Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?

Brian W. Kernighan and P. J. Plauger
The Elements of
Programming Style,
Second edition, McGraw-Hill, 1978.

THE TEX USERS GROUP NEWSLETTER

EDITOR BARBARA BEETON

VOLUME 4, NUMBER 2 • SEPTEMBER 1983
PROVIDENCE • RHODE ISLAND • U.S.A.

ADDRESSES OF OFFICERS, AUTHORS AND OTHERS

AUTOLOGIC, INC.
Frank Campanaro
1050 Rancho Conejo Blvd.
Newbury Park, CA 91320
805-498-9611 x144

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

BIGELOW, Charles
Department of Computer Science
Stanford University
Stanford, CA 94305

CARNES, Lance
163 Linden Lane
Mill Valley, CA 94941
415-388-8853

CHILDS, S. Bart
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

DOHERTY, Barry
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500

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

GOUCHER, Raymond E.
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500

HEWLETT-PACKARD COMPANY
Larry W. Haley
P O Box 15
Boise, ID 83707
208-376-6000

HOBBY, John
Department of Computer Science
Stanford University
Stanford, CA 94305

IMAGEN CORPORATION
Dan Curtis
2660 Marine Way
Mountain View, CA 94043
415-960-0714

KELLER, Arthur
Computer Science Dept
Stanford University
450B Margaret Jacks Hall
Stanford, CA 94305
415-497-3227
ARK@SAIL

KIM, Scott
Department of Computer Science
Stanford University
Stanford, CA 94305

KNUTH, Donald E.
Department of Computer Science
Stanford University
Stanford, CA 94305
DEK@SAIL

LAMPORT, Leslie
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
415-859-3652

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

MOONEY, James D.
Dept. of Statistics and Computer Science
West Virginia University
Morgantown, WV 26506
304-293-3607

NICHOLS, Monte C.
Exploratory Chemistry Division
Sandia National Laboratories 8313
Livermore, CA 94550
415-422-2906

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

PRICE, Lynne A.
CALMA
Research and Development
527 Lakeside Drive
Sunnyvale, CA 94086
408-970-3760

QUALITY MICRO SYSTEMS, INC
Don Parker
57 S Schillinger Rd
Mobile, AL 36608
205-633-4300

RODGERS, David
Textset, Inc
1612 Anderson
Ann Arbor, MI 48104
313-764-7228

ROESSER, James R.
Science Typographers Inc
15 Industrial Boulevard
Medford, NY 11763
516-924-4747

RUGGLES, Lynn
Department of Computer Science
Stanford University
Stanford, CA 94305

SIEGEL, Dave
Department of Computer Science
Stanford University
Stanford, CA 94305
NWD@SU-SAIL

SPIVAK, Michael
2478 Woodridge Drive
Decatur, GA 30033

STERKEN, Jim
2701 Lookout Circle
Ann Arbor, MI 48104

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

THEDFORD, Rilla
Mathematical Reviews
611 Church Street
P O Box 8604
Ann Arbor, MI 48107
313-763-6828

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

WOODSMALL, Roger
Intergraph Corporation
17629 El Camino Real #204
Houston, TX 77058
713-486-5620

ZABALA, Ignacio
Centro de Cálcolo
Facultades de Ciencias
Universidad de Valencia
Cametera de Ademuz
Valencia, Spain
011-34-6-357-4065

ZAPF, Hermann
Seitersweg 35
D-6100 Darmstadt, Fed Rep Germany

TUGboat, the newsletter of the TeX Users Group (TUG), is published irregularly 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, August 1984, Stanford University

A meeting of the TEX Users Group is tentatively planned for the week of August 20–24, 1984, at Stanford University. Joey Tuttle, of I. P. Sharp Associates, Palo Alto, will once again be in charge of compiling the program and of local arrangements. The last two days will be occupied by a TEX-related course. Any suggestions concerning subjects to be covered should be conveyed to either Joey Tuttle, (415) 327-1700, or Ray Goucher, (401) 272-9500, ext. 232. During the previous week, August 13–17, it is planned that Arthur Keller will teach a TEX for Beginners class, comprising morning lectures followed by afternoon labs. Further information on both the meeting and the two courses will be mailed out after the first of the year.

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,
  reduced rate of $20 for renewals received before January 31, 1984.

Outside North America (includes air mail postage):
- New (first-time) members or subscribers: $25.
- Membership and subscription renewals: $35,
  reduced rate of $25 for renewals received before January 31, 1984.

Membership privileges include all issues of TUGboat published during the membership (calendar) year. All new members and other persons inquiring about TUG will be sent a complimentary copy of TUGboat Vol. 1, No. 1 (1980).

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 on the order form for general correspondence.

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, special reduced rates for participation at TUG meetings and TEX-related courses, and being listed as an institutional member 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. 1 (1984), will be April 16, 1984; the mailing date will be May 24. 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 TEX 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

Late in the afternoon of Bastille day, after an hour or so of lively debate on structure and amendments, the TEX Users Group adopted its first formal constitution. We have tried to take all reasonable contingencies into account, but still keep the articles short, clear and not too numerous. It should serve us satisfactorily for a while, at least, and we do not expect that we will have to ask for the creation of a `\documentstyle{constitution}` in LaTEX in the near future.

As you will discover elsewhere in this issue, a new executive committee came into being together with the constitution, and I am addressing you for the first time as president of TUG. I begin in this office with feelings of profound gratitude to the officers of the pre-constitutional period. The new Executive Committee inherits a lively, active and financially solvent organization. It will be our job to live up to their example. Much of our past success as an organization has depended on the services of Ray Goucher, and on the generous assistance given us by the American Mathematical Society in a variety of ways. We are fortunate in being able to continue this relationship, and we have decided that one of our first actions should be to give formal recognition to Ray by asking him to accept the position of Business Manager to TUG.

Our financial position is just about right. We are neither too lean nor too fat. We carry forward a sufficient balance to cover normal annual expenses, and our revenues have so far kept pace very nicely with our outlays. We do, however, recognize that Don Knuth's generosity in offering the TEX short courses in association with the meetings of TUG is the real reason for our solvency.

In this general connection, we have given some thought to streamlining the necessary collection of TUG dues. Mailing out reminders is costly both in time and in postage, and you will notice that our new dues structure gives a substantial discount to those members who respond quickly. The discount comes quite close to representing the savings made by the avoidance of a second and third mailing of reminders.

That gets the principal organizational details out of the way, and now for the good news. By the time you read this, version 0.999 of TEX will be up and running on a variety of systems. Perhaps the version number will be 0.9999, or perhaps the final rounding operation will have brought us up to Version 1.0. It has been a great education to watch the development of a new software system through all the stages from 0.0 to 0.999. On three occasions, major changes have been made which required a certain amount of effort to bring old macro files into conformity with the new syntax. In some cases the adoption of a major change was so obviously beneficial (e.g., the new syntax for font declarations) that it simply appeared in the newest version. In other cases, (e.g., the change in `{if...fi}` syntax) there was a general polling of those already using TEX82. In every case except the one brief episode of the non-standard printer's point the change so obviously justified itself that the argument for it was overwhelming. Those of you who make the acquaintance of TEX82 for the first time with version 1.0 will find that it takes some care to make the transition from TEX80 to TEX82, but you will be glad of the increased power and flexibility of the new system. It does all sorts of things that you simply couldn't do in the old version, and it does even the old things better.

Concurrently with the development of TEX82, there are the several systems for smoothing the path between the end user and the more arcane details of TEX. Fácil TEX and AMS-TEX are familiar to many users already. Max Díaz has continued work on the one, and Michael Spivak and Ron Whitney on the other, so that they will be available for use with TEX82 at more or less the same time as Version 1.0. Leslie Lamport's LaTEX, which he himself pronounces in almost all the possible ways, is a new approach to TEX, inspired by Brian Reid's Scribe. Like Scribe, it allows the user to work with the logical structure of a document, and in most instances to avoid tinkering with the precise details of TEX formatting. (In a very specialized way, the **WEB** system of structured documentation does much the same thing.) Systems such as LaTEX and **WEB** will go a long way towards answering the most common criticism of TEX, that it drew the user too deeply and inexorably into the details of formatting at the primitive level. (The great secret, of course, is that many of us find these details almost irresistible.) There are rumors of a combination of the AMS-TEX macros with LaTEX, which lead one to wonder just how the name will be formatted.

One of the most impòrtant activities for our organization in the next few years will be to educate the unconverted masses. Many of those who, until now, have resisted the seductions of typesetting are ready to take a second look. It would help a great deal if we could easily lay our hands on convincing examples of the benefits of using TeX. It was proposed at this year's meeting that we provide for this information in two ways.

1. We should maintain a bibliography of articles about experiences with TeX and **METAFONT**. This will be particularly helpful for those articles which have appeared in unexpected journals. If you know of any, please send the reference to Barbara Beeton at the American Mathematical Society.
2. We should also keep a list of books published using TeX, and for this we need a really full description. Which TeX was used, and in which flavor? Which macro packages and fonts, on which CPU? What consultation, if any, on layout and design? What output device or devices? What printing system and what paper stock? Supplying this information should not discourage anyone from a more extensive description of the problems and triumphs of book-production, but let's start with the information.

As these lists grow, we will be able to answer questions about TeX in the most effective way of all, by pulling a book off the shelf and saying, "*That* is TeX."

The urge to predict is an occupational hazard of this sort of office, so I will venture a single guess that some time in the near future the TeX community and the TeX Users Group are going to grow very suddenly indeed. I would not be at all surprised to see them double in a single year, though I will not go so far as to guess which year. The future looks very interesting indeed.

* * * * * * * * * * *

MINUTES OF THE TeX USERS GROUP STEERING COMMITTEE MEETING

July 13–14, 1983
Palo Alto, Calfornia

The Steering Committee of the TeX Users Group met on July 13 and 14 to conduct business. An ad hoc version of the Finance Committee met on July 12 and drafted recommendations for action by the Steering Committee. Those recommendations and the Steering Committee actions taken were:

(1) In order to reduce clerical resources required by late membership renewals, two rates for dues should be established for 1984: a normal rate of $28, with a discount for renewal by January 31 of $3 for an early renewal rate of $25.
The Steering Committee agreed that an early renewal discount to help reduce clerical costs was needed and established a basic membership rate of $30, with a $10 dollar discount for renewals paid prior to January 31; that is, memberships paid prior to January 31 will cost $20.

(2) Institutional memberships for 1984 should be priced as follows:

educational	$200	with $25 discount per course or meeting
non-educational	$300	with $35 discount per course or meeting

The Steering Committee approved this recommendation.

(3) The majority of our foreign memberships include postage for air mail delivery. In order to reduce administrative costs, all non-North American memberships should be priced to include air mail.
With the information from Ray Goucher that air mail would cost about $5 per year, the Steering Committee decided to price non-North American memberships at $35 per year, subject to the above $10 early renewal discount.

(4) Back issues should be priced consistently at $15 per issue. The current scheme which sets the price for each back issue of TUGboat at the cost of membership for that year is confusing to administrate and even more confusing to explain to new members.
The Steering Committee agreed that back issue prices be consistently priced at $15/issue.

(5) The Finance Committee felt there was a potential for the abuse of institutional memberships by large organizations with multiple sites. They recommended that individual memberships which resulted from a single institutional membership be restricted to a single address or location.
The Steering Committee felt that this would prevent several of our current institutional members from being able to justify subsequent institutional memberships. Frequently a large firm may have only one TUG member in each of several sites and would not be willing to pay an institutional membership for each. The Steering Committee therefore decided that all of the individual memberships associated with a single institutional membership need not be at the same address.

(6) Some members expecting to attend only one day of the meeting have inquired about a rate

for partial attendance at the meeting. The Finance Committee recommended that registration is already complicated to administer and that there should not be a partial meeting rate.

The Steering Committee approved this recommendation.

(7) Questions about potential TUG-sponsored courses were raised without recommendations: Should TUG sponsor future TeX-related courses? Should these courses be always connected with the annual meeting? Should they be scheduled at other times and places?

The Steering Committee was in favor of the concept of offering TUG-sponsored courses on topics related to TeX and typesetting at various times and places in in addition to the traditional course with the annual meeting. Topics suggested for such courses included LaTeX, output devices, book design, output routines, and AMS-TeX internals. A motion to authorize courses courses in TUG's name and to hire professionals to teach as appropriate was tabled until after the General Meeting.

(8) There should be geographic area coordinators in addition to the current Site Coordinators.

The new title Area Coordinator was created. A request for volunteers and/or areas which feel the need for an Area Coordinator will be published in the next TUGboat. If appropriate, we will consider proposing a bylaw change next year.

(9) Provision should be made for annotating the membership lists to indicate members who do TeX consulting.

Ray Goucher was authorized to determine an appropriate fee for such annotation and make provision for these notes in the membership lists.

Arthur Keller has offered to teach a TeX for Beginners class at Stanford during quarter break next June or August. The class would last one week and consist of morning lectures followed by afternoons of lab time using LOTS. The course would be aimed at technical people who already understand at least one editor to minimize the amount of time spent teaching how to use the computer and to maximize the time spent learning TeX.

We need to provide in the budget for Ray's time; the AMS will no longer subsidize his considerable services to TUG. The Steering Committee recommended the following addition to the proposed TUG bylaws to provide explicitly for his services:

> The Finance Committee has the power to secure the services of a Business Manager who will report to the Finance Committee.

It was resolved that the costs of travel to the TUG meeting of the Executive Committee be covered if necessary to enable them to attend and to empower the Finance Committee to waive the conference fee when appropriate.

As its final item of business, the outgoing Steering Committee passed a resolution thanking Ray Goucher for the fine job he did in setting up this conference and for all of his service on behalf of the TeX Users Group.

At a brief meeting on July 15 the outgoing Steering Committee met to select members of the new Steering Committee as provided in the newly adopted Operating Procedures. This being accomplished, the outgoing Steering Committee adjourned.

Respectfully submitted,
Susan Plass

* * * * * * * * * * *

July 15, 1983

The following persons were present at the meeting:

Barbara Beeton
Lance Carnes
Barry Doherty
Chuck Dupree
Richard Furuta
Raymond Goucher
Arthur Keller
Don Knuth
Pierre MacKay
Monte Nichols
Susan Plass
Lynne Price
Joey Tuttle
Sam Whidden

On Friday, July 15, 1983, the Steering Committee of the TeX Users Group met at Stanford University. The Business Meeting of TUG took place the previous afternoon, and new officers were elected: Pierre MacKay, president; Joey Tuttle, vice president; Chuck Dupree, secretary; and Sam Whidden, treasurer. The new officers attended the Steering Committee meeting.

At this meeting the first order of business was to reconstitute the Steering Committee. The following issues were discussed:

- It was decided to send letters to the Steering Committee members (whose names appear on the top half of the inside front cover of TUGboat). The letters are to request information about future plans for participation in TUG and the Steering Committee. Thus the Committee plans to determine which members will continue to be active in committee work.
- There was general sentiment that Robert Welland should receive a letter of thanks for his

work as Editor-in-Chief of TUGboat. Barbara Beeton, who has been the Managing Editor, was appointed to replace him.

- The new members were present, and previous members will be contacted; thus, the Steering Committee has been reconstituted.
- There was a good deal of discussion of next year's TUG meeting, centering around the possible dates. Tentative dates were set, and Arthur Keller is attempting to determine that the dates are as convenient as possible, given the constraints discussed. Final decision on dates was delegated to the Finance Committee. The tentative dates are:

August 13–17	Arthur Keller's TeX users course
August 20–22	TUG meeting
August 23–24	short course

Arthur has suggested that the short course concern either the writing of complex macros or the design of typeset formats.

