and it worked without a hitch! The bad news is that it took 20 minutes to compile a page. This is due mainly to the 370 emulation; it would undoubtedly run faster with native code.

A new "small" T_EX, on a Synapse (a multiple-68000-based system), was implemented by Dick Wallenstein at Comcon. A very patient fellow, Dick managed to bring up his version with a Pascal that had no REAL (floating-point) type.

We have recently had two reports of the successful porting of T_EX onto Apollo (68000) systems, one from Yale and the other from COS Information Systems in Montréal.

David Fuchs says that the Corvus and Sage systems are not currently robust enough to support T_EX. We will leave these systems on the "wish" list.

Whoever has the Masscomp port, please get in touch so that we can include the contact information in the table on this page.

"small" T _E X implementations				
Manufacturer	Processor	T _E X version	Processor time per page	Company and contact
Hewlett-Packard 3000	16-bit	T _E X82	10-30	TeX ϵ T, Lance Carnes
Hewlett-Packard 1000	16-bit	T _E X82	10-30	JDJ Wordware, John Johnson
DEC PDP-11/44	16-bit			
Plexus, Onyx	Z8000	T _E X80	10-20	TYX, Dick Gauthier
IBM PC	8086/8			
Apollo	M68000	T _E X82	2-10	OCLC, Tom Hickey; Yale, ?; COS Information, Pierre Clouthier
Hewlett-Packard 9836	M68000	T _E X82	6-10	HP Boise Div., Jim Crumley
Sun	M68000	T _E X82 ¹		Textset, Jim Sterken
Corvus	M68000	T _E X82 ³		
Cyb	M68000	T _E X82 ¹		Texas A&M, Norman Naugle
Apple Lisa	M68000	T _E X82 ²		
Masscomp	M68000	T _E X82 ¹		
Sage	M68000	T _E X82 ³		
Synapse	M68000	T _E X82	10-30	Comcon, Dick Wallenstein

¹ in progress or recently completed² hopeful³ currently unimplementable

```
* * * * *
      Warnings & Limitations
* * * * *
```

\relax and Watch the Numbers

Thanks to the T_EXhax participants and Mike Spivak for these ideas.

When T_EX encounters a ⟨number⟩, it will keep parsing until it recognizes a non-number. This can lead to some unexpected situations.

Mr. Ogawa (Ogawa%SLACVM.BitNet@Berkeley) thought he had a problem with the @ character:

```
% UNLOCK TEST - test TeX unlock feature
% this character can now be used
%                               in macro names
```

```
\def\unlock{%
  \catcode\@=11%
}%
\unlock\rOggedbottomtrue\bye
```

yielded

```
This is TeX, VM/CMS Version 1.0 [SLAC]
(preloaded format=plain 84.2.17)
      19 APR 1984 21:07
```

```
**&PLAIN unlock.test
(unlock.test
! Undefined control sequence.
1.5 \unlock\r
      Oggedbottomtrue\bye
```

?

[1]

```
Output written on unlock.dvi
      (1 page, 224 bytes).
```

A slight modification gave the result he originally expected:

```
\def\unlock{%
  \catcode\@=11%
  \relax
}%
\unlock\rOggedbottomtrue\bye
```

```
This is TeX, VM/CMS Version 1.0 [SLAC]
(preloaded format=plain 84.2.17)
      19 APR 1984 21:13
```

```
**&PLAIN unlock.test
(unlock.test
No pages of output.
```

A different variation on the second line would also have done the job:

```
\catcode\@=11 %
```

(Note that the problem surfaces with the expansion, not with the definition, of \unlock, so one could also

avoid the problem by modifying the last line to say \unlock\relax\rOggedbottomtrue....)

Arthur Keller offers the following explanation: "When T_EX sees the "\catcode\@=11%" it is still parsing a number. For example, you could have said

```
\def\unlock{%
  \catcode\@=1}%
\unlock1\rOggedbottomtrue\bye
```

This would mean \catcode\@=11 (the two ones are put together). The \relax causes the mouth to realize that it has completed parsing a number and that allows it to pass the mess to the stomach, and the \catcode changes. In general, you should use \relax to terminate such parameters when it isn't self delimiting (such as 5pt)."

A related situation has been pointed out by Mike Spivak:

```
\if ... \advance\foo by \the\fam \fi
```

Since \the\fam expands to a number, T_EX will absorb the \fi looking for the end of the number, and if digits follow \fi they will be included in the value by which \foo is \advanced.

For the official explanation, see *The T_EXbook*, pages 208 and 269–270. There it is recommended that "for best results, *always put a blank space after a numeric constant*; this blank space tells T_EX that the constant is complete, and such a space will never 'get through' to the output." But only one space, and note the comment on aesthetics vs. efficiency.

A final word from Arthur Keller: "You should be very careful to avoid extra spaces in macros lest they creep into the output when in horizontal mode. Using \relax to delimit parameters is consequently much safer."

Barbara Beeton

THE HP T_EX MACROS

HEWLETT-PACKARD CO.

Susan Daniels

1 Introduction

This set of macros was written as part of HP T_EX, Hewlett-Packard's software package designed to facilitate the use of T_EX on the HP Series 200 Desktop Computers. The HP T_EX program can be run on either an SRM based, or a stand-alone (LIF) system, and requires 1.25 Mbytes main memory and a minimum of 15 Mbytes mass storage. It uses an HP 2688A Laser Printer.

The HP2688A is a page printer which uses scanning electrophotograph technology to print computer output on single sheet paper. The 300 dots per inch resolution, high contrast and speed, plus sophisticated character printing, allow a high degree of application flexibility. Unique features include 90° rotation of page print, multiple pages of data output per printed sheet, and graphics capability. The print rate is approximately 12 pages per minute.

The HP T_EX macro package is designed to supplement and enhance Plain T_EX by providing a higher level of functionality and an example of a macro package oriented toward a simple set of user formatting tasks. It also includes macros that simplify the use of the special features of the HP2688A Laser Printer.

The functions selected for HP T_EX macros are general formatting functions. They are not oriented toward a particular application, however the needs of manual production were considered in order to obtain a useful set of commands.

The HP T_EX macros are intended to be a superset of the Plain T_EX macros, and fully compatible with them. Whenever possible, Plain T_EX functions are unchanged in HP T_EX. In the few instances it was necessary to redefine a Plain function, the new definition remains consistent with the intent of the Plain definition. Hence, the user of HP T_EX has access to the Plain T_EX macros and can use the *T_EXbook* and its exercises and examples.

The HP T_EX Macros will soon be available on the Stanford Distribution Tape.

On syntax lines, when curly braces "{...}" are shown, they are required in the source file, text between angle brackets "<...>" identifies a parameter variable, and the character "□" represents a required space.

2 Document Formatting Macros

The following HP T_EX commands control the basic style and formatting of a document:

2.1 Page Layout

The HP T_EX defaults for page layout are the same as they are in the T_EX program. That is, all margins are set to approximately one inch from the edges. (Plain T_EX sets `\vsize=8.9in` and `\hsize=6.5in`.) Headings and footings are printed **outside** those margins.