Joey Tuttle has graciously agreed to make arrangements for next year's meeting, as he did for this.

- The Finance Committee was also reconstituted. The four new officers became members of the Finance Committee, and nominations of Arthur Keller and Rilla Thedford were approved by the Committee. Fortunately, both have accepted.
- The Committee wholeheartedly agreed to reappoint Ray Goucher as Business Manager, reporting to the Finance Committee. He was also made an ex officio member of the Steering Committee.
- There was some discussion of the possibility of holding a meeting of TUG members on the East Coast. Such a meeting would not be a formal business function, but would consist of short courses, seminars, birds-of-a-feather sessions, and the like. A TUG meeting on the East Coast might draw a number of new faces, who would buy memberships and contribute their knowledge to the group. It was agreed that we request opinions from the general membership on whether and where a meeting would be held.
- Finally, there was a short discussion of the state of TeX in Europe. Ignacio Zabala has volunteered to serve as coordinator of European and United States TUG activity, and the Committee was happy to accept.

Submitted by Chuck Dupree,
Secretary,
August 18, 1983.

* * * * * * * * * * *

Summary of the Technical Program
TeX Users Group Meeting
Stanford University, July 11–15, 1983

AMS-TeX82 Beginning Users Course

Monday and Tuesday, July 11-12, were devoted to a short course for beginning users of AMS-TeX82, presented by Mike Spivak. All sessions of the course were videotaped, and will be available for purchase or lease (see cover 3 for details). A summary of AMS-TeX82 commands begins on page 103; arrangements are being made to include the macro package and documentation files on the TeX82 standard distribution tape. Mike will begin in November to rewrite the manual, *The Joy of TeX*.

TUG Meeting

Wednesday morning, July 13, began with an introduction to TeX and TUG for new users, presented by Ron Whitney. This covered essentially the same ground as his presentation at the 1982 TUG Summer meeting; that session was written up in TUGboat Vol. 3, No. 2, pages 9–12.

Status of TeX82

Don Knuth gave an update on the current condition of TeX82 and answered questions from the audience. The most recent status information appears in the article by David Fuchs beginning on page 72.

The Site Coordinators reported the current status of TeX82 on their architecture groups' equipment. Birds-of-a-Feather sessions were held by many of the groups. The high points are reported in Site Reports elsewhere in this issue.

Using TeX82

Two practical sessions were held for users with different levels of TeX experience. Arthur Keller chaired a tutorial for beginners, and Don Knuth presented some new tricks for macro wizards. Don's first session was a guided tour through `TRIP.TEX`; some problems from his second session appear in the Macro Column.

Leslie Lamport described his document formatting package, LaTeX. The impetus for LaTeX was Leslie's desire for a tool as well-organized and easy to use as Scribe, but with the full power of TeX underneath. Several common document types are described, and the user is given the ability to choose different styles of titles, running heads, page arrangement, and the like from a diverse predefined set, enabling the user to concentrate on the content of his document. Bibliographic references are maintained as a separate database, and compiled for each document as needed. LaTeX is current to TeX version .97, and will be upgraded later in the year to version 1. AMS-TeX has adopted a number of LaTeX syntax conventions, and a future possibility is the mutual adaptation of these two packages to permit access to the mathematical capabilities of AMS-TeX within the document structuring framework of LaTeX.

Roger Woodsmall of Intergraph Corporation gave a presentation on the way his organization has implemented TeX in conjunction with graphic work stations used for a CAD/CAM system and linked to a VAX/VMS. One of their goals is to integrate line art with the text of manuals and other technical publications. Fonts are stored in vector rather than raster format, to avoid rotational problems. Third-party word-processing software manages the input, which is later translated to TeX control sequences. A proprietary program, Interpage, merges text and graphics, and has the ability to scale fonts. Output is to an Autologic typesetter; graphics are built up using graphics "fonts", comprising various line segments of suitable weights, dimensions and orientation.

Chuck Dupree described the document production system now being used by the DEC VAX Software Documentation Group. This group produces 15,000 pages of technical documentation per year. They wanted a system which would support multifunction text processing, to include: draft documents, typeset documents, slides, help files, glossaries, etc. Document storage and retrieval and high-quality, direct VAX-to-typesetter output were also important considerations. The system actually developed is based on generic coding of data, and includes automatic page makeup and the direct typesetter link. Generic coding requires that data be separated from applications. Elements are identified by their structural content rather than by their appearance on a printed page (thus **keyword** or *newterm* would be identified, rather than font specification). TeX is neither the input language nor, as of the time of the meeting, the typesetting language used. But TeX's principles conform to the generic coding concept, and translation of a generically-coded document to TeX is straightforward. Conversion of the typesetting function to TeX is a very attractive option.

Output devices

David Fuchs gave a summary of the status of output devices and drivers. Details appear in his article on page 72, also on the output device chart on page 71.

Representatives of several output device manufacturers gave presentations, held open houses, or were available for discussion. The following (in alphabetical order) were represented.

Autologic now manufactures three typesetters suitable for interface to TeX output:

APS-5	Micro-5	Nano-5
4000 lpm	1000 lpm	300 lpm
57, 70, 100, 108pc	57, 70pc	70pc
~$130,000	~$72,000	$29,000 base ~$35–41,000 for TeX

The APS and Micro are suitable for high-resolution CAD/CAM work. Basic resolution is 723 dots/inch. Speed is in terms of newspaper lines per minute. A high-speed interface, 19.2KB, is available for the APS and Micro. Rotation is available on, and unique to, the Micro. Two auxiliary devices were discussed, the APS-44 (a font/logo digitizer) and the BitBlaster (a plain-paper output device with 300 dot/inch resolution). Autologic will make available Computer Modern fonts as supplied by Stanford, and will update them, for $2,500, supported.

Hewlett-Packard has installed TeX on a work station, the 68000-based series 200. The output device is a desk-top laser printer, with 300 dot/inch resolution and output speed of 12 pages/minute; this should be available about the end of the year. A hard disk is needed for font storage; font compression is an option. Supported fonts are designed for the printer, not METAFONT fonts, although METAFONT fonts will also be available. A maximum of 32 fonts may be used on one page. This is a shared-resource system, with a maximum of 4 work stations per group, and up to 4 groups, or 16 work stations, per printer resource. Cost of a single station is about $50,000, including printer and a 10 megabyte Winchester disk; for a 16-station setup, per-station cost is about $15–17,000, plus about $30,000 for the printer. 1.5 megabytes of memory are required on a station to support TeX.

Imagen held an open house, at which was introduced their new 480 dot-resolution printing engine, which is manufactured by Canon as is the printing engine of the Imprint-10. See their ad on page 128.

A representative of Quality Micro Systems described the Lasergrafix 1200, a printing system based on a Xerox 2700 engine, with 300 dot resolution and a speed of 24 pages per minute. The machine is cassette loaded, with two cassettes which can change automatically, or be loaded with two different paper sizes; cassette capacity is 250 sheets. The system is 68000-based. Interfaces are available to a wide variety of computer equipment. 1.4 megabytes of memory are required for an 8.5×14 inch page; the full raster is generated to permit intermixing of text and graphics. About 30 fonts can be active at once. The hardware cost is $25,000 with the simplest interface; some interfaces add up to $1,250. QMS handles a Talaris "software support kit" for the Lasergrafix, which includes a .DVI translator and graphics support, and runs on VAX, DEC 10s and 20s; the Talaris software is fully supported, and is priced at $3,500.

FONTS

Chuck Bigelow announced the ATypI (Association Typographique Internationale) seminar on electronic and traditional methods of letter design, to be held the first week of August at Stanford (the program appeared in TUGboat Vol. 4, No. 1, page 33). He also showed off an advance copy of the August 1983 issue of *Scientific American*, which contains an article on digital typography by him and Donald Day. Presentations by three of his students followed, on current work in electronic type design, with emphasis on METAFONT.

John Hobby described his work METAFONTing Chinese characters. Each character is composed of one or more distinct segments, or radicals. The radicals, in turn, are composed of strokes. John's approach, which is truly "meta" in nature, defines the strokes in such a way that their shape within a character adjusts automatically to the shape of the space occupied. New spline routines were created to yield smoother curves, and "no surprises". These fonts have been constructed in three weights: a "normal" weight, as would be used in books or journals; a bold version, similar to that used for newspaper headlines or posters; and an extra-light version. The style of the stroke terminators varies with the weight: small flourishes accompany the normal weight; the bold strokes have squared-off, blunt ends; the flourishes on the extra-light strokes are quite prominent compared to total stroke length. John's work has been published as a Stanford Computer Science Department Report.

Lynne Ruggles described software for converting meta-fonts back and forth between property-list and .TFM form (PLtoTF, TFtoPL) and between character and .PXL form (PXtoCH, CHtoPX). Property-list (for font metrics) and character (for character shapes) files are readable by human beings on a terminal screen, and may be edited to make ad-hoc adjustments for testing, or to "clean up" the images especially of small-size fonts for low-resolution output devices. This font-manipulation software is written in WEB.

Dave Siegel described the construction of METAFONT descriptions for several alphabets drawn by Hermann Zapf for use in mathematical composition, and known collectively as "Euler". The final implementation is not truly "meta"—outlines and the "double draw" technique have been used extensively. The results are very good at 10-point, but they can be scaled only linearly, and their acceptability decreases sharply with increase or decrease in point size. (A lower-case Fraktur alphabet developed by Scott Kim using more nearly meta-descriptions may scale.) A bit-pad interface was developed to input point coordinates for the METAFONT descriptions; use of this tool has cut initial entry time to about a half hour per character. The means by which Euler fonts may be made generally available has not yet been determined.

HOW TO OBTAIN TeX82

The generic WEB source for TeX82 and associated software is available on tape from Maria Code; an order form is included in this issue on page 129. The generic tape will include change files for several computer types, but some architecture-specific tapes have now been made available, and users of those types of hardware are advised to obtain the specific distribution tape. Details are on the order form. Please note that Maria Code is only the tape distributor. While she can tell you what version of TeX82 is on the current tape, she cannot answer technical questions. Questions regarding particular implementations, availability of output drivers, etc., should be directed to the Site Coordinators.

PLANNING AHEAD

The technical program was arranged by Joey Tuttle. He has agreed to perform the same function for next summer's meeting, and would welcome any suggestions or volunteers; he can be reached at I. P. Sharp Associates, Palo Alto, (415) 327-1700.

Barbara Beeton

* * * * * * * * * * *

List of Participants, TUG Summer Meeting and Introductory AMS-TeX82 Short Course Stanford University, July 11–15, 1983

In the following list, names of persons attending only the meeting are not specially marked; attendees at only the Short Course are starred; attendees at both the Meeting and the Short Course are flagged by a †.

*Anderson, Keith – Automatic Data Processing
†Armetta, Lois – Lawrence Berkeley Laboratory
Ballance, Bob – Hewlett-Packard Research Labs.
*Barrera, Josephine – U.C./Lawrence Berkeley Laboratory
Beatty, Stuart – Hewlett-Packard
Beebe, Nelson – University of Utah
†Beeton, Barbara – American Mathematical Society

Bentley, Nancy – Unidot, Inc.
†Berry, Paul – I. P. Sharp Associates
Bertelsen, Erik – University of Aarhus
*Boozer, Shirley – Stanford Linear Acceleration Lab
†Bringmann, Michael – Quality Micro Systems, Inc.
*Brock, Rosemary – Computer Systems Laboratory, Stanford University
Bronson, Mark – Lawrence Berkeley Laboratory
Brotsky, Dan – Massachussetts Institute of Technology
†Brower, Virginia – Stanford Linear Acceleration Laboratory
Brown, Fon – Hewlett-Packard
†Brown, Malcolm – Center for Information Technology, Stanford University
†Bryant, Sara – McGraw-Hill, Inc.
*Buckle, Monique – Université du Québec à Hull
†Buckle, Normand – Université du Québec à Hull
Burr, Norman – Lawrence Livermore Laboratory
Burton, Mark – Scan Laser Limited
Campanaro, Frank – Autologic, Inc.
Carnes, Lance – TeXeT
Carter, Belinda – SAS Institute
Cerofolini, Luigi – University of Bologna
*Chappell, Elsie – SRI International
Chessman, Samuel – TYX Corp.
†Cheung, Lucy – Stanford Linear Acceleration Lab
†Christopher, Lydia – Center for Reliable Computing, Stanford University
Clement, Colin – Hewlett-Packard
Crumly, Jim – Hewlett-Packard
Curtis, Dan – Imagen Corp.
Cuzzo, Clint – Hewlett-Packard
Day, Chris – Lawrence Berkeley Laboratory
*Deus, Patty – American Mathematical Society
Diffie, Whitfield – Bell-Northern Research
†Doherty, Barry – American Mathematical Society
†Dupree, Chuck – Digital Equipment Corp.
Durland, Tom – Information Handling Services
Eastman, Pat – CADTEC Corp.
*Eldridge-Diaz, Evelyn – Artifical Intelligence Lab, Stanford University
*Ferandin, Bette – Stanford Linear Acceleration Lab
Ferguson, Michael – INRS - Telecommunications
Fidelman, Susan – University of California
†Fina, Pat – Massachusetts Institute of Technology
*Flynn, Kathleen – Stanford University
†Fossati, Elaine
*Frary, Nancy – Lawrence Livermore Laboratory
†Fuchs, David – Computer Science Department, Stanford University
Furuta, Richard – University of Washington
†Gaffney, Charles – Pascal News
*Garrett, Liz – Lawrence Livermore Laboratory
*Gaskins, Robert – Bell-Northern Research
*Gessner, Barbara – U. S. Geological Survey
Gold, Lynn – Kestrel Institute
Goldberg, Donna – Center for Information Technology, Stanford University
Goucher, Raymond E. – American Mathematical Society
Grosso, Paul – University of Michigan
Guenther, Dean – Washington State University
†Gurel, O. – IBM Scientific Center
Haley, Larry – Hewlett-Packard
*Hall, Diana – Computer Science Department, Stanford University
Hansen, Linda – Lawrence Livermore Laboratory
*Harrison, Michael – University of California
Heatlie, Valerie – Lawrence Berkeley Laboratory
†Hsu, Agnes – Computation Center, National Research Council of Canada
†Ion, Patrick – Mathematical Reviews, American Mathematical Society
†Isensee, Philip – Iowa State University
Jackson, Calvin – California Institute of Technology
Jacobson, Van – Lawrence Berkeley Laboratory
†Jaeger, Dennis – Automatic Data Processing
Janko, Wolfgang – Karlsruhe University and University of California, Los Angeles
Janson, Barbara – American Mathematical Society
*Jensen, Sharon – Stanford Linear Acceleration Lab
†Kaylor, Richard – Hewlett-Packard
Keller, Arthur – Computer Science Department, Stanford University
Kellerman, David – Oregon Software
†Kim, Julius – Grumman Data Systems
†Kitajima, Yasuko – Hugh Graham Secretarial Service
Knuth, Donald – Computer Science Department, Stanford University
Krapp, David – CALMA
†Lamport, Leslie – SRI International
Langley, Jim – Hewlett-Packard
†Laurent, Kevin – U. S. Geological Survey
Leary, Pat – Sandia National Laboratories
Leung, Sheon – Signetics, Inc.
MacKay, Pierre – University of Washington
Mallett, Rick – Carleton University
*McCluskey, Joseph – Center for Reliable Computing, Stanford University
*McCormick, Jutta – Computer Science Department, Stanford University
*McGilton, Henry – Sun Microsystems
†Melen, Shirley – Center for Information Technology, Stanford University
Merrell, Greg – MSM Company
†Mesirov, Jill – American Mathematical Society
*Milenkiecz, Joan – University of California
†Moffat, Shannon – Center for Information Technology, Stanford University
Mohr, August S. – Editor, UniForum
*Monroe, Beverly – U. S. Geological Survey
Moortgat, Hugo – University of Santa Clara
*Moran, Rita – Computer Systems Laboratory, Stanford University
Morris, Tom – University of North Carolina
*Naugle, Mary – Texas A&M University
Naugle, Norman – Texas A&M University
Nichols, Monte – Sandia National Laboratories
Olum, Ken – Fairchild
Osmond, Carolyn O. – Information Handling Services
†Pagan, Carol – U. S. Geological Survey
Palais, Richard – Brandeis University
†Parker, Don – Quality Micro Systems, Inc.
†Penny, S. Keith – Union Carbide Corp.
†Pesch, Roland – I. P. Sharp Associates
*Pickering, Jane – Computer Science Department, Stanford University
†Plass, Susan – Center for Information Technology, Stanford University