The following macros affect the running head and the footing of each page. The heading and footing lists are expanded during the output routine so the T_EX commands `\firstmark`, `\topmark` and `\botmark` are compatible here.

```
\centerheading{(horizontal list)}
\leftheadings{(horizontal list)}
\righthedings{(horizontal list)}
\outsideheading{(horizontal list)}
\insideheading{(horizontal list)}
```

These macros accept text which is to be placed at the top of every page. The first three macros place the argument in the center, left, or right of the page respectively. `\outsideheading` and `\insideheading`, if specified, override the left and right headings. On odd pages, the inside heading will appear on the left, outside on the right. Even pages are the opposite. (This feature is helpful when the output is to be used as two-sided copy.)

```
\centerfootings{(horizontal list)}
\leftfootings{(horizontal list)}
\rightfootings{(horizontal list)}
\outsidefootings{(horizontal list)}
\insidefootings{(horizontal list)}
```

These are like the above commands, only they are for footings at the bottom of each page. For example, if page numbers are desired in the left side of the bottom of each page, type: `\leftfootings{\folio}`

`\outsidefootings` and `\insidefootings` will produce similar results as `\outsideheading` and `\insideheading` (see above). The default in HP T_EX is `\centerfootings{\folio}`. This produces page numbers at the center of the bottom of each page.

```
\noheading
\nofooting
```

These macros turn off headings or footings for the current and all successive pages until the `\resumefootings` or `\resumeheading` command is used.

```
\suspendheading(integer)
\suspendfootings(integer)
```

These macros suspend headings or footings for the specified number of pages or until the use of a `\resumeheading` or `\resumefootings`.

```
\resumeheading
\resumefootings
```

These macros undo the effects of the `\suspendheading`, `\noheading`, `\suspendfootings` and `\nofooting` macros above.

2.2 Paging

`\newpage`

This macro forces a page eject if not on a new page.¹

`\oddpage`

This macro causes a page eject, and if the current page is an odd-numbered page, leaves an extra blank page so that the following text is guaranteed to begin on an odd-numbered page.

`\evenpage`

This macro is similar to `\oddpage`, but the following text will appear on an even rather than an odd numbered page.

The Plain `TEX` command, `\pageno` will set the current page number to the specified page number. (See “Some Useful `TEX` Commands” in Chapter 2 of this manual.)

2.3 Paragraph Style

As `TEX` accepts text from the input file, the text is formatted into paragraphs. The following commands can be used to control the shape of the paragraphs.

`\inset=(dimension)`

Specifies a general amount of indentation to be used with itemized lists, notes, warnings and indent blocks. Default is 0.5 inch.

`\start{indent}`

`\finish{indent}`

These commands are used to indent the left margin by the `\inset` dimension. The right margin is unaffected by this block.

`\indentSPACE=(dimension)`

This command assigns the indentation value for the first line of all succeeding paragraphs when `\indentstyle` is in effect. The default is 20 points.

`\indentstyle`

This causes the first line of each paragraph to be indented (the amount of the indentation is assigned by the `\indentSPACE` command). This is the default paragraph style.

`\noindentstyle`

This causes the paragraphs to be formatted with no indentation. The nominal spacing between paragraphs is 5 points greater without indentation than when `\indentstyle` is being used.

- NOTE -

Spacing between paragraphs can be user specified by using the `\parskip` command.

¹ If `\newpage`, `\oddpage`, or `\evenpage` occur at the bottom of a full page, they will cause an extra page eject. If this happens, insert an `\eject` just before the page command.

2.4 Itemized and Bulleted Lists

The following macros are aids in producing lists. The various list commands cause indentation at various levels and use various tokens (numbers, letters, dots, dashes, *etc.*) to the left of the first line of the listed item to set off the list entry.

```
\numbereditems
\lettereditems
\Lettereditems
\romanitems
\Romanitems
\squareditems
\dotteditems
\dasheditems
```

These macros initialize the tag allocation macro `\itemtag`. Default is `\numbereditems`.

```
\numberedsubitems
\letteredsubitems
\Letteredsubitems
\romansubitems
\Romansubitems
\squaredsubitems
\dottedsubitems
\dashedsubitems
```

These macros initialize the tag allocation macro `\subitemtag`. Default is `\letteredsubitems`.

```
\itemtag
```

This macro causes a number, letter, roman numeral, square, dot or dash to be printed as the item specifier depending upon what initialization has taken place (see above).

```
\subitemtag
```

Similar to `\itemtag` but pertains to subitems.

```
\square
```

This macro prints a .4em-by-.4em square (■). This macro can be used at any time a horizontal list is being built (such as the middle of a paragraph), but it is particularly useful as the argument for the `\itemlist` macro. (This command produces the same symbol as TeX's `\bull` command.)

```
\dott
```

This macro prints a solid round dot (•). (Same as Plain TeX's `\bullet`.)

```
\emdash
```

Sets a horizontal rule one "em" long (—).

```
\itemlist{⟨horizontal list⟩}
```

This macro introduces an item of a list by indenting both left and right margins by the ⟨dimension⟩ specified in the last `\inset` command. The ⟨horizontal list⟩ is inserted within braces to specify the token which is to set off the text. If the token has already been specified (using `\dotteditems`, `\Romanitems`, *etc.*), or the default `\numbereditems` is acceptable, then the command `\itemtag` can be inserted here. The text of the item follows. (The text does **not** need to be enclosed within braces.) Indentation continues only for the duration of one paragraph (if more than one paragraph is desired in an item, `\itempar` should be used). The use of `\itemlist` resets the subitem tag.

`\subitem{⟨horizontal list⟩}`

This macro is similar to `\itemlist`, but the indentations are twice as large as for `\itemlist`. This can be used to indicate a second level of list. (See `\itemlist` for contents of the `⟨horizontal list⟩`).

`\itempar`

`\subitempar`

These macros are used to start new paragraphs within an item or subitem respectively.

`\enditems`

This commands properly ends an itemized list by resetting the item counter and appending `\bigskip` glue.

`\itm`

Similar to `\itemlist{⟨itemtag⟩}` except that a period is appended to the tag if it is a letter, number or roman numeral. With this command, the item tag must be specified ahead of time using the command `\dotteditems`, `\lettereditems`, *etc.*, unless the default (`\numbereditems`) is acceptable. Again, `\itm` will **not** accept any `⟨horizontal list⟩` as an item tag specifier.

`\sitm`

Similar to `\itm` but pertains to subitems. Like `\itm`, `\sitm` will **not** accept any `⟨horizontal list⟩` as a subitem tag specifier. The subitem tag must be specified beforehand, unless the default (`\letteredsitems`) is acceptable.

- NOTE -

If you enter a local block structure prior to setting an itemized list, and want to exit that structure immediately after your last item, then you must first be sure that you are in **vertical mode**. Otherwise, the last item may not be indented properly.

Three ways to accomplish this are by using `\enditems`, `\vskip`, or `\par`.

2.5 Line Specifier Macros

The content of individual lines can be controlled, and they can be centered or justified using the following commands.

`\centerline{⟨horizontal list⟩}`

This takes the text within the brackets and centers it between the margins. The line does not count as a paragraph, so the normal interparagraph space is not inserted above the line. If this command follows a paragraph, it is usually appropriate to precede it with a `\vskip`. This command has been redefined from Plain `TEX` in that it uses the HP macro `\lline` (described below), so `\leftskip` and `\rightskip` values are taken into account.

`\lline{⟨horizontal list⟩}`

This is very similar to the plain `TEX` `\line` command in that it creates a horizontal box that is the width of the current `\hsize`, except that `\lline` takes into account the values of `\leftskip` and `\rightskip` if they have been specified. In other words `\lline` stays within the current margins. The `⟨horizontal list⟩` is the contents of the horizontal box.

`\leftline{⟨horizontal list⟩}`

This left justifies the contents of the `⟨horizontal list⟩`. The comments about vertical spacing above apply here, also. This command has also been redefined to use `\lline` (see `\centerline`, above).

`\rightline{⟨horizontal list⟩}`

This right justifies the contents of the ⟨horizontal list⟩. The comments about vertical spacing above apply here, also. This command has also been redefined to use `\lline` (see `\centerline`, above).

`\raggedright`

The `\raggedright` macro causes paragraphs to be formatted in such a way that they are not necessarily justified on the right margin. This command has also been redefined from Plain T_EX in that `\rightskip` values are preserved.

`\justify`

This macro has the opposite effect of the `\raggedright` macro and causes the text to be right justified (`\justify` is the default in HP T_EX).

2.6 Boxes

`\boxline=⟨dimension⟩`

This macro assigns the width of the lines used for boxes. The default boxline dimension is 0.01332 inch.

`\boxspace=⟨dimension⟩`

This macro assigns the width of the space between a boxed item and the line of the box. The default width in HP T_EX is 5 points.

`\boxit{⟨horizontal list⟩}`

This macro encloses the argument in a box. If encountered while making a line of text, the box's bottom line will be along the baseline of the text. The ⟨horizontal list⟩ may contain multiple lines separated by `\cr` to be centered within the box. Note that each line is then treated as a **group**, so font changes, *etc.*, on one line will not affect the next line. When using this command, the ⟨horizontal list⟩ will be raised somewhat

like this.

above the line, like this. If you desire that the text not be raised above the baseline, use the `\textbox` command (described below).

`\textbox{⟨horizontal list⟩}`

This command causes a box to be placed around the ⟨horizontal list⟩, **without** altering the text, like this. A `\textbox` cannot be broken from one line of text to another.

`\centerbox{⟨horizontal list⟩}`

This macro centers a box horizontally on the page and inserts space above and below. Multiple lines can be specified using `\cr` (see `\boxit` description). A `\centerbox` within a `\centerbox` will not work, but `\boxit` inside `\centerbox` will.

2.7 Notes and Warnings

Two other text structures available with the HP T_EX macro package are notes and warnings.

`\start{note}`

`\finish{note}`

A note is inset twice the `\inset` dimension on both margins and set apart from the rest of the text by extra vertical space. If a note would otherwise start less than half an inch from the bottom of a page, a page eject is performed prior to the note. `\finish{note}` signifies the end of the note.

```
\start{warning}
\finish{warning}
```

A warning is similar to a note, except that is also set apart by horizontal rules above and below the text. `\finish{warning}` signifies the end of the warning text.

2.8 Verbatim Mode

Verbatim mode will cause text to be printed “as is,” without any justification. Special characters in this mode are the backslash (`\`) and the curly brackets (`{}`, `}`). Most other characters can be used and will be printed verbatim.

Verbatim mode is intended for use with simple text which does **not** contain a large number of control sequences. The reason for this restriction is that there are many characters (such as a space, a tilde, *etc.*) which take on different meanings in verbatim mode. Some of these may be imbedded within a control sequence and can cause problems when they are expanded.

You will usually want to select a “non-proportionally spaced” font (like “`tt`”) for use in verbatim mode. The reason for this is that “proportional fonts” (like “`rm`”) will cause the printed output to be aligned differently than what appears on the CRT during source file typing. Other non-proportional fonts available with the HP 2688A Laser Printer are Courier, Gothic, Pica, Script, Prestige and Line Printer.

– NOTE –

Although verbatim mode using non-proportional fonts will usually produce output that exactly matches what you see on the CRT, there is at least one exception to this. Long sequences of characters may be spaced slightly differently than a string of blank spaces of identical length. This is a result of the rounding anomalies that occur when \TeX 's ideal character sizes are converted to the printer's actual sizes.

```
\start{verbatim}
```

This macro causes the following text to be printed “verbatim” without any justification.

```
\finish{verbatim}
```

This macro ends the verbatim mode described above.

2.9 Paragraph Levels

These macros can be used to create paragraph headings of four different levels. The `(horizontal list)`, and page number can be written to the file `(jobname)*` to be used for a table of contents (the asterisk “`*`” signifies the contents file).

The command `\ctswrite(horizontal list)` will automatically open a file and write the `(horizontal list)` and the page number to the file. The `\contents` command then uses the `(jobname)*` file to create the table of contents (refer to the `\contents` command description in this section).

```
\level(integer between 1 and 4){(horizontal list)}
```

This is the command used to print a paragraph heading. Whatever format is specified using the commands listed below (`\firstlevelhead`, `\secondlevelhead`, *etc.*) will be used. For example, the command used to set the paragraph heading of this section looked like this:

```
\level3{Paragraph Levels}
```

The `\level` command will automatically update the `\levelno` to print the paragraph head.

```
\firstlevelhead{(horizontal list)}
\secondlevelhead{(horizontal list)}
\thirdlevelhead{(horizontal list)}
\fourthlevelhead{(horizontal list)}
```

These commands specify the tokens that are inserted to format each of the various head levels. They should be used when the default conditions of HP T_EX, as described below, are not satisfactory. A control sequence `\title` is defined to be the `(horizontal list)` from the corresponding `\level` command. For example, the command:

```
\secondlevelhead{\need.75in\bigskip\leftline{\fourteenbf%
\levelno\enspace\title}\medskip%
\ctswrite{\hskip15pt\tenrm\title}}
```

would cause all second level paragraph heads to be printed on a new page if less than .75 inch remains on the current page. The title will be left justified in fourteen point boldface type. The level number is followed by an “enspace.” “Bigskip” glue is inserted before the title with a “medskip” after. The `\ctswrite` command specifies how the level heading will be written in the table of contents, as described below.

Whenever you define a `\firstlevelhead`, `\secondlevelhead`, *etc.*, command to specify the formatting of paragraph headings, you must also specify how it will be written in the table of contents (if you plan to create one). The command for doing this is `\ctswrite`. For example, the fourth level head was redefined from the HP T_EX defaults in this document, and the contents writing command was used as follows:

```
\ctswrite{\hskip45pt\ninerm\title}
```

This causes all fourth level headings in the table of contents to be inset 45 points and be written in nine-point roman type.