† Platt, Craig – University of Manitoba
Poggio, Margaret E. – Lawrence Livermore Laboratory
Price, Gary – Rational Machinery
Price, Lynne – CALMA
Pritchard, Paul A. – Information Handling Services
† Ragan, Donal – Arizona State University
† Renzetti, Phyllis – U. S. Geological Survey
Rodgers, Dave – Textset, Inc.
† Roesser, Jim – Science Typographers, Inc.
Saito, Nobuo – Keio University
Samuel, Arthur – Computer Science Department, Stanford University
Schneble, Jack – McGraw Hill, Inc.
† Schulze, Bernd – University of Bonn
• Seeto, Anna – Xervice Center
Selfridge, John – Mathematical Reviews, American Mathematical Society
† Senn, Mark – Consultant
Severance, Charles – Michigan State University
• Shanks, Alison
Shepherd, Gary – Sandia National Laboratories
† Sih, Peter – IBM
† Skinner, Lydia – Kestrel Institute
Smith, Hart – Lawrence Berkeley Laboratory
† Spivak, Michael
† Spragens, Alan – Stanford Linear Acceleration Lab
Sterken, Jim – Textset, Inc.
Sturdivant, Gary – Ascent, Inc.
Sublett, Betty – Lawrence Berkeley Laboratory
• Sullivan, Carol – U. S. Geological Survey
Sultan, Eric – ViewTech, Inc.
† Sumner, Thos – University of California
† Thedford, Rilla – Mathematical Reviews, American Mathematical Society
† Thomas, Margaret – Science Applications, Inc.
† Tuttle, Joey – I. P. Sharp Associates
† Van Stone, Lisa – I. P. Sharp Associates
Villere, Gary – Informatics General Corp.
† Walker, Gordon L. – American Mathematical Society
• Waller, Jan – Texas A&M University
• Want, Jan – Texas A&M University
Westmiller, Jane – ESL, Inc.
† Whidden, Sam – American Mathematical Society
Whipple, Edgar – Lawrence Berkeley Laboratory
† White, Howard – Stanford Linear Acceleration Lab
Whitney, Ron – American Mathematical Society
Woods, Cheryl – American Mathematical Society
† Woodsmall, Roger – Intergraph Corp.
Young, Richard – Oracle
† Zajkowski, Roger – McGraw Hill, Inc.
Zamora, Tom – Washington State University

* * * * * * * * * * *

FINAL REPORT OF THE BYLAWS COMMITTEE

Susan Plass

Operating Procedures for the TEX Users Group were drafted by a committee composed of Lance Carnes, Barbara Beeton, Ray Goucher, Sam Whidden, Lynne Price, and Susan Plass. These Procedures were revised and then adopted at the General Meeting of the TUG membership on July 14, 1983. The Operating Procedures as adopted are published on the following pages.

TEX USERS GROUP
TUG OPERATING PROCEDURES

Adopted July 14, 1983

Article I
Name

1.0 The name of the organization is TEX Users Group, to be abbreviated TUG.

Article II
Purpose

2.0 TUG is established to serve members having a common interest in TEX, a system for typesetting technical text, and in METAFONT, a system for font design, to:

2.1 Provide a forum for the interchange of information about TEX and METAFONT — their status, use, and future.

2.2 Promote the exchange of information concerning the use of computers and computer peripheral equipment with TEX and with METAFONT.

2.3 Establish standards and provide channels to facilitate the exchange of macro packages, etc., between TUG members.

Article III
Membership

3.0 Membership shall be open to anyone interested in TEX or METAFONT. A member in good standing is a person for whom dues for the current year have been paid. Hereinafter the term member shall be construed to mean a member in good standing.

Article IV
Organization

4.0 The overall structure of TUG shall consist of:

4.1 Officers

TUG shall have four (4) elected officers. These are President, Vice President, Secretary, and Treasurer.

4.2 Executive Committee

4.2.1 Composition

The Executive Committee shall consist of the President, the Vice President, the Secretary, and the Treasurer of TUG.

4.2.2 Responsibilities

To adopt interim procedures and policies when necessary on behalf of TUG, subject to the ultimate approval of the membership.

4.3 Finance Committee

4.3.1 Composition

The Finance Committee shall consist of the Executive Committee and two other members chosen by the Steering Committee.

4.3.2 Responsibilities

The Finance Committee is empowered to take whatever actions are necessary between meetings, with the prime directive to keep TUG out of debt.

4.3.3 Business Manager

The Finance Committee is empowered to obtain the services of a Business Manager for TUG.

4.4 Steering Committee

4.4.1 Composition

The Steering Committee shall consist of the Executive Committee, Site Coordinators, Wizards, and other active TUG members nominated by the Steering Committee. The Steering Committee shall have the power and responsibility to appoint and to remove its own members. Steering Committee members whose term is not otherwise limited shall serve for two years; members may be reappointed for successive terms.

4.4.2 Responsibilities

The Steering Committee is the governing body of TUG. It has the responsibility to conduct the business of the organization.

4.4.3 The President shall have the power to call a meeting of the Steering Committee.

4.4.4 Twenty-five percent (25%) of the members of the Steering Committee shall constitute a quorum.

4.5 Majority Rule

A motion passes when more committee members vote for it than against it.

Article V
Duties of Officers

5.0 The individual responsibilities of the officers shall be:

5.1 President

5.1.1 To serve as the Chief Executive and Operational Officer of TUG.

5.1.2 To perform normal administrative functions.

5.1.3 To chair meetings of the General Membership, the Steering Committee, and the Executive Committee.

5.2 Vice President

5.2.1 To serve in the absence of the President.

5.2.2 To undertake other administrative duties as designated by the President.

5.3 Secretary

5.3.1 To maintain records pertaining to TUG business.

5.3.2 To conduct TUG correspondence.

5.4 Treasurer

5.4.1 To serve as the Chief Financial Officer of TUG.

5.5 Site Coordinator

5.5.1 To provide coordination for information about TEX and METAFONT with respect to a specific computer architecture.

5.5.2 To provide technical direction for the future growth of TEX and METAFONT.

Article VI

Meetings

6.0 Time of Meetings

6.1 General Membership

A business meeting of the general membership shall be conducted in conjunction with each TUG Conference. Such a conference will normally be held at least once per year.

6.2 Committee Meetings

Standing Committees and ad hoc Committees are appointed by the President and will meet as needed.

Article VII

Voting by General Membership

7.0 Who may vote

Any individual member in good standing may cast one vote.

7.1 Majority rule

A yes vote by more than fifty percent (50%) of the members present is necessary to carry any action.

7.2 Procedure

A voice vote will normally be considered sufficient for business actions. A show of hands may be requested when the outcome is in doubt.

Article VIII

Election of Officers

8.0 Eligibility for Nomination

Any active member in good standing may be nominated for TUG office. Said member must accept the nomination before being placed on the ballot.

8.1 Nomination Procedure

8.1.1 At the meeting prior to that at which the terms of any officers normally expire a Nominating Committee shall be appointed by the President for the purpose of suggesting candidates to fill those offices. This committee shall nominate at least one member to fill each office up for election.

8.1.2 In addition any member may have his name placed in nomination by submitting a petition to the Nominating Committee at least 30 days prior to the election signed by two (2) other members in good standing.

8.2 Election Procedure

8.2.1 All elections will be conducted by secret ballot. The candidate receiving the most votes shall be elected.

8.2.2 The requirement for a secret ballot may be waived by the President for an election for any office in which there is only one candidate.

Article IX

Vacancies

9.0 Vacancies shall be filled in the following manner:

9.1 Officers

When an office becomes vacant for any reason, the President shall appoint a member to serve out the remainder of that term. When the office of President becomes vacant, the Vice President shall become President for the remainder of the President's term and shall then as President appoint a member to serve as Vice President.

9.2 Site Coordinators

New and replacement Site Coordinators may be nominated by any member of the Steering Committee and shall be confirmed by a majority vote of the Steering Committee.

Article X

Amendments

10.0 Amendments to these Operating Procedures shall be adopted by the following procedure:

10.1 Submission of Amendments

10.1.1 Amendments to the TUG Operating Procedures shall submitted by petition to the Executive Committee. Each petition must bear the signatures of twenty (20) TUG members or of five (5) TUG members and the approval of the Steering Committee.

10.1.2 A petition concerning an amendment to these Operating Procedures may be initiated by any member of TUG in good standing.

10.2 Amendment Ratification

10.2.1 Any amendment to these Operating Procedures must be published with a ballot in the next newsletter distributed after submission of the petition, or in a mailing to the membership.

10.2.2 Amendments to these Operating Procedures must be approved by an affirmative vote of two-thirds of the votes cast by TUG members.

10.2.3 Only ballots postmarked within 45 days of the date of mailing of the amendment will be included in the tally.

Article XI

Ratification and Implementation

11.0 Ratification

These articles shall be ratified by an affirmative vote of 2/3 of the members present at the next general business meeting of TUG.

11.1 Implementation

11.1.1 Officers

An election shall be held immediately upon the adoption of these articles to select the four TUG officers. The President and Treasurer shall be elected to two-year terms; the Vice President and Secretary shall be elected to one-year terms. All subsequent terms shall be for two years.

11.1.2 Site Coordinators and other Steering Committee Members

Upon the adoption of these articles the existing Steering Committee shall select a successor committee.

TUG TREASURER'S REPORT

August 31, 1983

	Budget 1983	Actuals as of 8/31	Estimated thru 12/31	Budget 1984
Income:				
Membership/Publications[1]				
Individual	$ 13,000	$ 10,690	$ 11,000	$ 14,000
Library	600	1,665	1,750	1,800
Postage	900	1,199	1,200	-0-
Back issue sales	1,350	3,466	3,600	2,500
Supplements[2]	500	506	600	500
Institutional Membership[3]				
Educational	2,000	5,380	5,380	5,500
Non-educational	3,000	3,482	3,482	5,000
Meetings[4]				
Summer	4,000	14,398	16,000	15,000
Course	4,000	13,787	15,000	15,000
Manufacturers' fees	-0-	450	450	600
Fall	7,000	-0-	-0-	6,000
Other				
Videotape sales/rental	3,000	4,998	6,000	3,000
Advertising & mailing list sales	500	125	300	500
Royalties (TeX manual)	500	-0-	-0-	1,000
Total income	$ 40,350	$ 60,146	$ 64,762	$ 70,400
Expenses:				
TUGboat (2 issues)				
Printing	2,200	1,151	2,478	2,596
Postage	800	351	826	944
Editorial services	4,300	1,974	4,720	5,074
Clerical services	4,000	2,604	5,900	6,490
Computer expenses	3,200	2,926	4,012	4,130
Meetings				
Summer/Course	4,960	1,333	5,900	5,900
Fall	4,960	-0-	-0-	2,360
Other				
Supplements	350	293	295	354
ANSI meetings[5]	1,180	-0-	1,392	1,652
Legal and tax consulting	500	-0-	590	590
Advertising membership & TUGboat[6]	1,500	-0-	-0-	-0-
Postage, general mailings	1,100	2,250	2,950	2,950
Printing back issues	1,500	5,267	5,310	-0-
Printing, other	680	1,224	1,770	2,360
Administrative support[7]	-0-	8,428	10,620	11,800
Subsidies[8]	1,180	-0-	-0-	1,180
Video tape duplication	432	1,604	2,360	1,180
Miscellaneous[9]	1,868	2,887	3,540	4,130
Total expenses	$ 34,710	$ 32,292	$ 52,663	$ 53,690
Summary:				
Balance forward	$ 24,607	$ 24,607	$ 24,607	$ 36,706
Total income (budgeted or actual)	40,350	60,146	64,762	70,400
Total expenses (budgeted or actual)	34,710	32,292	52,663	53,690
Balance	$ 30,247	$ 52,461	$ 36,706	$ 53,416

Treasurer's Report

(Continued from preceding page)

Notes:

The 1983 budget column is identical to that published in TUGboat Vol. 4, #1. All expense figures include an AMS overhead charge of 18%.

1. There are 738 memberships/subscriptions: 188 foreign, including Canada and Mexico; 558, U. S. Beginning in 1984, foreign air mail postage is included in membership/subscription fee.
2. 55 copies of reprints of Max Diaz's "Fácil TEX" have been sold.
3. TUG has 44 institutional members, listed on the inside cover of this issue. 20 – educational; 24 – non-educational.
4. 84 individuals attended Michael Spivak's "Introductory AMS-TEX82 Users Course" and 135 members participated in the summer meeting conducted at Stanford University, July 11–15, 1983.
   Representatives from Autologic, Hewlett-Packard, Imagen and Quality Micro Systems gave presentations. A second meeting on the East Coast is planned for 1984.
5. Support is budgeted for attendance at one meeting of ANSI X3J6.
6. Advertising of 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.
7. While TUG was becoming established during 1981 and 1982, the American Mathematical Society made available the services of Ray Goucher at no charge. He manages all the administrative details associated with TUG, to include daily income/expense accounting, budgeting/treasurer's reports, coordination of all aspects of meeting preparations/accounting, publicity, advertising, in addition to numerous other details. He was appointed TUG Business Manager at the Stanford Meeting in July.
8. Money available to the Finance Committee to subsidize travel and membership fees for individuals when appropriate.
9. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

* * * * * * * * * * *

REPORT ON ANSI X3J6

Lynne Price

The American National Standards Institute Technical Committee ANSI X3J6 on Computer Language for the Processing of Text meets concurrently with the International Standardization Organization's Expert Group (ISO/TC 97/SC 5/EG CLPT) on the same topic. As described in TUGboat Vol. 3, No. 1, the committee's charter is to define a standard language for tasks such as text editing, text formatting, and generalized markup. Availability of a standard will promote the ease with which document source files can be moved from site to site and will reduce retraining of individual users who transfer (even temporarily) to a new system. The sixth working draft of the language specification was submitted for comment to ISO/TC 97/SC 5 (SC 5 is the subcommittee on Programming Languages of TC 97, the technical committee on Computers and Information Processing) on June 15. The comment period extends through September 15 and the following draft is expected early next year.

The standard is divided into a series of parts. The parts are distributed as separate documents so that individuals who wish to use only part of the material may conveniently do so. The parts are enumerated in the following table, where an asterisk marks parts that were included in the June version:

*	1	General
*	2	Vocabulary
*	3	Programming Language
*	4	Entry and Editing Functions
*	5	Formatting and Composition Functions
*	6	Document Markup Metalanguage
	7	Markup Support Functions
	8	Binding to the Graphical Kernel System
	9	Application to "What You See is What You Get" (WYSIWYG) Processing
	10	Registration Procedures

Of the six parts so far submitted to SC 5, Parts One, Two, Five and Six are the most polished. Part Three, which is not yet completed, describes an interpretive language geared toward text processing. While the editing, formatting, and markup functions described in other parts can be implemented in the language described therein, the implementation language is not dictated by the standard. The language of Part Three does provide the end user with the ability to build macros of editing and formatting commands. The current version of Part Four (Entry and Editing Functions) is an expression of a possible philosophy. It notes that while opinions on text editor commands and syntax are highly individual, most sophisticated features are built from a very small number of editing primitives. It suggests therefore that the standard define any text editor implemented in the language of Part Three to be compliant and that an Appendix to the document contain examples of such definitions for typical line, character, and screen editors. These examples would suggest to users of related editors how the listed code could be modified to incorporate each user's preferred enhancements. Part Ten will describe procedures for registering items such as document styles, markup conventions, and formatting macros. Although registered items will not be standardized, they will be available to all users who wish to access them. Users at one site can thereby take advantage of work done by individuals at another location.

Copies of the current working draft can be obtained from

Charles D. Card
Sperry Corporation
M.S. C1-NE10
Blue Bell, PA 19424

In addition, visitors are welcome to observe and participate in the meetings. The next sessions will take place October 24–28 in Detroit, January 23–27 in Anaheim, and May 14–18 in Phoenix. TUG's X3J6 liaison is Lynne Price, who will happily transmit feedback from TUG members to the committee. TUG members are of course encouraged to contact her for more information.

* * * * * * * * * * *

Software

* * * * * * * * * * *

A NOTE ON HYPHENATION

Donald Knuth

Some people occasionally write to me about hyphenations that TeX finds, because TeX doesn't always match the way their own dictionaries do it. In almost all cases, such discrepancies prove to be unavoidable, because different dictionaries don't agree with each other.

Consider, for example, the word "process." TeX hyphenates 'pro-cess', in accordance with *Webster's Third*, while many dictionaries say 'proc-ess'. I don't believe TeX does anything wrong here; indeed, I would never like to see 'proc-' at the end of one line and 'ess' at the beginning of the next, since I would probably have already pronounced the word wrong in my mind before my eyes reached the second line.

Another interesting case is "performance." Here *Webster's Third* and *American Heritage*, etc., say 'per-form-ance', but TeX says 'per-for-mance'. This case is interesting because it turns out that *Webster's New Collegiate*—published many years after the infamous *Third*—also says 'per-for-mance'; so does *Random House Unabridged*. The latter hyphenation is evidently more consistent with other words of English, since TeX's patterns are based on a large mass of data, so here we see a trend in dictionaries to be more uniform.

So far I have run across only one improperly hyphenated word, in thousands of test pages: 'exam-sman-ship'. But I wasn't too upset, because I deserved such a fate after making up that word.

Bob Filman has also shown me the very unfortunate 'Di-jkstra'; there's a case where many TeX users will want an entry in their exception dictionaries.

I think it would be useful to have a catalog of desirable hyphenation exceptions maintained somehow in *TUGboat*; let me begin this with its first entry, 'Dijk-stra'. Let me also beg readers not to contribute further entries unless they are sure that all of the standard authorities disagree with TeX's hyphens. (Sometimes we have found that *Webster's* is not as good as others, but we usually have followed it.)

And one more point: If any computer center decides to preload different exceptions from those in plain TeX (i.e., in the file HYPHEN.TEX), the changed exceptions should not under any circumstances be put into HYPHEN.TEX or PLAIN.TEX. All local changes should go into a separate file, so that TeX will still produce identical results on all machines. You can run your program elsewhere by simply sending the file of local changes. In fact, I recommend not preloading those changes, but rather assuming that individual users will have their own favorite collection of updates to the standard format files.

* * * * * * * * * * *

Editor's *note:* At the TUG meeting, Don gave some interesting statistics on the performance of the hyphenation algorithm in TeX82: For the 676 most common English words, hyphenation is 100% correct. And 89.7% of *all* English words are hyphenated correctly. So among the remaining 10% there must be a few words that might show up in a TeX file. The editor of TUGboat will be happy to keep a list.

Don also showed a simple way to find out what TEX is doing:

```
RUN TeX
\relax
\showhyphens{...}
```

As many words as desired can be entered at once; all hyphenation points will be shown for each word. Actually, TEX finds two hyphenation points are found in `Di-jk-stra`; one of them is right.

* * * * * * * * * * *

TEX VS. INITEX

David Fuchs

This note deals with TEX vs. INITEX, `\dump`, FMT files, and how to make a 'pre-loaded' TEX. Those who have already seen this information on the TEXhax computer bulletin board may be interested to read the last paragraph, which is new.

First, a review. In the best of all possible worlds, there would only have to be a single version of TEX. Every TEX file would start with a line that `\input` a big package of lots of macros and fonts. In the real world, reading many macros each time you run TEX can become fairly tedious because it is not instantaneous. Reading in fonts is even slower, since most systems take their time in opening files, and each font mentioned requires TEX to access a separate TFM file. To help speed things up, we introduced the FMT file. INITEX is able to read in macros and fonts (in fact it can do everything TEX can) and then it can dump its entire internal state out into a single file when it is given the `\dump` command. Such a file is called a FMT, or 'format' file. Thus, if we:

```
RUN INITEX
\input plain
\dump
```

INITEX will exit and leave us the file PLAIN.FMT. We can then run TEX (or INITEX, for that matter) and say:

```
RUN TEX
&plain
```

and we'll be in exactly the same state we were in before we did the `\dump` above. Thus, we can continue by saying `\input myfile`, and TEX will behave as if we had `\input plain` and then `\input myfile`. We have already won, because it is faster to read in PLAIN.FMT than it is for TEX to process PLAIN.TeX from scratch.

In fact, it was not necessary actually to say `&plain` to regular TEX, because it automatically reads in PLAIN.FMT, unless you override it by saying `&fancy` or something. (Of course, INITEX does not automatically read in PLAIN.FMT if we don't ask for another FMT file; otherwise there would be no way to create PLAIN.FMT in the first place!) If you have another set of macros, say THESIS.TEX, you could:

```
RUN INITEX
\input thesis
\dump
```

to make THESIS.FMT, and then users of that macro package would be able to get their jobs going quickly by saying:

```
RUN TEX
&thesis mythesis
```

(`\input` is assumed even after the `&thesis`.)

On some systems, this is as far as we can go: instead of reading lots of files and doing lots of processing to get TEX set up to read the user's input, TEX need only read one FMT file each run, and do a little processing. But FMT files are pretty big; maybe there is an even better way to go. Well, consider a system on which the user is able to halt a program in mid-execution, and save away the entire thing to be continued at a later date. If TEX with no FMT file built-in is called 'VIRTEX', then we can build a TEX with PLAIN.FMT built-in by:

```
RUN VIRTEX
&plain
(Wait for TeX's * prompt, and then HALT it)
SAVE TEX
```

If a TEX wizard does this and puts the results on a common system area, then when users give the `TeX` command they will actually get a copy of TEX with PLAIN already set up. If it only were that simple everywhere! There are a few things that need to be considered before you attempt to do this sort of thing on your system. We have to delve deeper into the implementation ...

When the user runs this saved TEX, is it started up from the beginning or is it continued from where it left off? In all the systems I've seen that have this sort of capability, the program is actually restarted from the beginning, but the global data areas are not re-initialized. Well, that's ideal for TEX, because of TEX's use of a magic global variable called *ready_already*. When TEX starts up, it takes a look at *ready_already* (without first assigning a value to it). If the value is not 314159, then TEX figures that there is not a FMT file built-in, so it goes ahead and initializes itself, and reads in PLAIN.FMT (unless the user asks for a different FMT file). Then it sets *ready_already* to 314159, in case the user decides to halt execution and save the core image. If the core image were then to be restarted, *ready_already* would still be 314159, and TEX would realize that it already had a FMT file loaded in, so it would skip over the whole initialization process.

Great. But there are problems. On some systems, the runtimes are completely oblivious to being interrupted and restarted. On these systems, the scheme described above works fine. Other systems, however, require the program to have come to a graceful exit before being saved; otherwise when it is restarted, everything will run amok. (Typically, the runtime might become confused by the fact that the log file was not closed when the core image was saved.) On such systems, you'd think you could build a TeX with PLAIN.FMT built-in by:

```
RUN VIRTEX
&plain
\end
SAVE TEX
```

because the \end will cause TeX to complete its execution by 'falling out the bottom.' The only problem is that, just before it does this, TeX sets *ready_already* to 0. The idea behind this is that if you are running on a system in which global variables are initially random, and you get put in the same spot in real memory as an old TeX job, then your TeX might get fooled into thinking that it has a FMT built-in (since the prior TeX job left *ready_already* at 314159), when all it really has is the left-over internal state of someone else's TeX job. If the last TeX job cleared *ready_already* to zero, then you have less chance of being fooled, so this is what we have TeX do by default. On the other hand, the previous job might have gotten interrupted in the middle, and you'd be fooled anyway, so on this kind of system you're best off using another technique for checking built-in-ness. Anyway, the point is that you should remove the final *ready_already* := 0 if you have the kind of system that requires you to say \end to get all the files closed and other things cleaned up before saving a TeX + (FMT file).

The astute reader may be asking "That still doesn't explain why there needs to be INITeX as well as VIRTeX. Why aren't things set up so that we can just

```
RUN VIRTEX
\input plain
HALT
SAVE TEX  ?"
```

The answer is that VIRTeX is missing one or two capabilities that INITeX has. If you take a look at the WEB-language source for TeX, you'll see that there are two macros, named INIT and TINI, that delimit various sections of code. The idea is that if you Tangle TeX.WEB with INIT and TINI defined as null, then you'll get an INITeX when you compile the resulting Pascal program. If INIT is defined to evaluate to '{', and TINI '}', then when you Tangle you'll get almost the same Pascal program, but with portions commented out. Compiling this program gives VIRTeX.

Well, what are the sections that are commented out, and why? The code in question consists of the modules that initialize TeX's hash tables, read in the POOL file, and process hyphenation \patterns. We can get away with this because the hash table, string pool, and hyphenation trie all get dumped to the FMT file, so non-INITeXs need not be able to recreate them from scratch. This is strictly an efficiency move, since there is no reason to bog regular TeX down with the extra code and data space necessary to process these data structures. Only the missing \patterns primitive can be detected by the ordinary user, but only foreign language hyphenation hackers should be messing with that.

The other code not in VIRTeX is the stuff that handles \dump itself. Of course, if you have an infinitely fast machine with no address-space limitations, and if extra memory is free, then a simple change to TeX would let it load \patterns and do \dumps, and INITeX could be thrown away. Actually, so could FMT files, and the \dump feature.

Early tests on IBM VM/CMS systems seem to indicate that if the FMT files are blocked up into very large records, the system is able to process them quite quickly, so it is not necessary to do pre-loading. It turns out to be convenient to have seperate INITeX and VIRTeX executable programs on this system, however. The advantage is not only that VIRTeX is somewhat smaller due to the missing INIT code, but we also turn on the DEBUG-GUBED and STAT-TATS switches in INITeX, and compile it with the optimizer turned off and the full run-time checks turned on. Thus, INITeX is a relatively large, slow TeX, but it is good not only for creating new FMT files, but also to help check out any suspected bugs in TeX, as well as help in heavy-duty macro debugging.

* * * * * * * * * * *

TeXHAX SUMMARY

David Fuchs

There were a few typos in the TeXhax summary in the last issue of TUGboat. Page 6, second column, last paragraph should say **@**, not **@/**. Similarly, page 7, line 7 should say **@** instead of just **@**. Finally, the program example in the right column should have 8em removed in two places. This might be a good place to mention that there is one more new feature in WEB: due to popular demand, the format of change files has changed. The new system lets you change just the middle of a module, and also warns you if the code you're changing has changed itself. Here's how it works: The change file has any number of changes of the form

```
⟨Comment lines⟩
@x
⟨Old lines⟩
@y
⟨New lines⟩
@z
```

When Tangle or Weave sees a line that matches the first "⟨Old line⟩", it checks that all the "⟨Old lines⟩" match the main web file. If not, an error message is given. In either case, the "⟨Old lines⟩" are discarded, and the "⟨New lines⟩" take their place. Comments can be given on the **@x**, **@y**, and **@z** lines, and between changes. All **@x**, **@y**, and **@z**'s must be at the beginning of a line.

To help you bootstrap the new Tangle and Weave, it may help to know that the code which has changed was in non-system-dependent modules, so your old change files should (almost) work with the new web files when run on the old Tangle. So you should be able to create a working new Tangle easily, using your previous change file; and you can test it out by making a new-format change file for Tangle and seeing if the new Tangle can recreate itself.

The 'almost' is because you may have to change `\count0` to `\pageno`, to conform to the new TeX. You may also have to change **@**'s in TeX files, and lots of **@@**'s in WEB files, to ~'s, since tilde is now the tie character. Also, take a look at the note in `TeX82.DIF` about the big `\set` and `\the` change introduced in version 0.98 to see if that affects you. Don't let all this scare you; there haven't been any problems reported on the systems that have already been done.

A few people have gotten into trouble by trying to alter Tangle to output all reserved words and identifiers in lowercase. Note that Tangle looks back in its output buffer for things like `MOD` and `DIV` to help it decide when it's OK to do constant folding. If you aren't careful, you'll end up with a Tangle that outputs incorrect programs. The red WEB manual has a Tangle listing with an index entry 'uppercase' that points to all the relevant modules.

Most of our WEB programs now include a 'history' feature that keeps track of how the run went. Generally, the results are either 'good', 'warning issued', 'error encountered', or 'fatal error'. The history variable is used to print a final status message as each program ends, and it can also be used to send a status code back to the operating system on those systems that support such things.

There are a few things to watch out for when installing the new TeX. The module "⟨Globals in the outer block⟩" has been renamed "⟨Global variables⟩", so you should check that your change file always refers to "⟨Global ...⟩" or "⟨Glob ...⟩". Also, all references to single-letter identifiers that used to look like "**x**" are now in the form "|*x*|". This may affect your change files slightly. Similarly, `$\ldots$` has been changed to `\dots` in many comments; a few files that used to use `\ifodd` now use `\ifeven` (which takes a number rather than a counter as a parameter); `\ifabsent` is now called `\ifvoid`. All instances of "**debug**", "**init**", and "**stat**" have **@!** in front of them, so that they will be indexed. Discretionary hyphens (`\-`) have been removed from words that TeX82 can hyphenate.

The module that defines the macros "*qi*" and "*qo*" now also defines "*hi*" and "*ho*", so if this module is in your change file, you'll have to alter it in the obvious way. Similarly, the module that defines "*set_glue_ratio_zero*" now includes two new macros, "*float*" and "*unfloat*", to aid in porting TeX to systems where "*glue_ratio*" can not be "*real*". The macro "*float_const*" has been added to point out the few places that use a floating point constant in the code.

All of our WEB programs that use the procedure "*input_ln*" now remove spaces at the end of lines. This means that all input files and macro packages will be impervious to being moved to and from fixed-line-length systems. Since most change files include an altered version of "*input_ln*", you'll probably have to change a little code here.