The default HP T_EX paragraph level headings are formatted as follows: `\level1` begins on a new page, prints the title in 14 point bold face type and down 1.5 inches from the top of the page. `\level2` requires that at least .75 inch remain on the page. If so, the title is printed with some vertical space above it, otherwise, it is set on the top of the next page. In any case, it is printed in 12 point boldface type. `\level3` and `\level4` are similar, but require less space to remain on the page and print the title in 10 point bold face and 10 point roman respectively.

The “levelhead” macros are **local** to their block structure. This provides a simple means to return to the HP T_EX default levelhead style. All you would need to do is enclose the command and any text you want affected within braces. After the closing brace, T_EX will return to the default format. Note that this should only be attempted for shorter blocks. Creating large blocks of several pages in length can cause T_EX to run out of memory.

– NOTE –

If you need to change the font style or size within a level heading, you must specify the entire font name, (for example: `\tentt...` not `\tt`).

`\levelno`

This macro can be used whenever the author desires the number of the heading to appear in the document. The number of the various levels of headings appear, separated by periods. For example, typing `Section \levelno` might result in: Section 1.2.7.1 if this was the first fourth-level heading of the seventh third-level section of the second... *etc.* Levels lower than the last modified level do not appear in `\levelno`.

`\setlevelno{⟨integer⟩.⟨integer⟩..⟨integer⟩}`

This macro can be used for presetting the one through four heading numbers. If you use this command before a `\level` command, then the counter will increment to the next number when the `\level` command is encountered. Therefore, you should preset to one number below the desired level. You cannot preset the level 2 counter without specifying the level 1, or the level 3 without level 1 and 2 and so on. (For example, you cannot specify `\setlevelno{1. . .4}`, you must specify `{1.⟨integer⟩.⟨integer⟩.4}`.)

`\contents`

This macro forces a page eject and produces a table of contents on the following pages as dictated by the previous heading macros. No vertical glue is inserted, so you may want to use a `\vfil` command immediately before `\contents`.

`\topofcontents{⟨vertical list⟩}`

`\botofcontents{⟨vertical list⟩}`

These macros specify a list to be placed above and below the table of contents. The `⟨vertical list⟩` may include logos, rules, *etc.* The table of contents is quite rigid, so a `\vfill` is appropriate in at least one `⟨vertical list⟩`. Defaults in HP \TeX will format the table of contents as they appear in the beginning of this manual (with the exception of the headings).

- NOTE -

`\topofcontents` and `\botofcontents` must both be specified before the `\contents` command is used. (That is, you cannot specify `\botofcontents` after `\contents`, even though this seems logical.)

2.10 Multiple Columns

The following macros control the number of columns on the page. Multiple column and single column text can be mixed on a page by using the following commands.

`\columnspace=⟨dimension⟩`

This command specifies the amount of space between columns on a page. This dimension should be assigned before entering multicolumn mode and should remain unchanged for the duration thereof. The default column space in HP \TeX is 0.5 inch.

`\start{twocolumns}`

`\start{threecolumns}`

`\finish{twocolumns}`

`\finish{threecolumns}`

These four commands cause the text to be formatted into multiple columns. If a `\balance`, `\newpage`, `\evenpage` or `\oddpaper` macro is encountered while in multicolumn mode, the columns are balanced on the page prior to the page eject. Exit from multicolumn mode causes the columns to be balanced as well. Unbalanced columns may be obtained by using `\vfill \eject` while in multicolumn mode. Balancing forces the top lines of each column to be lined up. The bottom lines are lined up as well unless `\raggedbottom` has been specified. Discardable items (such as glue, penalties, *etc.*) immediately following these commands will be ignored. (To prevent this, you may use `\null` immediately after `\finish{ncolumns}`.)

`\balance`

This command causes the columns to become balanced. Discardable items (such as glue, penalties, *etc.*) immediately following these commands will be ignored. (To prevent this, you may use `\null` immediately after `\finish{ncolumns}`.)

- NOTE -

When using multiple columns, you may find it helpful to use the plain T_EX commands `\tolerance` and `\hbadness` (to increase the stretch and shrinkability of interword glue and decrease complaints regarding “underfull hboxes”). See *The T_EXbook* for more information.

2.11 Tables

The following macros are useful for setting tables. These macros are different from the plain T_EX `\halign` and `\valign` commands in that they format one row at a time. The table can be justified as a whole either left, right or centered. As long as the number of columns in each entry remains constant, the entire table appears justified. (Interesting figures, such as pyramids, hopscotch boards, *etc.*, can be produced by changing the format and the number of columns in each entry.) Each column of the table has its own user specified width. Between each pair of columns is a vertical line (which can be easily made invisible). This vertical rule takes no space from any column so the rule width may be varied without altering the column dimensions. An entry may also be designed to span two or three columns. Horizontal bars are treated exactly the same as data entries. The macro `\tbar` can be used to produce such a bar.

`\tableline=<dimension>`

This parameter specifies the dimension of bars and rules in all subsequent entries. This parameter should remain unchanged throughout the entire table. Default value is 0.01332 inch.

- NOTE -

There may be some occasion when printing a table that you get the error message,

“Printer error:

**Could not process all the data, data lost.”

The printer uses a variety of different length line segments to “build” a line. If the `tableline` specification you have chosen requires too many individual segments to make up the exact specification, it may cause this error. There are a few remedies available, as follows:

—If you are using the default magnification (1.0), then try using the default `tableline` dimension (just leave out the `\tableline` command).

—If you are using a file magnification value other than 1.0, then alter your `\tableline` specification so that its value, when converted to dots (multiply length in inches \times 300), is a power of 2 (*i.e.* 1, 2, 4, 8, 16, 32 or a multiple of any of these except 1).

`\tablespace=<dimension>`

This parameter specifies the minimum amount of space to be placed between the vertical rules and data entries in the table. Default in HP T_EX is 5 points.

`\tableformat{<table spec.>|<column spec.><dimension>|...|<column spec.><dimension>}`

This command specifies the table format. Note that a vertical bar (|) is required as a separator between specifications, but not at the beginning or end of the list (different than `\tablerow` or `\tablebar`).

`<table spec.>` can be any of `\leftline`, `\rightline`, or `\centerline`. `<column spec.>` can be any of `\leftline`, `\rightline`, `\centerline` or `\paragraph`. Other options are allowed if you make them yourself; for example, if you type:

```
\def\mything#1{\line{\hskip 1in #1\hss}}
```

then `\mything` would be a valid `(table spec.)` option that would print the table 1 inch from the left margin. The `(column spec.)` is always a single token (`\leftline`, `\rightline`, `\centerline` or `\paragraph`), and is followed by a dimension. The token specifies the standard justification of the column (`\paragraph` must be used if the column is to contain paragraphs), and the dimension specifies the absolute width of the column. Again, if you are not satisfied with the selection you can make your own—the rules are that the macro must consume one token (containing the text) and produce a box of width `\hsize`.