Some of the modules in TeX that changed in interesting ways (interesting for the TeX installer, anyway) can be found by searching for the strings "*wlog*", "*wterm*", "*name* <= 16", "*name* > 16" and "*read_open*" in `TeX.WEB`. Also note a change of variable name ("*n*" vs. "*m*") in "⟨Input the first line ...⟩". The "*input_ln*" procedure now has a second parameter that tells whether it should do an initial "*get*"; this simplifies the reading of the first line of `\input` and `\read` files. It also should aid installation on systems with either 'lazy-lookahead' or 'interactive' terminal files.

A new macro called "*wake_up_terminal*" is now called each time TeX is about to issue an error message. On systems where the user might flush the output to the terminal (with a control-O, for instance), this macro marks all the good times to turn output back on. In order to help TeX do this, standard error messages now say “*print_err*(""message...” instead of “*print_nl*(""! message...” The “*print_err*” takes care of printing the exclamation point and space, after it calls “*wake_up_terminal*”. Note that “*wake_up_terminal*” is called from a few other places as well.

Other items from TEXhax: On Tops-20, the ‘log file’ spoken of in the TEXbook will have an extension of `.LST`, and on VAX/VMS, it will be `.LIS`, so as to avoid conflicting with `.LOG` files produced by batch jobs. The new extensions were chosen to match the systems’ conventions about compiler listing file names (TEX is a compiler, after all).

Watch out for an unfortunate ‘feature’ of `\everypar`: If a paragraph begins in a group, as in `...\vskip 5pt {\it Horizontal mode...` then the effects of the `\everypar` are local to that group (except for `\global`s, of course) even if the group ends before the paragraph does.

The VM/CMS version of TEX that is now available through our standard distribution channels represents the combined efforts of many people, who I won’t list here for fear of leaving someone out. If you have an IBM system that you’d like to be able to re-compile TEX on, you should be aware of a bug in the Pascal/VS optimizer found by Craig Platt.

```
program foo;
var i: integer;

procedure changeit; forward;

procedure doit;
begin
i := 1;
changeit;
writeln('Should not be one: ',i:1);
end;

procedure changeit;
begin
i := 2;
writeln('Should be two: ',i:1);
end;

begin
doit;
end.
```

The Pascal/VS developers have a fix for this, but the procedure for obtaining it is totally obscure. You’re on your own, as far as I can tell.

Here are the current contents of the “Red Alert” file mentioned above:

A few bugs were discovered in TEX 0.999. Details of the changes may be found at the end of the `TeX82.BUG` file. The `TRIP` files had to be modified slightly to accommodate these changes. There was also a change to the TEXbook to make it consistent with the fact that `\input` files no longer have a blank line appended automatically.

A bug in `WEBHDR` caused index entries to have double **@**’s where there were supposed to be single **@**’s. This error shows up in all the red manuals.

A bug in Weave caused any unchanged module followed by a module whose first line had changed to be incorrectly marked as changed. This is fixed in Weave 2.2 with a small change in the logic of “*get_line*”.

A minor change to `\finsm@sh` in `PLAIN.TEX`.

And here’s what `TeX82.BUG` has to say about changes made after the Version 0.999 red listing of TEX82 (these changes are shown in **@x**, **@y**, **@z** format for clarity; they are not meant to be used in an actual change file):

248. Module 1215, allow space in `\read` *n* to `\cs` (by FY, July 25, 1983)

```
@x patch in get_r_token routine
begin restart: get_token;
@y
begin restart: repeat get_token;
until cur_tok <> space_token;
@z
```

. . . .

249. Module 498, we must stack the current if type (FY, July 27)

```
@x patch in conditional routine
begin @<Push the condition stack@>;
          @+save_cond_ptr := cond_ptr;@/
@y
@!this_if: small_number; {type of this conditional}
begin @<Push the condition stack@>;
          @+save_cond_ptr := cond_ptr;this_if := cur_chr;@/
@z
```

Also replace *cur_if* by *this_if* in modules 501, 503, 506. The following patches do only what is necessary to make things work:

```
@x
     print_cmd_chr(if_test, cur_if);
@y
     print_cmd_chr(if_test, this_if);
@z
@x
if cur_if = if_int_code then scan_int
          @+else scan_normal_dimen;
@y
if this_if = if_int_code then scan_int
          @+else scan_normal_dimen;
@z
@x
if cur_if = if_char_code then b := (n = cur_chr)
          @+else b := (m = cur_cmd);
@y
if this_if = if_char_code then b := (n = cur_chr)
          @+else b := (m = cur_cmd);
@z
```

250. Module 507, \ifx need not put a control sequence in hash table (July 29)

```
@x
get_token; n := cs_ptr; p := cur_cmd; q := cur_chr;
get_token; if cur_cmd <> p then b := false
@y
get_next; n := cs_ptr; p := cur_cmd; q := cur_chr;
get_next; if cur_cmd <> p then b := false
@z
```

. . . .

251. Module 86, message is lost (noticed by HWT, July 31)

```
@x
print("..."); print_ln; return;
@y
print("..."); print_ln; update_terminal; return;
@z
```

. . . .

252. Don't put empty at end of \input file! (Aug 1) [This simplifies the rules and the program, and also gets around a bug that occurred at the end of files with \endlinechar < 0.]

```
@x Module 362:
@ An empty line is inserted at the end of the file, if the
last line wasn't already empty, because |input_ln|
sets |last := first| when it discovers an |eof|.
@^empty line at end of file@>

@<Read next line of file into |buffer|, or
      |goto restart| if the file has ended@>=
begin incr(line); first := start;
if not force_eof then
      begin if input_ln(cur_file, true) then {not end of file}
            firm_up_the_line {this sets |limit|}
      else if limit <> start then firm_up_the_line
            {if |pausing|, the user can add more lines}
      else force_eof := true;
@y
@ @<Read next line of file into |buffer|, or
      |goto restart| if the file has ended@>=
begin incr(line); first := start;
if not force_eof then
      begin if input_ln(cur_file, true) then {not end of file}
            firm_up_the_line {this sets |limit|}
      else force_eof := true;
@z
```

*** The changes above went into Version 0.9999, which was widely distributed.

. . . .

253. Ridiculous blunder made in change 146 (found by FY, August 16)

```
@x Correction to module 497
else loop@+begin q := cond_ptr;
      if link(q) = p then
            begin type(p) := l; return;
            end;
      if q = null then confusion("if");
@:this can't happen if}{\quad if@>
      q := link(q);
@y
else begin q := cond_ptr;
      loop@+ begin if q = null then confusion("if");
@:this can't happen if}{\quad if@>
            if link(q) = p then
                  begin type(p) := l; return;
                  end;
            q :=link(q);
            end;
@z
```

. . . .

254. Minor amendment to stat(s) printing (cf. change 129) (August 16)

```
@x in module 1334
      wlog_ln('  ', str_ptr - init_str_ptr : 1,
            ' strings out of ',
            max_strings - init_str_ptr : 1);@/
@y
      wlog('  ', str_ptr - init_str_ptr : 1, ' string');
      if str_ptr <> init_str_ptr + 1 then wlog('s');
      wlog_ln(' out of ',
@z
```

. . . .

255. Bug in \xleader computations (found by FY, August 18)

```
@x in module 592
@!lq,@!lr,@!lx: integer;
            {quantities used in calculations for leaders}
@y
@!lq,@!lr: integer;
            {quantities used in calculations for leaders}
@z

@x in module 626
      begin edge := cur_h + rule_wd;
      @<Let |cur_h| be the position of the first box, and
                  set |leader_wd| to the spacing between
                  corresponding parts of boxes@>;
      while cur_h + leader_wd <= edge do
            @<Output a leader box at |cur_h|,
                  then advance |cur_h| by |leader_wd|@>;
@y
      begin edge := cur_h + rule_wd; lx := 0;
      @<Let |cur_h| be the position of the first box, and
                  set |leader_wd + lx| to the spacing between
                  corresponding parts of boxes@>;
      while cur_h + leader_wd <= edge do
            @<Output a leader box at |cur_h|, then
                  advance |cur_h| by |leader_wd + lx|@>;
@z

@x in module 627
            leader_wd := leader_wd + lx;
@y
@z

@x in module 628
cur_h := save_h + leader_wd;
@y
cur_h := save_h + leader_wd + lx;
@z
```

```
@x in module 635
      begin edge := cur_v + rule_ht;
      @⟨Let |cur_v| be the position of the first box, and
               set |leader_ht| to the spacing between
               corresponding parts of boxes@⟩;
      while cur_v + leader_ht <= edge do
            @⟨Output a leader box at |cur_v|, then
               advance |cur_v| by |leader_ht|@⟩;
@y
      begin edge := cur_v + rule_ht; lx := 0;
      @⟨Let |cur_v| be the position of the first box, and
               set |leader_ht + lx| to the spacing between
               corresponding parts of boxes@⟩;
      while cur_v + leader_ht <= edge do
            @⟨Output a leader box at |cur_v|, then
               advance |cur_v| by |leader_ht + lx|@⟩;
@z
@x in module 636
            leader_ht := leader_ht + lx;
@y
@z
@x in module 637
cur_v := save_v-height(leader_box) + leader_ht;
@y
cur_v := save_v-height(leader_box) + leader_ht + lx;
@z
```

Also insert the following in modules 619 and 629:
@! lx: scaled; {extra space between leader boxes}

. . . .

256. \/ should apply to ligatures! (August 20)

```
@x in module 1113
var f: internal_font_number;
               {the font in the |char_node|}
begin if is_char_node(tail) and (tail <> head) then
      begin f := font(tail);
      tail_append(new_kern(char_italic(f)
               (char_info(f)(character(tail)))));
@y
label exit;
var p: pointer;
               {|char_node| at the tail of the current list}
@! f: internal_font_number; {the font in the |char_node|}
begin if tail <> head then
      begin if is_char_node(tail) then p := tail
      else if type(tail) = ligature_node then
               p := lig_char(tail)
      else return;
      f =: font(p);
      tail_append(new_kern(char_italic(f)
               (char_info(f)(character(p)))));
@z
```

```
@x later in that same module
end;
@y
exit: end;
@z
```

. . . .

258. Redundant code eliminated (August 27)

Module 531 needn't set and reset *name_in_progress* [but it's harmless].

. . . .

259. Bug: \input shouldn't occur during font size spec (Spivak; fixed August 27)

```
@x module 1258
@ @⟨Scan the font size specification@⟩=
@y
@ @⟨Scan the font size specification@⟩=
name_in_progress := true;
            {this keeps |cur_name| from being changed}
@z
@x module 1258
else s := -1000
@y
else s := -1000;
name_in_progress := false
@z
```

. . . .

261. Serious data structure error (found by Todd Allen, August 29)

```
@x module 478 (an error introduced in change 231)
   q := the_toks; link(p) := link(temp_head); p := q;
@y
   q := the_toks;
   if link(temp_head) <> null then
         begin link(p) := link(temp_head); p := q;
         end;
@z
```

. . . .

262. Minor patch for efficiency (August 29)

```
@x module 466
         begin store_new_token(info(r)); r := link(r);
@y
         begin fast_store_new_token(info(r)); r := link(r);
@z
```

* * * * * * * * * * *

Output Devices

* * * * * * * * * * *

OUTPUT DEVICES AND COMPUTERS

Table I: Proof-Quality Devices

	Diablo 630	Epson MX-80	Facit 4542	Fla. Data OSP	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						OCLC						OCLC		
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												Livermore		
Siemens BS2000								GMD Bonn						
Sun				Textset		Sun Inc.								Textset
VAX (Unix) *										UC* SantaCruz		Cal. Tech.*		
VAX (Unix)						UC Irvine						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

The following combination has been dropped:

DEC 10/Xerox XGP/Stanford

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	Linotron 202
Amdahl (MVS)*				Washington State U*	
DEC 20 *	AMS*				Adapt, Inc.*
HP3000			Univ. of Sheffield		
IBM 370 *		Info. Handling*			
Sun		Textset			
Univac 1100 *				Univ. of Wisconsin*	
VAX (VMS)		Intergraph †			

* * * * * * * * * * *

Site Reports

* * * * * * * * * * *

NEWS FROM THE TEX PROJECT

David Fuchs

As of mid-August, the state of the world is as follows: All of the WEB listings have been re-printed in red covers. This is the final printing of TEX82 before it becomes a real book. The TEXbook was also re-printed in a red cover, missing only Appendices D and E; we will wait a few weeks to get people's reviews, then we'll phototypeset it in September, and off to the publisher it goes. The whole set of preprints is available through our distribution service, along with the latest distribution tapes.

TEX is frozen. The only changes being made are bug fixes. We are currently on version 0.9999 (no kidding). It is slightly different than the red manuals, but all changes made to the system are listed in a file called `RED.ALERT` on the distribution tapes, and they will also be published in TUGboat (see below for the changes that have happened so far). When the TEXbook is available, we'll declare version 1.0, and there will be much rejoicing and throwing out of old TEX manuals. By the way, bugs in the code are now worth $5.12, while bugs in the TEXbook will get you $2.56.

Tapes formats available are generic ASCII, generic EBCDIC, Tops-20, VAX/VMS, VM/CMS (by the time you read this, we hope) and MVS (likewise). Tapes for Berkeley VAX/Unix are available from R. Furuta, as mentioned in his note on page 74. There are also some new font tapes for 200, 240 and 300 dot/inch devices. Mike Spivak's first shot at AMS-TEX for TEX82 is also included on the distribution tapes.

The Vax/VMS port was done by me, using the new Pascal 2.1 compiler from Digital. The distribution tape is in standard `BACKUP` format, and has all the sources as well as `EXE` files. You do not need to have the compiler to run the `EXE` files, since they were produced with `LINK/NOSYSSHR`. If you do have the compiler, you'll want to re-link everything so that the runtimes will be shared at execution time. If you get this tape, a good way to start is to look at the file `[.TEX.DOC]VMS.DOC` for further details.

The Tops-20 port was done by me, using the Hedrick Pascal compiler from Rutgers. The distribution tape is in standard `DUMPER` format, and has all the sources as well as `EXE` files. You'll need to have the compiler to run the `EXE` files, since the runtimes are mapped in from a common file at runtime. This shouldn't be a problem, since the compiler is virtually free. If you get this tape, a good way to start is to look at the file `<TEX.WEB>TOPS20.DOC` for further details.

The VM/CMS port was a combined effort, using the IBM Pascal/VS compiler. The distribution tape (by the time you read this) is in standard `TAPEDUMP` format. You shouldn't need to have the compiler, since the executable programs are included. If you get this tape, a good way to start is to look at the file `VMCMS DOC` for further details.

At present, there are no device drivers on the distribution tapes. We would be happy to add any drivers that people see fit to contribute. Other, related software can also be included where appropriate.

Actually, a bit more needs to be said about the font distribution. First, we have totally moved over to the new "AM" series of Computer Modern. For a while we were using a mix of some old "CM" and some incomplete "AM" fonts, but now the "AM" series is complete. The older "AM" and "CM" fonts are now out of date, because some characters have changed position, and all of the empty slots have been filled in (lots of new characters, including a proportionality sign). `PLAIN.TEX` knows about the layout and the `\skewchar` features in the new "AM" fonts, so you'll need them to run TEX82. The new "AM" fonts are also frozen in the sense that characters won't be moving around any more, but there will be improved versions of the characters themselves some months down the road. Finally, some of the font names have changed a little, so that the whole system is more logical and self-consistent (the old CMATHX is now AMEX10, while CMDUNH is now AMDUNH10, etc.).