– NOTE –

If the `\paragraph` column specification is used, `\parskip` must be 0 points (this is the default). Otherwise, the paragraph entry will be raised or lowered by the current `\parskip` amount and the table will be out of alignment. If you are using `\noindentstyle` and have not reset `\parskip`, then inserting an `\indentstyle` command just prior to the table will return to the proper (0 points) `\parskip`.

```
\tablerow{|(column list)|(column list)...|(column list)|}
```

```
\tablebar{|(column list)|(column list)...|(column list)|}
```

These commands are used to build tables. `\tablerow` should be used if the `(column list)` contains text, while `\tablebar` should be used for bars or other non-text entries. Horizontal bars are obtained by using either `\vrule height (dimension) width (dimension)` or `\tbar` as a `(column list)` (`\tbar` uses `\tableline` as the rule height). The following syntax holds true for both macros:

- A vertical bar (`|`) or a tilde (`~`) must be present to indicate the beginning and end of each column, including the first and last entry (differs from `\tableformat` syntax).
- The vertical bars (`|`) may be replaced by tildes (`~`) if a visible vertical bar in that position is not desired.
- If the first thing in a `(column list)` happens to be the control sequence `\span{(integer)}\tbar`, then the number of columns specified as `(integer)` are spanned by the entry (in this case a horizontal bar). The natural justification for the spanned entry will be that of the leftmost column spanned. (Note that `\span{(integer)}` is **not** followed by a blank space as this can cause difficulties.)

– NOTE –

While building tables, if a `\tablebar` or `\tablerow` runs more than one line of text (on the CRT), it is good practice to use the percent sign (`%`) at the end of each line. This will tell `TEX` to ignore anything else on that line and prevent possible problems with extra spaces being misinterpreted by `TEX`.

2.12 Fonts

Only a few fonts are preloaded in HP `TEX`. These macros allow you to access a variety of font families, styles, and sizes.

`\fontdef⟨command⟩={⟨library⟩,⟨font name⟩}`

This command equates a command of your choice with a font (as used in the library). You can use `⟨library⟩(optional)` to specify other than the default font library. This command differs from T_EX's `\font` command in that the font won't actually be loaded until the first request (if any).

`\fivepoint`
`\sixpoint`
`\sevenpoint`
`\eightpoint`
`\ninepoint`
`\tenpoint`
`\twelvepoint`
`\fourteenpoint`
`\eighteenpoint`
`\twentyfourpoint`

These commands select the font point size from five to twentyfour. (In **Math Mode**, these commands only apply to the default font within a font family. If you are using a current family other than `\fam0`, you must make the appropriate font assignments (see Appendix A, "Changing Fonts in HP T_EX").

`\rm`
`\it`
`\bf`
`\sl`
`\sa`
`\tt`

These commands select roman style, italic, bold face, slanted, sans serif, and typewriter style.

If the selected font is undefined, the font style is changed to roman. If that new font is still undefined, the size is changed to ten points. Computer Modern ten point roman will always be defined in HP T_EX.

2.13 Miscellaneous Macros

`\note{⟨horizontal list⟩}`

This macro puts a footnote at the bottom of the current page and inserts a superscript footnote number at the location of the command and the footnote. Footnote numbers are allocated starting with 1 and can be reset by the use of `\resetnotes`. The `⟨horizontal list⟩` is the footnote text.

`\need⟨dimension⟩`

The result of this macro is that if the page breaking algorithm of T_EX determines that the current position would optimally fall within `⟨dimension⟩` of the bottom of the page, the page is broken leaving some empty space at the bottom.

- NOTE -

Since the `\need` command uses a negative penalty to **encourage** (not **force**) T_EX to break the page, it will not always have the effect you might anticipate. This is especially true if the command is encountered near the top of the current page and the remaining text will not adequately fill the current page.

`\super{⟨horizontal list⟩}`

The argument is printed as a superscript.

`\sub{⟨horizontal list⟩}`

The argument is printed as a subscript.

`\lbreak`

This macro forces a line break within a paragraph. It inserts “\fil” glue before the break, so the line will not be right justified. If you want right justification, use plain T_EX’s `\break` command.

`\uline{⟨horizontal list⟩}`

This macro causes the argument to be underlined.

`\mon`

This command prints the current month name. For example, if the current month were August, `\mon` would print the letters “August” in the current font.

`\date`

This command prints the current date in the format Month Date, Year. For example, if today’s date were 12 December 1984, `\date` would print the characters “December 12, 1984” in the current font.

`\hour`

This command prints the current time of day (for example – 4:07 PM)

2.14 Block Structure

Certain local document styles are considered to belong to a “block.” (Block structure, as used here, simply refers to a portion of a file that has some common formatting instructions applied to it.)²

The current block is defined to be the most recently opened block. Pending blocks are blocks that have been opened but not closed. The HP T_EX macros `\start` and `\finish` are used to open and close blocks.

Valid ⟨blockname⟩s pre-loaded in HP T_EX include `indent`, `note`, `warning`, `verbatim`, `twocolumns` and `threecolumns`, all of which have been described in this chapter. User defined blocks may be implemented by defining a control sequence, `\BEGIN⟨blockname⟩` and, optionally, another control sequence, `\END⟨blockname⟩`.

`\start{⟨blockname⟩}`

This command determines if a control sequence `\BEGIN⟨blockname⟩` has been defined. If one exists, a new block is opened and the control sequence is invoked. Otherwise, an error message is issued and the command is ignored.

`\finish{⟨blockname⟩}`

This command is used to close the current block and invoke the control sequence `\END⟨blockname⟩` if one has been defined. If ⟨blockname⟩ matches the current block name, the current block is closed. If not, an appropriate error message is issued and corresponding corrective action is taken. If ⟨blockname⟩ is not valid or there are no pending blocks, the command is ignored. If ⟨blockname⟩ is valid but does not match the current block name, `\done` commands are inserted until either ⟨blockname⟩ matches the current block name or until all pending blocks are closed.

² The concept of **grouping**, explained in this manual and in *The T_EXbook*, is basically the same as the “block structure” referred to here.

\done

This command closes the current block without any error checking and invokes the control sequence `\END(blockname)` if one has been defined. A T_EX error message will be issued if there are no pending blocks.

Following is an sample usage of the `\BEGIN` and `\END` commands in creating user-defined blocks:

- EXAMPLE -

This example was created by defining the following block:

```
\def\BEGINexample{\bigskip\centerline{- EXAMPLE -}
\medskip\leftskip.75in\rightskip.75in}
```

The block was opened using the command: `\start{example}`. The optional control sequence `\END(blockname)` was also used, as follows:

```
\def\ENDexample{\medskip}
```

The `\medskip` vertical glue will be inserted after the block is closed using the command: `\finish{example}`.