As discussed at the TUG meeting, the latest TEX has a few interesting notions about magnifying fonts. In particular, a font can be magnified by a factor of 1.2 by saying `\font\foo=amr10 scaled \magstep1`. An entire document can be magnified 1.2 times by saying `\magnification\magstep1`. Similarly, `\magstep2` is a factor 1.44. In fact, you can use up to `\magstep5`, each one being a factor of 1.2 bigger than the previous. There's also `\magstephalf`, which is almost 1.1 (actually, `\sqrt{1.2}`). Anyway, what we're distributing is all of our AM fonts in the lower `\magsteps`, plus the main ones in the higher `\magsteps`. (The "main ones" are exactly the ones from `PLAIN.TEX` that are not named `\preloaded`.)

One interesting result of doing things this way is that there need be only one series of pixel files for 200 and 240 dots/inch devices. Consider: the PXL file for `\magstep2` of a 200 dot/inch font is the same as the PXL file for `\magstep1` of the same font at 240 dots/inch. Indeed, we now provide a single distribution tape for both 200 and 240 dots/inch. The only complication is that `\magstephalf` fonts from each series don't fit into the other series, so recipients of the distribution tape may delete the inappropriate fonts if desired.

The main problem with providing so many magnifications is that the amount of font data is getting too large to fit onto the distribution tapes. For this reason, we are forced to change the way that fonts are stored on the tapes. Now, each byte from the file is a single byte on the tape. The block size is 8000 bytes, and the last block of a file is only as long as it needs to be.

One item discussed at the TUG meeting was that it would be nice if TUG members could dial up a Stanford computer and read a bulletin board with recent TEX news. As soon as we get our Sun machines going, we'll try to provide this service. This should allow folks without CSnet access to read TEXhax between TUGboat issues.

* * * * * * * * * * *

TEX82 ON CP-6

Rick Mallett
Computing Services
Carleton University
Ottawa, Ontario, Canada.
Tel: (613) 231-7145

TEX82 version 0.95 is up and running on the Honeywell CP-6 computer system here at Carleton University, and thanks to the folks at Stanford, it took very little effort to get it going. From an implementer's point of view TEX has certainly come a long way from the days of TEX in Pascal, which took us more like 6 months to get working, although I must admit that a good portion of that time was spent debugging our Pascal compiler. Getting TEX to run efficiently is quite another matter though— we've been "changing" TEX all summer and we only finished about 3 hours ago.

Our biggest problem stemmed from the fact that, although our operating system allows a core image to be saved and restored, the Pascal runtime is such that the I/O streams are not properly re-initialized. Therefore, we had to take the other obvious approach and block the FMT and TFM files and change TEX to handle them. This turned out to be more work than we had expected, mainly because we kept tripping over the hacks in the I/O routines to get around the problem that the DEC-20 handles terminal and file I/O differently.

Before further describing our conversion difficulties, however, I would like to tell you a bit about who we are and what we have been doing with TEX. I'd also like to comment briefly on the CP-6 operating system, since I suspect that very few people have ever heard of it.

We first implemented TEX in 1981, when we used it to help debug the Pascal compiler that we were writing for CP-6, but without an output device it did not see a lot of use. However, in the spring of 1982 we set to work to build a controller for the Canon LBP-10 laser printer, and by September we were finally printing pages formatted by TEX. Since that time we have printed over 19000 pages, from course notes, reports, and theses to an article for a Ukrainian hiking journal, and more recently a linguistics text that required the entire phonetic alphabet.

Now for CP-6. CP-6 is a relatively new (ca. 1980) operating system, which runs on the large Honeywell 36 bit machines, and it was designed to be similar to the CP-V operating system which ran the Xerox Sigma and 560 series computers. Some of you may recall that Xerox dumped support for its large scale computers in 1976; fortunately for us, Honeywell Information Systems hired the same team of people who had created CP-V and put them to work writing a similar operating system to run on their 36 bit machines. The result is a "state-of-the-art", large-scale, time-sharing system that seems optimal for someone who needs something larger than a VAX 11/780, with equivalent or better functionality, but who does not want to pay the exorbitant prices that IBM likes to charge for a "large" machine.

Actually, there isn't all that much to say about our implementation of TEX on CP-6, as most of the problems we encountered have been reported in the last issue of TUGboat. We had to change all the file names, of course, and modify *Inputln* so that the first character of the first line of terminal input wasn't lost, and set *name_length* before opening the pool file, but for the most part it was pretty straightforward. Unfortunately, we wasted a bit of time writing PXL to CHR and CHR to PXL conversion programs, as we did not realize that these routines had already been written and are now on the distribution tape, but it was not that hard anyway. Like the people porting TEX to VAX/UNIX, we had to modify our compiler to support `CASE OTHERWISE`, but, since we had written the Pascal compiler, it was fairly easy to change it, once we had convinced ourselves that it really was acceptable to provide features not included in the "proposed" standard. (The experience of putting up TEX80 without `CASE OTHERWISE` was a rather convincing argument.)

A more serious problem arose from the fact that we had decided long ago to store all reals as 72 bit quantities, and we did not want to go back and change the code generator to use single precision reals. Although the address space on our machine is such that TEX fit easily even with the double-sized MEM array, our experience with TEX80 had shown us that we could not afford to make heavy use of TEX without seriously degrading system performance—it's pretty hard to run over 100 users with only 6 million words of real memory. Fortunately, the last issue of TUGboat arrived just at the moment that we were agonizing over what to do about the problem, and we learned from Pavel Curtis' and Howard Trickey's comments that modifying all references to the MEM array so that only the most significant bits of all reals are stored there isn't as hard as we had imagined. It turned out that we were able to make the modification in one afternoon. Of course, it should be even easier the next time around with the *float* and *unfloat* routines that have now been added to TEX.

Other than that there were no major problems to the initial conversion. As I pointed out earlier though, our latest round of performance improvements which involves replacing all the Pascal I/O was quite another story.

Finally, there is one problem that we have encountered with TEX that I suspect is widespread, but I have not seen it reported elsewhere. It turns out that TEX is so useful that everyone wants to use it, and we are still trying to figure out how we can afford to make TEX available without swamping the system. The solution that we are looking at is to charge for all TEX services, but we have not been forced to charge academic users before, and we are not looking forward to setting up the necessary bureaucracy. It sure seems stupid to have to penalize those students who go out of their way to create nice, neat, legible reports and theses. Nevertheless, it looks as though that's the way things will have to go around here for the time being.

* * * * * * * * * * *

IBM SITE REPORT

Susan Plass

TUG members from 13 IBM-architecture sites met in a Birds-of-a-Feather session at the July 1983 annual meeting. About half of these sites currently run a version of TEX; most of the rest are in the process of installing TEX on their systems. Of these sites, 6 run MVS, 5 run VM/CMS, 1 runs both MVS and VM, and 1 runs TSS. We explored the idea of breaking the IBM group into two groups, one for MVS and a second for VM. The consensus seemed to be that there was no need yet to break the group formally, but Alan Spragens of SLAC has volunteered to serve as a focal point for a VM-oriented information exchange. Give Alan a call if you have VM/TEX questions and especially if you have VM/TEX answers!

* * * * * * * * * * *

UNIX TEX SITE REPORT

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

Work on Unix TEX for Berkeley Unix, 4.1 bsd, has been proceeding quite uneventfully since our last report in these pages. Well, perhaps not completely uneventfully—we ran comparative benchmarks against the Unix and VMS TEX82, Version 0.97, implementations and were quite surprised to discover that the VMS version was several times faster than the Unix version. Thanks to Howard Trickey's Herculean efforts, and replacement of the critical inner loop input routines with C language routines, the Unix TEX82 has been sped up quite noticeably and we feel that the Unix running speed is now quite acceptable.

Several other people have contributed to TEX on Unix recently. Steven Correll of the Lawrence Livermore National Labs sent along a patch to allow an editor escape from TEX into **vi**. Howard added an escape for Gosling's **emacs** and this code is now in the distributed version of TEX82. Mark Senn sent a BBN Bitgraph DVI page previewer he wrote while at Purdue. This will be included in the Unix distribution when the necessary fonts become available. Further contributions to the Unix TEX distribution are invited.

[†] This work is funded, in part, by a grant from Northern Telecom.

The current version of TEX82 on Berkeley Unix, 4.1 bsd, is now 0.9999. Mike Harrison at Berkeley has verified that this version of TEX will also run on 4.2 bsd, and a number of sites are running it on 4.1c, as well. Some work has been done to get this version of TEX working on the Sun workstation, also running 4.2 bsd. We are eagerly awaiting news of this conversion effort.

Ordering TEX82 for 4.1 bsd

TEX82 is now available for sites running Berkeley Unix, versions 4.1, 4.1c, and 4.2. We are now making available a "beta test" distribution which includes sources for TEX and WEB, libraries, fonts, and whatever device drivers we can obtain. At the time of writing, we have drivers for the Versatec, thanks to Carl Binding of our Department, for the Imagen laser printer, thanks to Pavel Curtis of Cornell, and for the Symbolics LGP-1 laser printer. If you have DVI 2 drivers for other devices, please send them to us and we'll happily include them in future distributions. The font bit maps included in the distribution are for devices with resolutions of 200 pixels/inch, 240 pixels/inch, and 300 pixels/inch.

This distribution will be the latest version of TEX82 available to us (presently 0.9999). Note that versions before 1.0 are considered to be pre-releases, and hence the "beta test" designation. TEX82 has been frozen, however, and the only changes made between the present version and Version 1.0 will be bug fixes. If you want to wait for Version 1.0, please let us know when you send in your order and we'll hold it. Version 1.0 may be out by the time this report is published.

As the implementation uses a modified version of Berkeley's **pc** Pascal compiler, we will only be able to provide a complete distribution to those sites with source licenses for 4.1 or 4.2 bsd. We will, however, try to accommodate those with 4.1 or 4.2 bsd binary licenses. Please make sure that you indicate clearly whether you have a source or a binary license.

The size of the distribution has increased to about 20 megabytes, due mostly to increases in the size of the fonts area. However, most sites can get away with using quite a bit less disk space since only those font magnifications which are actually used on the intended output device need be loaded.

Tapes will be in **tar** format, blocked 20, and written at 1600 bpi, unless otherwise specified (we can also write 800 bpi).

To order, send a check for $50 (U.S. Funds) made to the University of Washington, a copy of your 4.1 or 4.2 bsd license, and your address to:

Richard Furuta
Computer Science, FR-35
University of Washington
Seattle, WA 98195

We would appreciate it if foreign sites could increase the amount of their check as appropriate to pay the added postal costs necessary for mailing the tape. If you have a CSNet, Arpanet, or uucp mail address, please include it and we'll add you to our electronic mailing list.

Please note that it is not necessary for you to send any of your Western Electric Unix Licenses—only the 4.1 or 4.2 bsd license. But please *do* remember to include the 4.1 or 4.2 bsd license—we've had to write or call many sites asking for it which delays things considerably. Please do not send purchase orders as we have no facilities for handling them.

TEX82 on Other Versions of Unix

Several people have expressed interest in porting TEX to varieties of Unix other than 4.1 bsd. We have heard from people who either have begun ports or are contemplating beginning ports to Amdahl's UTS Operating System and to 2.8 bsd. We will keep *TUGboat* readers up to date on progress on these ports. However, since a certain amount of time passes between *TUGboats*, I would like to ask those of you who are interested in one or another of these efforts to drop me a note. In this fashion, I'll be able to judge the amount of interest in each of these ports and I'll be able to send out announcements should they be completed before the next *TUGboat*.

If anyone is interested in attempting ports to other versions of Unix (say System III or V), we want to hear from you. We're quite anxious to see TEX become available on all versions of Unix and will try to help in whatever ways we can.

* * * * * * * * * * *

VAX/VMS

Monte C. Nichols
Sandia National Laboratory
Livermore, California

There's good news and not-so-good news to report for VAX/VMS. Through the good graces of David Fuchs (see page 72), TEX82 has been ported to VAX/VMS where it is up and running well. Unfortunately, there is no spooler available on the distribution tape at the present time. Hopefully, there will be soon.

The most recent version of TEX82 (.9999) for VAX/VMS is available from Maria Code (see page

129) in BACKUP format. Be sure to specify that you are on a VAX/VMS system. Hopefully, the .PXL files will also soon be available in BACKUP format.

Anyone with a spooler for TEX82 who is willing to have it put on the distribution tape, please call me. The interfaces listed in the table on page 71 are not on the distribution tape; information should be obtained from the sites listed.

* * * * * * * * * * *

REVISED VARIAN OUTPUT DRIVER IN VAX/VMS FORTRAN

Jim Mooney
West Virginia University

The VAX output driver for the Varian Plotter, written in VMS FORTRAN, has been updated to process Version 1 DVI files, and has received some other improvements as well. The first version of this program was reported in TUGboat, V2 No. 3, pp. 14–15.

The revised package was developed as a consulting project for Science Applications, Inc., McLean, VA, with the assistance of Buff Miner.

The new system consists of two programs, DVITOVAR and OUTTOVAR. DVITOVAR can produce raster files like its predecessor, which can be plotted separately by OUTTOVAR; alternately, it can drive the plotter directly.

Other improvements include better use of VMS logical names and command line parsing; an option to rotate the output 90 degrees; and control of magnification and page margins.

DVITOVAR accepts Version 1 DVI files. Alas, the stability of this version has been somewhat less than originally predicted. TEX has moved on to Version 2. A DVITOVAR for Version 2 is a possibility for the future.

The set of files comprising the DVITOVAR package has been provided to Oregon Software and will be included on future releases of their VAX/VMS TEX tapes.

A document describing the package is available from me on request. Technical questions may be addressed to me at:

Dept. of Statistics and Computer Science
West Virginia University
Morgantown, WV 26506
(304) 293-3607

Note: I cannot supply copies of the programs themselves, as I have no facility for making tapes. We do not even have a system capable of running TEX available to us at the present time.

* * * * * * * * * * *

Fonts

* * * * * * * * * * *

TEX FOR ARABIC SCRIPT

Pierre MacKay
University of Washington

The principal focus of our work with TEX at the University of Washington is the development of a capacity to typeset Arabic Script texts. For this we must extend and modify the basic program, but we are making every effort to ensure that our Arabic Script version of TEX will be an enhancement, and will leave all the basic features of TEX82 intact. The TRIP.TEX file will give us a good test of our success in this regard.

Nevertheless, our eventual DVI output will be distinct from that produced by other sites in the following ways:

1. The ID number will be 102. Our special driver programs will be set so that a driver for DVI102 will accept DVI2 files, but a standard driver will complain about an Arabic Script file. On the UNIX distribution tapes, incidentally, you will always get the standard DVI2 drivers unless you specifically ask for a DVI102 driver.
2. DVI codes 250 and 251, which are unimplemented in "vanilla" TEX, will be used to mark off right-to-left text in a horizontal list. These codes will be balanced at any completed level of `\hbox{}` before the surrounding `\vbox{}` is built. Code 250 marks the beginning or right-to-left text and code 251 marks the return to a normal TEX environment.
3. The " (double-quote) character is activated as a new primitive in TEX text input. It operates as a toggle in the way the $ (dollar-sign) math-mode toggle works. The environment between balanced pairs of " toggles is known as "reversed text mode" and may be useful for Hebrew text as well as for Arabic. Math-mode may be nested within reversed text mode and will be governed by the normal rules of math mode. A double quote character in math mode retains its normal TEX significance. If you want Arabic Script text inside math mode, you will have to `\hbox{}` it separately and to `\unhbox{}` it at the desired place.

None of this actually works yet, but we are getting very close. The TEX for Arabic Script project is supported in part by grants from Northern Telecom and BNR Inc.

* * * * * * * * * * *

"small" TeX

* * * * * * * * * * *

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

The July meeting at Stanford was informative and enjoyable. Several new small TeX versions were announced, with several more in progress.

Below is a grid summarizing the known implementations of TeX on small machines. The term "small" is as vague in the TeX world as it is in the rest of the industry. A loose definition of a small processor is one which either has a 16-bit word size, or one which will rest on a tabletop.

The data given under "processor time per page" is a rough approximation, usually given by the implementer. Since there is no "standard" page to compile to give a rated time, these times should not be used to predict anything important.

Under company and contact, these are the names of the implementer and not the manufacturer of the hardware. For a phone number or address, consult the most recent TUG membership list.

A brief note about the TYX products. They have used the smallest machines to date (IBM PC), and offer these implementations as fully-supported products. The version of TeX available (TYXTeX) is a rewrite in C of the SAIL version, and is closest to TeX80. TYX plans to update their version to TeX82 in the next year or so.

As the editor for small TeX, I receive several phone calls each month inquiring about microcomputer implementations of TeX. The most common request is from IBM PC users, and it would be useful to have a TeX82–WEB implementation to offer. All of you who would like to see this happen, and who would be willing to commit funding to such a project, contact me at the above address.

It is nice to see the variety of implementations, either completed or in progress. If you are currently using TeX on a small system, or are planning to, and the hardware does not appear in the grid below, be sure to contact me and supply information.

"small" TeX implementations

Manufacturer	Processor	TeX version	Processor time per page	Company and contact
Hewlett-Packard 3000	16-bit	TeX82	10–30	TeXeT, Lance Carnes
Hewlett-Packard 1000	16-bit	TeX82	10–30	JDJ Wordware, John Johnson
DEC PDP-11/44	16-bit			
Plexus, Onyx	Z8000	TeX80	10–20	TYX, Dick Gauthier
IBM PC	8086/8			
Apollo	M68000	TeX82	2–10	OCLC, Tom Hickey
Hewlett-Packard 9836	M68000	TeX82	6–10	HP Boise Div., Jim Crumley
Sun	M68000	TeX82[1]		Textset, Jim Sterken
Corvus	M68000	TeX82[2]		
Cyb	M68000	TeX82[1]		Texas A&M, Norman Naugle
Apple Lisa	M68000	TeX82[2]		
Masscomp	M68000	TeX82[1]		
Sage	M68000	TeX82[2]		

[1] in progress or recently completed
[2] hopeful

* * * * * * * * * * *

MACRO
O
L
U
M
N

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
527 Lakeside Drive
Sunnyvale, CA 94086

PROBLEMS FROM THE TUG MEETING

Framed Slides

At the July TUG meeting, Don Knuth conducted a Macro Wizards Roundtable. One problem raised was to define an output routine (to be used in constructing slides) that puts a rectangle around the completed page, with a variable-sized label appearing in a break in the bottom rule. Don's solution is:

```
\newdimen\fullsize
\fullsize = 7in
\output{\shipout\vbox{\hrule
                      \midpart
                      \nointerlineskip
                      \botline{macro for slides}}}

\def\sleaders{\leaders\hrule height 2.4pt depth -2pt\hfil}
\def\botline#1{\setbox0=\hbox{\tenrm #1}\ht0 = 0pt
               \hbox to \fullsize{\sleaders\ \box0\ \sleaders}}
\def\midpart{\hbox to \fullsize
                     {\vrule
                      \hfil
                      \innerbox
                      \hfil
                      \vrule }
             }
\def\innerbox{\vbox{\kern 10pt
                    \hbox to \hsize{\box 255}
                    \kern 10pt }
             }
```

Note that the font for the label is explicitly specified. Without the call to \tenrm in the \botline macro, a label would be printed in whatever font happened to be current when the page break occurred. Also note that the reference to \box 255 in \innerbox is forced to \hsize; or one could use a definition from Plain, \pagecontents. Either variation would protect against the rare possibility of a generated page whose width is not the current \hsize.

macro for slides

Multiple Marks

Another question dealt with a need for multiple marks. Consider, for example, a glossary in which entries are numbered. It might be useful to place guide words in the heading on each page and "guide numbers" in the footing. TeX's mark feature was specifically designed to save such labels, but only one mark can be specified at a time. While it is clear that the mark construct can be used for either guide words or guide numbers it is not obvious how to use it for both. Don suggested using TeX82's \ifcase construct. For example, the macro that starts a new entry might expand to include an instruction such as

```
\mark{Dachshund\or 20}
```

if "dachshund" is the twentieth word entered. The output routine could use the construct

```
\ifcase\expandafter 0\topmark
```

to find the guide word in effect at the top of the page and

```
\ifcase\expandafter 1\topmark
```

to find the guide number.

* * * * * * * * * * *

TUGBOAT MACRO INDEX

The following list catalogues macros that have appeared in TUGboat. Entries are listed by volume, number, and page as well as author's name. Items that could not be categorized by an obvious head-word have been listed under "miscellaneous". Many items refer to parts of large macro packages; users of other packages may find them valuable models for macros of their own.

Readers' comments on the format as well as the contents of this index are welcome.

ACM style	II:1	61, 82–83	A. Keller
Addresses	II:1	54	B. Beeton
	II:2	A-35	M. Díaz
Appendices	II:2	A-21	M. Díaz
Array operations	III:2	34–36	L. Lamport
Baseline, set to top of box	II:1	60, 77	A. Keller
Bibliography	II:2	A-25	M. Díaz
Boxes	II:1	59, 73	A. Keller
Box numbers, automatic allocation	III:1	33	M. Plass
Branching, *see* If			
Capital letters			
dropped initial	IV:2	81	Problems column
first line of paragraph	IV:2	80	Problems column
large ~ at beginning of paragraph	II:1	60, 78	A. Keller
	II:3	62	TeXarcana Class
	II:2	A-16	M. Díaz
	IV:2	81	Problems column
Roman numerals	II:1	120–121	P. Milligan, L. Price
Centering a sequence of lines	II:2	A-13	M. Díaz
Chapters and Sections	II:1	60–61, 79–81	A. Keller
	II:1	111-118	L. Price
	II:2	A-8–9, 20-22	M. Díaz
Character width determination	IV:1	38	R. McClure
Characters, macros to produce special	II:1	57, 67–70	A. Keller
Chemical notation	II:3	57–58	M. Nichols, B. Beeton
Columns			
balanced	II:3	58–59	L. Price
multiple	II:2	A-38–40	M. Díaz
	II:3	24–25	B. Beeton
	III:2	33	B. Beeton
Comparison of integral values	II:1	119–120	P. Milligan, L. Price
Counters			
automatic allocation	III:1	33	M. Plass
pseudo	II:1	60, 77	A. Keller
	II:1	120	P. Milligan, L. Price
	III:2	30	B. Beeton
Cross references	II:3	24	B. Beeton
Deferred output	II:1	60, 86–86	A. Keller
Division	II:2	47	B. McKay
Equality of integral values	II:1	119–120	P. Milligan, L. Price
\expandafter	IV:2	79	July '83 TUG meeting
Figures	II:2	A-25–27	M. Díaz
Font			
character width determination	IV:1	38	R. McClure
declaring families of a particular point size	II:1	56–57, 65–66	A. Keller
	II:2	A-11	M. Díaz
definition	II:1	119	P. Milligan, L. Price
	II:2	44–45	P. Milligan
display in table form	III:1	35	R. Beeman
Fontcodes	III:2	26	C. Jackson
Footnotes	II:1	58, 71–72	A. Keller
	II:2	A-24–25	M. Díaz
French	II:2	A-12	M. Díaz
Graphics	II:2	48–49	B. McKay
	II:3	63	TeXarcana Class
Hashsize	IV:1	36	B. Beeton
Headings, page	II:2	A-23–24	M. Díaz
Hidden Text	II:3	61	TeXarcana Class
If			
comparison of integral values	II:1	119–120	P. Milligan, L. Price
groupless \if	II:2	46	B. McKay
null string, *see* Null string			
testing math-style (display, script or scriptscript)	II:2	46	B. McKay
\ifcase	IV:2	79	July '83 TUG meeting
Index production	I:1	Appendix A	T. Winograd, W. Paxton
	II:2	A-28	M. Díaz
Justification			
of reviewer's names	II:3	62	TeXarcana Class
right ~	II:3	63	TeXarcana Class
Layout macros	IV:1	37	A. Mohr
Letters	II:2	A-32–35	M. Díaz
Letterhead	II:2	A-33	M. Díaz
Line numbering	III:1	43	TeXarcana Class
Lists	II:1	59, 72–72	A. Keller
	II:1	98–110	L. Price
	II:2	A-15	M. Díaz
Margins	II:2	A-19	M. Díaz
\mark	IV:2	79	July '83 TUG meeting

Matrices	II:2	A-30	M. Díaz
Memos	II:2	A-32–35	M. Díaz
Miscellaneous			
automatic printing of macro names	II:3	60–61	L. Price
avoiding "**Argument of** ⟨control sequence⟩ **has an extra** }."	II:2	50	M. Spivak
conditional evaluation of macros	II:2	50	M. Spivak
input-dependent macro redefinition	II:3	59–60	L. Price
\input within \if	II:2	50	M. Spivak
single tokens, identifying	II:2	52	M. Spivak
Multiplication	II:2	47	B. McKay
Nofill			
macros	II:1	59–60, 74–76	A. Keller
	II:2	A-16–18, 36	M. Díaz
program (SAIL)	II:1	87–93	L. Price, P. Milligan
program (Pascal)	II:1	94–97	L. Price, P. Milligan
program errata (SAIL and Pascal)	II:2	43–44	
Notes			
output to the writer on a separate file	II:1	60, 76, 85	A. Keller
printed at end of document	II:2	A-25	M. Díaz
Null string, testing for	II:1	60, 77	A. Keller
	II:2	51–52	M. Spivak
Numbering, page	II:1	57, 70–71	A. Keller
line	III:1	43	TeXarcana Class
Output routines	II:1	57–58, 60–62, 71, 82–85	A. Keller
	II:2	A-18, 40	M. Díaz
	III:2	33	B. Beeton
Overlining	II:2	A-13	M. Díaz
Page layout	IV:1	37	A. Mohr
Page numbering	II:1	57, 70–71	A. Keller
	II:2	A-18, 23	M. Díaz
Paragraphs			
beginning with dropped initial caps	IV:2	81	Problems column
beginning with large capital letters	II:1	60, 78	A. Keller
	II:2	A-16	M. Díaz
in tables	III:2	38	Problems column
indented	II:1	58, 72	A. Keller
	II:2	A-13–15	M. Díaz
numbered, *see* Lists			
with first line all caps	IV:2	80	Problems column
Parentheses, assorted sizes	II:2	A-11	M. Díaz
Pictures, plotting	II:2	48–49	B. McKay
Point, declaring font families of a particular ∼ size	II:1	56–57, 65–66	A. Keller
	II:2	A-11	M. Díaz
Proofs	II:2	A-31–32	M. Díaz
Punctuation, 'hanging'	III:2	38	Problems column
	IV:1	39	Problems column
Push-down stacks	III:2	34–36	L. Lamport
Recursion	II:2	46–48	B. McKay
	II:2	53	M. Spivak
References	II:2	A-25	M. Díaz
Registration marks	III:2	30	B. Beeton
Roman numerals, uppercase	II:1	120–121	P. Milligan, L. Price
Seating charts	III:1	39	R. Beeman
Slides	IV:2	78	July '83 TUG meeting
Spanish	II:2	A-12	M. Díaz
Strings			
testing for ∼ equivalence	II:3	61	L. Price
testing for the null ∼	II:1	60, 77	A. Keller
	II:2	51–51	M. Spivak
Struts	IV:1	35–36	B. Beeton
Syntax charts	II:3	39–56	M. Plass
Table of Contents	II:1	60, 62, 86	A. Keller
	II:1	111–118	L. Price
	II:2	A-27–28	M. Díaz
	II:3	24	B. Beeton
Tables	II:2	A-25–27	M. Díaz
paragraphs in ∼	III:2	38	Problems column
Testing			
integral values	II:1	119–120	P. Milligan, L. Price
math-style (display, script or scriptscript)	II:2	46	B. McKay
for string equivalence	II:3	61	L. Price
for the null string	II:1	60,77	A. Keller
	II:2	51–52	M. Spivak
Theorems	II:2	A-31–32	M. Díaz
Top, baseline set to ∼ of box	II:1	60, 77	A. Keller
Transparencies	IV:2	78	July '83 TUG meeting
TUGboat submissions	II:1	53–54	B. Beeton
	II:3	25	B. Beeton
Underlining	II:1	59, 73	A. Keller
	II:2	A-13	M. Díaz
Uppercase letters			
dropped initial	IV:2	81	Problems column
first line of paragraph	IV:2	80	Problems column
large ∼ at beginning of paragraph	II:1	60, 78	A. Keller
	II:2	A-16	M. Díaz
	IV:2	81	Problems column
Roman numerals	II:1	120–121	P. Milligan, L. Price
Verbatim			
mode	II:1	59–60, 74–76	A. Keller
	II:2	A-16–18, 36	M. Díaz
program (SAIL)	II:1	87–93	L. Price, P. Milligan
program (Pascal)	II:1	94–97	L. Price, P. Milligan
Vertical text	II:3	64	TeXarcana Class

* * * * * * * * * * *

Problems

* * * * * * * * * * *

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
527 Lakeside Drive
Sunnyvale, CA 94086

Two problems have been submitted by Jim Sterken, Textset Inc. Both deal with the special treatment often given to the first line of the first paragraph of a document.

First line of paragraph all caps

A test for *Ars Orientalis* required the following:

TWO OWL-SHAPED *TSUN*^{a} ATTRIBUTED TO EITHER the Shang or Chou dynasty were examined in the Freer Technical Laboratory to determine their materials, method of manufacture, and age. ...

The first line is in all caps, 8pt instead of 10pt. To do this, I cheated and made the first line with "`\hbox{T\eightpoint\rm WO OWL-SHAPED...}`".

My TEX quiz question is:

Design a smarter macro to do the line breaking, capitalization, and font switch automatically:

```
\beginchapter Two owl-shaped ...
```

Paragraph with dropped initial

Consider:

> Very often a fancy book
> will begin chapters with a
> dropped initial like this ...

Note that it's not V ery ... will ...

Question:

How can TEX be made to do this generally?

Editor's suggestion: Since TEX is privy only to the font metrics, which do not include a specific definition of character shape, this type of kerning doesn't seem possible without providing further information. But if one specific alphabet were always to be used, it might be possible to provide several additional values for each letter, e.g. the proportion of the letter's width at several heights chosen such that the appearance of a text letter set beginning at that point would have a pleasing appearance.

* * * * * * * * * * *

Letters et alia

* * * * * * * * * * *

OBSERVATIONS ON TEX FROM A DIVERGENT VIEWPOINT: A CRITICAL COMMENTARY

Some years ago, the American Mathematical Society began use of a computer typesetting program written by Science Typographers, Inc. (STI) for composition of many of its mathematical publications. At the time, the Society found the STI language the most effective and efficient of the computer composition languages which were intended for setting complex mathematical formulations and were available for testing.