3 HP 2688A Control Macros

These macros are for controlling certain features of the 2688A Laser Printer:

3.1 Page Copy Control

\copies(integer)

This macro assigns the number of copies per page starting with the current page. The copies are uncollated; in other words, if `\copies5` occurred on page 3, the output will have five copies of page 3, then five copies of page 4, *etc.* The control of the number of copies is accomplished through one of T_EX's counters. Default is counter number 1. Possible integers and their effect are shown below:

- 1 to 32,767 will produce the specified number of copies.
- Greater than 32,767 will produce 32,767 copies.
- 0 will produce 1 copy.
- Less than 0 (negative number) no copies (can be used to suppress certain sections of a document, *etc.*)

The counter can be changed using the following macro.

\selectcopycounter(integer from 0 to 9)

This is used for changing which counter is used for controlling the number of copies per page (default is 1). This command generally should not be used except by people who are writing their own macros.

`\copieson`
`\copiesoff`

If the multiple copies feature is being used in a document, these macros can be used to alternate between one copy per page and the number of copies assigned by the `\copies` macro. The difference between `\copiesoff` and `\copies1` is that when the former is followed with `\copieson`, the original number of copies is restored, whereas when the latter is followed with `\copieson`, there is, of course, no effect.

3.2 Logical Page Control

The Print Server features a concept called “logical paging” which is controlled using the following set of macros. A logical page is a rectangular addressing space on the sheet of paper (referred to as the “physical page”). Logical pages have an orientation (portrait, landscape, reverse portrait or reverse lanscape) and a position on the physical page. Through the operator interface of PS2688A, you may define a list of logical pages that may be written to on any sheet of paper. These logical page specifications may also be stored in a PS2688A input file to avoid typing in the specifications for each job. Using these macros, you can address the logical pages with either of two methods: either by explicitly specifying a logical page for each page of \TeX output, or by specifying an ordered list of logical pages and letting the system cycle through the list.

`\lpdef{(logical page definition)}`

This command defines a logical page size and orientation. The `(logical page definition)` consists of the logical page number, followed by the `left`, then the `top` dimensions (distance from the edges of the physical page), and the logical page orientation (portrait, landscape, rev-portrait or rev-landscape). Orientation may be abbreviated P, L, RP, and RL, respectively, and either upper or lower case characters will work. The edges of the page that would normally be the top and left as viewed from that logical page’s orientation are always considered top and left in this context. An example usage follows:

```
\lpdef {1, 1.5in, 3in, L}
```

This would cause logical page number 1 to be printed in **landscape** orientation, with the left edge of the print (as viewed from landscape orientation) beginning 1.5 inches from the edge of the physical page. The “top” (left-hand edge along the paper path) will begin 3 inches from the edge.

The default values for logical pages in HP \TeX are: LEFT=1in, TOP=1in.

- NOTE -

The dimension specified as distance from the top edge does not take into account the space needed for headings. If running headings are to be used, the logical page definition should allow extra room for them when specifying “top” dimension. (Footings are also printed “in the margin,” so when specifying `vsize` be sure to leave room for them.)

`\lpelist{(logical page list)}`

This command specifies the logical page list to be used whenever logical pages are being used implicitly. Only one such specification is allowed within a document; if more than one exist, all but the last are ignored. This command initializes implied logical paging.

The `(logical page list)` is a list of integer numbers from 1 to 32 which have been defined using the `\lpdef` command. Positive number entries will cause a physical page eject prior to initiating the logical page entry. Negative numbers require no such physical page eject. Each entry number should be separated from the next by a comma.

The remaining commands control explicit logical paging. If any of these commands occur within a page, a page eject is issued and the next page to be printed will be dependent upon the particular command.

If any of these commands occur on the top of a page, the prescribed action will be delayed until the next page break (be it natural or forced by a `\newpage`, `\eject`, etc.). For example, suppose the first page of every chapter in a document uses logical page 5, then `\ppageto5 \lpresume` would eject the current page (assuming it is not empty), issue a physical page eject and begin printing on page 5. Later, when the page is full, TeX breaks the page and resumes printing according to the logical page list.

`\lpageto`(integer between 1 and 32)

This command causes the next page to be printed on the specified logical page. No physical page eject is implied.

`\ppageto`(integer between 1 and 32)

This command is similar to `\lpageto` but implies a physical page eject prior to printing on the specified logical page.

`\lpresume`

This command is used after a `\ppageto` or `\lpageto` command to resume printing on the current page of the logical page list.

`\lppreset`

This command resets the logical page list and prints on the first page in the list.

`\lpexit`

This command exits the current loop in the logical page list and prints on the next page in the list.

`\selectlpcounter`(integer)

This is used for changing which counter is used for specifying the logical page (default is 2). This macro generally should not be used except by people who are writing their own macros.

4 Formatting a Sample Document

In this section we will format a simple document utilizing many of the HP \TeX macros. For this example, we will accept all the HP \TeX default values for the various commands.

A Sample Document

Isn't that nice? If we wanted to, we could have easily set that off to the right like this:

A Sample Document

Now, how about a textbox? Those are always fun. Or, if we really want to set something off, we could ask \TeX to:

put it inside
a nice little
centered box,

or, maybe just

“boxit” – like this.

But say you're really serious about getting someone's attention.

LOOK!

You can use the “Warning” command, like we did here.

Now, let's use the extraordinary capabilities of \TeX to generate a mathematical formula:

$$\hat{n}_2(s) = \frac{1}{\alpha_2} \left[\left(\frac{\partial C_2}{\partial x} \right)_{x=0} + \frac{k_1 \hat{n}_1}{D_1} \right]$$

Next, we'll create a table to show the dimensional units available in \TeX :

\TeX UNIT	DESCRIPTION	PER INCH	\TeX UNIT	DESCRIPTION	PER INCH
in	inch	1	mm	millimeter	25.4
cm	centimeter	2.54	dd	Didot point	67.54
pt	printer's point	72.27	bp	big point	72
pc	pica	6.02			

Itemized lists are very useful:

1. For listing things.
2. For making a series of points.
 - a. You can even use subitems.

Now, look at the next page to see exactly how this sample document was formatted.

Here's the source file for "Sample Document"

```
\noindentstyle
\centerfooting{\eightbf THIS DOCUMENT WAS CREATED USING \HPTEX}
\null\bigskip
\centerline{\twelvebf A Sample Document}
```

Isn't that nice? If we wanted to, we could have easily set that off to the rightlike this:\lbreak

```
\rightline{\twelvebf A Sample Document}
```

Now, how about \textbox{a textbox?} Those are always fun. Or, if we really want to set something off, we could ask \TeX\ to:

```
\centerbox{put it inside\cr
           a nice little\cr
           centered box,}
```

or, maybe just \boxit{'boxit' -- like this.}

But say you're really serious about getting someone's attention.

```
\start{warning}
\centerline{\bf LOOK!}
\centerline{You can use the "Warning" command, like we did here.}
\finish{warning}
```

Now, let's use the extraordinary capabilities of \TeX\ to generate a mathematical formula:

```
$$\hat{n}_2(s)=\frac{1}{\alpha_2} \operatorname{biggl[} \operatorname{biggl[}\frac{\partial C_2}{\partial x} \operatorname{biggr]}_{x=0} +\frac{k_1}{D_1} \hat{n}_1 \operatorname{biggr]}$$
```

Next, we'll create a table to show the dimensional units available in \TeX :

```
\medskip
\tableformat{\centerline|\centerline 1in|\leftline 1.2in|%
             \centerline 1in|\centerline .05in|%
             \centerline 1in|\leftline 1.2in|\centerline 1in}
\tablebar{|\span7\tbar|}
\tablerow{\TeX\ UNIT|DESCRIPTION|PER INCH| |\TeX\ UNIT|DESCRIPTION|PER INCH|}
\tablebar{|\span7\tbar|}
\tablerow{\tentt in|inch|1| |\tentt mm|millimeter|25.4|}
\tablerow{\tentt cm|centimeter|2.54| |\tentt dd|Didot point|67.54|}
\tablerow{\tentt pt|printer's point|72.27| |\tenttbp|big point|72|}
\tablerow{\tentt pc|pica|6.02| | | |}
\tablebar{|\span7\tbar|}
```

```
\medskip
```

Itemized lists are very useful:

```
\itm For listing things.
\itm For making a series of points.
\sitm You can even use subitems.
\enditems
```

Now, look at the next page to see exactly how this sample document was formatted.

```
\vfil\ejct
```

* * * * *
 Letters et alia
 * * * * *

To the Editor:

Several recent reviews in *Computing Reviews* have confirmed my long-standing impression that lots of readers are sensitive to matters of quality printing. So far TeX has gotten good press in that journal; for example, Dick Andree began review #39,590 by saying "You will certainly be fascinated to see the excellent mathematical typesetting displayed in this book, set using Don Knuth's TeX system. It is worth examining for this alone."

Of course, such reviews are somewhat embarrassing to me, because the lion's share of the praise should obviously go to the authors for the wonderful things they wrote; the format is only one of the many things that were well done.

This particular book—*Practical Optimization* by Gill, Murray, and Wright—was one of the first to be produced on Stanford's Alphatype. I can recall being pleasantly surprised to discover isolated pages (of unknown authorship) in our darkroom while I was working on *Seminumerical Algorithms*; later I found out that Gill, Murray, and Wright were responsible for this fine work. They took the time to "go the extra mile" by combining superb mathematical exposition with numerous refinements. For example, they added a unique appendix about "questions and answers"; they included excellent illustrations and tables; they chose their notation carefully; they made a good index and bibliography; they found wonderful quotations for the beginnings of each chapter. If TeX had any part in this, it was merely to encourage the authors to strive for such quality because they were more personally involved.

Lynn Steen's "telegraphic review" of Arthur Keller's *First Course in Computer Programming Using Pascal* is another instance of format appreciation; he says, "The elegance of the text is matched by the elegance of its appearance: it was prepared and typeset at Stanford in TeX." [*American Math Monthly*, January 1983, page C8.]

Conversely, Bob Fenichel's review #40,719 in *Computing Reviews* speaks of a book that "is photographically reproduced from the output of the author's daisy-wheel printer The article headers are set randomly in at least two different fonts, while the font of the text, for reasons that the author does not share, is the inhuman OCR-B."

Such judgments are obviously matters of taste, and we can't expect universal agreement. For example, I went to the bookstore to see the book just

mentioned, and I didn't find its format disturbing; indeed, I think Adrian Frutiger did an excellent job, under the circumstances, when he designed OCR-B.

Fenichel's review goes on to make a point that I think is much more important: "There are errors in spelling and grammar on nearly every page. Granted that some authors have special interest and competence in copy editing, proofreading, and typographical design, are all authors now to be required to have such interest and competence? If not, then how can a publisher, presented with so-called camera-ready copy, reassert his traditional control of these matters? Should he refuse to accept such copy, accepting only traditional manuscript or machine-readable text?"

We must realize that fine formatting is only one of many aspects of book quality. I hope the day will come when there are copy editors and book designers familiar with TeX, to deal with authors who are typesetting their books with TeX. Meanwhile, I want to encourage authors who are using TeX today to seek professional help, instead of assuming that publishers do nothing but print, bind, and distribute books. It would be terrible if TeX were to lead to *decreased* quality because these other aspects were being neglected.

Of course, there is no royal road to quality; editors can make mistakes too. The immediately following review [*Computing Reviews* #40,720] mentions that another book "is particularly ill-served by its editors, who have failed to correct a large number of errors in English usage."

The best way to solve all of these problems is with teamwork. I hope that TeX will ultimately help to provide better means of communication between people with different kinds of book expertise. At the moment we are seeing a sudden shift in who has the ultimate power to input and change copy; with care, we should be able to find an appropriate way to distribute that power.

—Don Knuth

* * * * *

Advertisements

* * * * *

Classified

Book production in TeX, designed to meet your publisher's specifications, from manuscript or formatting of on-line material; also consulting and custom macro production. Amy Hendrickson, TeX-ing Service, 57 Longwood Ave. #8, Brookline, MA 02146. (617) 738-8029. Net Address: Amy@MIT-MC.

TEXTSET, Inc. P.O. Box 7993, Ann Arbor, Michigan 48107
direct-typesetting services and software

TEXTSET offers a full range of services and products to T_EX users.

DIRECT-TYPESETTING

Direct-typesetting from T_EX DVI files, T_EX source, and other formatting systems on an Autologic APS-5. Macro consultation and design. Wide range of fonts available including Computer Modern.

T_EX SYSTEM CONSULTING

Expert consulting and programming services for in-house T_EX users. Design and implementation of complete working T_EX systems.

- preprocessors
- T_EX installation
- macro package design
- device drivers
- font assistance
- system integration
- special applications support
- programmer level instruction

T_EX SOFTWARE

T_EX installation program for Sun Workstations.

T_EX screen preview program for Sun Workstations.

T_EX DVI-to-printer drivers for Autologic APS-5 series phototypesetters, Linotron 202, Xerox 9700, and other selected printing devices.

Multiple copy discounts and maintenance agreements available.

For more information call Bruce Baker at (313) 996-3566.

Typeset using T_EX

TEX82 ORDER FORM

The latest official versions of TEX software and documents are available from Maria Code by special arrangement with the Computer Science Department of Stanford University.

Nine different tapes are available. The generic distribution tape contains the source of TEX82 and WEB, the test program, a few "change" files, the collection of fonts in TFM format, and other miscellaneous materials; a PASCAL compiler will be required to install programs from a generic tape. The AMS-TEX macro package is included on the TEX distribution tapes; other macro packages, including L²TEX and HP TEX, will be added as they become available. The special distribution tapes are for the indicated systems only, and should be ordered for these systems instead of a generic tape. Two tapes are PXL font collections covering various magnifications at 200/240 dots/inch and 300 dots/inch respectively. The METAFONT tape contains the SAIL source for the METAFONT program and includes the .MF source files.

Each tape will be a separate 1200 foot reel which you may send in advance or purchase (for the tape media) at \$10.00 each. Should you send a tape, you will receive back a different tape. Tapes may be ordered in ASCII or EBCDIC characters. You may request densities of 6250, 1600 or 800 (800 is discouraged since it is more trouble to make).

The tape price of \$82.00 for the first tape and \$62.00 for each additional tape (ordered at the same time) covers the cost of duplication, order processing, domestic postage and some of the costs at Stanford University. Extra postage is required for first class or export.

Manuals are available at the approximate cost of duplication and mailing. Prices for manuals are subject to change as revisions and additions are made. It is assumed that one set of manuals will suffice you. If you require more than two sets, please write for prices since we must ask for more money for postage and handling.

Please send a check or money order (payable on a US bank) along with your order if possible. Your purchase order will be accepted, as long as you are able to make payment within 30 days of shipment. Please check this out before sending a purchase order since many large firms seem to be unable to make prompt payment (or don't worry about it).

The order form contains a place to record the name and address of the person who will actually use the TEX tapes. This should *not* be someone in the purchasing department.

Your order will be filled with the most recent versions of software and manuals available from Stanford at the time your order is received. If you are waiting for some future release, please indicate this. Orders are normally filled within a few days. There may be periods (like short vacations) when it will take longer. You will be notified of any serious delays. If you want to inquire about your order you may call Maria Code at (408) 735-8006 between 9:30 a.m. and 2:30 p.m. West Coast time.

If you have questions regarding the implementation of TEX or the like, you must take these to Stanford University or some other friendly TEX user.

Now, please complete the order form on the reverse side.

T_EX82 ORDER FORM

**** TAPES **** density (6250, 1600 or 800) = _____

T_EX generic distribution tapes (PASCAL compiler required):

_____ ASCII format _____ EBCDIC format

T_EX distribution tapes in special formats:

_____ VAX/VMS Backup format _____ IBM VM/CMS format

_____ DEC 20/Tops-20 Dumper format _____ * IBM MVS format

* Not yet available; call before ordering

Font tapes:

_____ Font library (200/240 dots/inch) _____ Font library (300 dots/inch)

_____ METAFONT (SAIL compiler required)

_____ Total number of tapes.

Tape costs: \$82.00 for first tape; \$62.00 for each additional.

Tape cost = \$ _____

Media costs: \$10.00 for each tape required.

Media cost = \$ _____

**** MANUALS ****

_____ T_EX82 - \$20.00 _____ Test Manual - \$8.00

_____ WEB - \$10.00 _____ T_EXware - \$8.00

_____ T_EXbook - \$20.00 _____ I³T_EX (preliminary edition) - \$8.00

Manuals cost = \$ _____

California orders only: add sales tax = \$ _____

Domestic book rate: no charge.

Domestic first class: \$2.50 for each tape and each manual.

Export surface mail: \$2.50 for each tape and each manual.

Export air mail to North America: \$4.00 each.

Export air mail to Europe: \$7.00 each.

Export air mail to other areas: \$10.00 each.

Postage cost = \$ _____

(make checks payable to Maria Code)

Total order = \$ _____

Name and address for shipment:

Person to contact (if different):

Telephone _____

Send to: Maria Code, DP Services, 1371 Sydney Dr., Sunnyvale, CA 94087

Request for Information

The T_EX Users Group publishes a membership list containing information about the types of equipment on which members' organizations plan to or have installed T_EX, and about the applications for which T_EX would be used. It is important that this information be complete and up-to-date.

Please answer the questions below, and also those on the other side of this form, obtaining information from the most knowledgeable person at your installation if necessary. Some sites have more than one computer system on which T_EX has been or might be installed. Please list all such machines below. Output device information should be given on the other side. If you need more space than is provided here, feel free to use additional paper.

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:
T_EX Users Group
c/o American Mathematical Society
P.O. Box 1571, Annex Station
Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers*
the name and address of the AMS bank is:
Rhode Island Hospital Trust National Bank
One Hospital Trust Plaza
Providence, Rhode Island 02903, U.S.A.
- *General correspondence*
about TUG should be addressed to:
T_EX Users Group
c/o American Mathematical Society
P.O. Box 6248
Providence, Rhode Island 02940, U.S.A.

Name: _____
 Home [] Address: _____
 Bus. [] _____

DELIVERY:
 The T_EXbook and The Joy of T_EX are normally sent via UPS within the U.S., and as printed matter outside the U.S. For **FASTER DELIVERY:**
 First Class in U.S., add \$2.00 per book;
 Air Mail outside U.S., add \$3.00 per book.

QTY	ITEM	AMOUNT
	1984 TUGboat Subscription/TUG Membership (Jan.–Dec.) – North America New (first-time): [] \$20.00 each Renewal: [] \$30.00 each	
	1984 TUGboat Subscription/TUG Membership (Jan.–Dec.) – Outside North America * New (first-time): [] \$25.00 each Renewal: [] \$35.00 each	
	TUGboat Back Issues, \$15.00 1980(v.1) 1981(v.2) 1982(v.3) 1983(v.4) per issue, circle issue(s) desired: #1 #1, #2, #3 #1, #2 #1, #2	
	<i>The Joy of T_EX</i> (rev. prelim. ed., 1982, with AMS-T _E X82 suppl.) @ \$10.00 each Code: JOYTT	
	<i>The T_EXbook</i> by Donald E. Knuth, 1984 @ \$15.00 each Code: TEXBKT	
	<i>First Grade T_EX: A Beginner's T_EX Manual</i> by Arthur L. Samuel @ \$10.00 each	
	<i>User's Guide to the HP T_EX Macros</i> by Susan Daniels @ \$6.00 each	
	T _E X and Metafont: Errata and Changes (final edition, September 1983) – \$4.00 each	
	The T _E Xbook: Errata and Changes (included with TUGboat) – additional copies \$3.00 each	
	T _E X Lectures on Tape (see cover 3, Vol. 5, No. 1)	

* Air mail postage is included in the rates for all memberships and subscriptions outside North America.

TOTAL ENCLOSED: _____
 (Prepayment in U.S. dollars required)

* * *

Membership List Information

Institution (if not part of address):
 Title:
 Phone:
 Specific applications or reason for interest in T_EX:
 My installation can offer the following software or technical support to TUG:

Date:
 Status of T_EX: [] Being installed
 [] Up and running since
 [] Under consideration
 Version of T_EX: [] SAIL
 Pascal: [] T_EX82 [] T_EX80
 [] Other (describe)

From whom obtained:

Please list high-level T_EX users at your site who would not mind being contacted for information; give name, address, and telephone.

Approximate number of users:
 Computer system(s):

Please answer the following questions regarding output devices used with T_EX unless this form has already been filled out by someone else at your installation.

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: () .tfm () .tfx

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.