During the years since, Jim Roesser and Roger Jones, co-founders of STI, have worked to improve the language, adding new features—many at the Society's request—and increasing its versatility. When TEX appeared on the scene, the AMS began investigating its potential as a language by which authors with access to computing systems might communicate their manuscripts to the Society in ways which could eliminate some of the costs of scientific publication. TEX showed much promise in this area, and considerable effort has been, and is being, expended to see whether this promise can be realized.

(Both systems have their strengths, however, and it is likely that the AMS will continue using both well into the future. TEX itself, in fact, is still in limited production use at the AMS for typesetting mathematical literature; the STI program continues to be relied on heavily for that. TEX is in regular use at the Society for typesetting material requiring very complex page layouts which the STI program would handle with greater difficulty. We expect that TEX82 will be put into production use for mathematical typesetting during the first quarter of 1984. Here and elsewhere, of course, TEX has been used to typeset very many mathematical documents.)

Jim and Roger have followed with interest the development of TEX at Stanford, at the Society, and elsewhere. Both have attended TUG meetings. At the meeting at Stanford in July, Jim was often critical of TEX's approach to mathematical typesetting. Thinking that the TEX community might benefit from Jim's criticisms, I asked him to write them into an article for TUGboat. Jim declined, but told me that he would write up some comments for his own staff from his notes. He promised to send us a copy, which we could use as we liked.

The following article is Jim's analysis, unedited. His criticisms are occasionally sharp, but generally are based on his very great experience as a mathematical typesetter. We publish it in the hope that, where his criticisms are well-founded, the TEX community may move the development of TEX in directions which answer them. If so, we will have benefited.

Following Jim's article are commentaries by Don Knuth, Dave Fuchs, Mike Spivak, and Richard Palais, and, lastly, by Barbara Beeton of the AMS, who, probably more than any other person, is qualified to compare the strengths and weaknesses of the TEX and STI programs. The series closes with a final statement from Jim. TUGboat will welcome constructive responses to any of these statements from its readers.

Sam Whidden

* * * * * * * * * * *

Editor's note: Camera copy for Jim Roesser's memo was prepared at Science Typographers, Inc., using the STI typesetting program. The typesetter was a Harris 7400, and fonts from the Times Roman family were used.

TO: STI Staff

FROM: J. R. Roesser

SUBJ.: TEX Users Group Meeting, July 1983

I. Introduction TEX is a mathematical typesetting program developed by Donald Knuth of Stanford. It is necessary that we understand TEX.

1. It is a competitor.
2. We may learn from it.
3. We will soon begin to receive manuscripts coded in TEX.

What follows is based on my notes taken during the meeting. I shall begin by attempting to explain the TEX philosophy, i.e., what TEX is trying to accomplish. Part III will be a description of the meeting. Part IV lists the things which TEX can do and STI cannot. Part V is a critique of TEX. In general I will attempt to see if TEX is accomplishing its goals. Part VI is a summary.

II. TEX Philosophy TEX was developed to provide cheaper and higher quality mathematical typesetting. (If we project ahead and consider electronic output perhaps we should replace "typesetting" with "communications.") This is to be accomplished by the following steps.

1. Develop a high quality math typesetting program (TEX).
2. Provide it free of charge so that it becomes the standard for the mathematical community. Thus it will be used by:
   i) The AMS
   ii) Individual authors and institutions, and
   iii) Commercial typesetters.
3. Develop fonts (METAFONT) so that TEX will drive the new laser printers and, thus, provide easily accessible, cheap output.

The proliferation of TEX would guarantee good, high quality, inexpensive communication. Further, it would make author-generated copy realistic. The AMS (see E. Swanson, *TUGboat* Vol. 1, 1) seem to believe that author-generated copy will provide great cost advantages.

III. July Meeting The first two days of the meeting were used by Michael Spivak for a short course in the use of AMS-TEX82. AMS-TEX82 is a particular version of TEX. The AMS prefix refers to a set of "macros" developed by the American Mathematical Society which simplify input.

I must digress here to explain what is meant by macros. The STI typesetting program has evolved over the past 12 years by sampling many thousands of pages of author copy. We have tried to consistently incorporate into our program a complete set of capabilities. The developers of TEX have had at least two orders of magnitude less material available for examination. The TEX program is thus quite general and, therefore, inefficient. For efficient use it requires an adjunct program (in this case AMS macros) for input. These macros are just the sort of input functions which are already in use at STI. Furthermore, because STI has developed its input functions by large scale sampling, STI's are much more complete.

The TEX people believe that their flexibility is an advantage because each author can set his paper exactly as he wants it set. As we went through the meeting I tried without success to find examples of this flexibility. The truth appears to be that TEX is imposing a serious and unnecessary limitation on its ability to communicate and that as it evolves it will be recognized that most of this "flexibility" must be removed. AMS-TEX82 is a step in this direction.

End of digression.

This two day session was interesting because the typesetting problems discussed by Spivak were all old friends. That is, problems which we had seen and solved years ago.

Now for some high points.

1. A question from the floor: "I have set the fraction $\frac{a_y}{2}$ and the 'y' is too far above the fraction bar." Answer: "Use the command `\botsmash` to make the program think that a_y has no depth. This will, unfortunately, make the y print on top of the fraction bar. So we use `\vphantom(` to give enough depth to clear."

   Need I continue? This is rubbish.

2. Much was made of the use of macros for keyboarding. Consider $\frac{\partial^2 G}{\partial x_i \partial x_j}$. One makes a macro (in TEX) with two parameters (in this case for i and j) and only keys that macro for the many occurrences of the fraction. This seems a good feature for an author. I asked Spivak if he thought there was an advantage here for a production typist. He did not really understand the question. His reply was that anyone with sense would see that less keystrokes are better than more keystrokes.

   About eleven years ago I would have agreed. In fact we were surprised that our production keyboarders had never used macros in this way. Finally we realized that such a fraction is 27 keystrokes and requires only about 10 seconds for keying. Contrast that with the time required for the keyboarder to break rhythm, make sure it is the right form (particularly if there are a number of macros), note what the parameters are, and, finally, key the macro.

Some random comments.

1. Space is used as a delimiter. If one were to list a set of axioms for a good input language the first would be: "Choose as control codes and delimiters characters which appear infrequently in ordinary copy." The use of spaces clearly comes from the programming background of the TEX developers and is the worst possible choice.
2. Goes all around the barn for double spacing after sentences. Is it worth it?
3. Question about automatic intercharacter spacing. Spivak claimed that TEX always did this properly. Roger reported from last year's meeting that it was a big problem. I checked with Joe Fineman who has seen a fair number of TEX-set papers. He said that the intercharacter spacing is poor.
4. I do not like the mnemonic input (more later).
5. Sizing of parentheses is awkward. There are four ways in TEX to key parens (which, of course, require a choice by the keyboarder). First, hitting the regular paren key will give a normal one line paren. Second, for a little larger paren key `\big(`. Third, to enclose a 2 line function key `\bigg(`. Last, for a paren which will size key `\left(`. The last case requires a match. Our automatic sizing is clearly superior.

If you are interested in more specifics on the TEX input language please see me.

The next three days were used to discuss problems and developments with TEX users. On the first morning there was a talk by D. Knuth. He promised that TEX82 (which was due for release in August 1982) would finally be ready on July 20, 1983. He said that he had gone through 20 versions of the program since October 1, 1982 but was now ready to distribute the final error-free version. When he was later asked how it would be maintained he said that he would personally fix the few problems which might arise.

This presents another flaw in thinking. The reason that TEX has taken so long to develop is that mathematical typesetting is a complex problem. To believe that no maintenance is required is terribly naïve.

The next problem for TEX is who will maintain TEX? If it is continually changed by the users, then the goal of good communication will be lost. If an organization is to maintain TEX it will be a very, very expensive proposition.

Knuth then stated that the manual (TEXBOOK) will be available in bookstores by October. Volume 2 will be on METAFONT and should be available in about 2 years. He also mentioned that TEX82 requires about one-half of a megabyte of core.

IV. TEX yes; STI no The following items are available in TEX but not in our program.

1. Our page make up program is not yet complete. The part that we are using is, however, superior to TEX's.
2. Multiline justification. This is certainly a necessary addition for us.
3. Automatic double spacing after sentences. An unnecessary complication for the keyboarding.
4. Automatic positioning of footnotes. These "SCRIBE-like" features are desirable for author input but not so important for production typesetters.
5. Fewer keystrokes for superior/inferior. Note that Computype changed the STI input for down and up arrows for subs and supers. This gives one less keystroke.
6. Macros may be redefined and may have arguments.
7. Arrows over groups of characters. This is on our list for inclusion.
8. `\vphantom` can be used on functions as well as single characters.
9. Sized fences match generally. That is, (matches], etc. This makes for simpler keying for cases of ( ]. However, in the great majority of cases it eliminates a very nice error checking ability.
10. Allows alignment of equations on other than equals signs. It must in all cases be marked.
11. It appears that TEX has a superior hyphenation package. We are now looking into obtaining this package for inclusion in the STI program.

V. Critique In order to see how well TEX is likely to meet its goals, I will first list what I consider significant problems in TEX. This is by no means a complete list. In particular I will be unable to provide a list of STI capabilities which are unavailable in TEX. This is because there is really no easy way to examine the TEX capabilities (see 7 below).

1. I consider the greatest problem to be that TEX does not have a standard simple supported input language. The idea that a desirable flexibility is achieved by allowing each author to define his own input language will prevent TEX from meeting its goals.

A math typesetting program, if it is to be of real value to the mathematics community, must be efficient for both authors and commercial typesetters. Consider first TEX's use by authors. For the occasional author it is a distinct disadvantage to have to define macros. Authors require a complete input system so that they will not waste time redoing what has already been done.

Next consider the commercial typesetter. For standard manuscript input someone would be required to scan each manuscript and define macros. For author-generated input think of the problem when every manuscript is presented with a different set of macros. This will introduce handling costs which will more than offset the cost advantages of author-generated copy. (In this case STI has had experience with a number of author tapes. I can assure you that what I say here is true.)

Finally, note that the reason for this emphasis on flexibility is that the designers of TEX are designing on the basis of theory exclusively. STI has dealt with hundreds of authors and hundreds of thousands of pages of math. We have found out experimentally that it is not only feasible but necessary to have a complete input language.

2. Another problem of similar magnitude is the lack of a system for maintenance of TEX. The two possibilities mentioned above (user maintenance or an organization for maintenance) are both unlikely. The first will cause a divergence of the program. The second can only be provided by a commercial organization. This however brings out the question of whether the TEX emphasis on being a noncommercial system is in fact an advantage.

3. Part of the reasoning behind the TEX development is faulty. Consider the following quote from an article by Richard S. Palais in *TUGboat*. "Anyone who looks at the process for producing scientific journals must be struck by the tremendous wastefulness of human time and effort it entails. After the author in collaboration with technical typists, referees and editor has at great effort and expense created a supposedly error-free typescript, the paper is sent out for composition. What happens next seems almost ludicrous. At more great effort and expense (and with all good will) the compositor introduces dozens or even hundreds of errors in the proof version of the paper. At additional effort and expense these errors are laboriously removed until the paper is at last ... finally back in the form in which it was sent to to be composed. This activity of adding errors and then removing them is actually responsible for half the cost of producing the journal."

   This is nonsense. One is tempted to ask who his compositor is. Factually he is incorrect in several instances. (1) Authors seldom, if ever, present error-free manuscripts. (2) Decent compositors seldom introduce dozens (certainly not hundreds) of errors. (3) The sum of the costs for printers' errors and authors' errors is not even close to half the cost of producing the journal.

Thus TEX is attacking a problem which it does not seem to have adequately defined. To assume that author-generated copy will solve publication problems one must approach the problem in a scientific manner. Ask the following questions. (1) Is the average author able to do a high quality job of keying his manuscript? (After handling a number of author tapes I would say no.) (2) Will mathematics journals accept the lower quality and inconsistent typesetting which will come from authors? (3) Will the savings achieved by author input balance the extra charges which will inevitably be required for receiving author input?

These questions must be answered experimentally over a period of time. To assume their answer now is naïve.

4. Next is a controversial subject. I do not like TEX's use of mnemonics for input. Mnemonics assume that the best way for a keyboarder to key a special character is to first think of the name of the character and then to truncate the name to obtain a four or five character mnemonic. This reasoning breaks down in two ways. First, a production keyboarder is unlikely to know the names of most math symbols. Secondly, when a large group of symbols is available it is difficult to guess the truncation.

I believe that a better way to access special characters is to group such characters in classes (arrows, canceled symbols, Greek, etc.). This has three advantages. First, if the symbol must be looked up a smaller set is involved. Second, fewer keystrokes are needed. Third, we found from analyzing our production keyboarders that they do not follow the sequence; see a symbol, identify it by name, obtain (by memory or look up) the code and key. The second step is left out. Therefore the use of mnemonics would lower their efficiency. If the symbol must be looked up the non-mnemonics have a further advantage that they all fit on a single page (see enclosure).

To close this subject consider that TEX uses `\binom ab` for $\binom{a}{b}$. How many production keyboarders know this is the binomial coefficient? Similar questions may be asked about most of TEX's mnemonics.

5. Intercharacter spacing is poor. This is one of the most important factors determining the quality of output. Until it is corrected TEX will never be able to provide its advertised high quality.

6. Table capabilities are primitive.

7. User manuals are only usable by programmers. In general my impression is that TEX was written by programmers for other programmers.

8. TEX does not have the auxiliary programs which help make the process of composition easier. Consider how much more difficult our task was before we had the check program, fotrun, and the ability to set patches.

I believe that the above limitations will restrict the use of TEX to those individuals who are willing to invest considerable time in training/study. It is not now nor will it become a program which can be used by occasional authors and production typesetters unless there are drastic changes in direction.

VI. Summary There is an interesting problem here. I believe that STI's typesetting program is clearly superior to TEX. By the end of the year we should be able to incorporate the few features noted in part IV above. The more important points where STI is ahead of TEX are not so easily acquired. Thus STI should be able to continue its advantage. Yet the math community (led by the AMS) continues to see TEX as the answer. I get the impression that STI is not even seriously considered. Why?

Part of the answer is the emphasis on TEX being free. When will it be realized that this is, in fact, a disadvantage. Perhaps STI should let it be clearly known that we are interested in providing our program to the math community on some mutually beneficial basis.

This, however, would probably not make much impression because of the second problem. That is, the average mathematician is quite limited in his knowledge of typesetting problems. Because of Knuth's reputation as a programmer many mathematicians are unwilling to consider any solution other than TEX. If we were considering only a programming problem perhaps this would be justified. In this case, however, the human interface with the program is of greater importance.

WHAT YOU SEE IS WHAT YOU GET

*t		*e	*f		*m		*O	*g		*q		=		*a	*v	*n	*b	*j	*p	*y	@a		
160	161	164	170	171	100	101	151	70	71	140	141			60	50	30	20	64	40	10	150		
a	A	&	a	𝒜	𝔞	𝔄	A	α	A	а	А	■	–	–	⫅	⊈	∀	∧	ℵ	&	¨	a	01
b	B	–	b	ℬ	𝔟	𝔅	B	β	B	б	Б	■	–	⇂	⊆	⊈	∃	–	ℶ	☿	˜	b	02
c	C	–	c	𝒞	𝔠	ℭ	C	χ	X	ц	Ц	■	–	⇃	⊂	⊄	§	⋐	ℂ	¢	¯	c	03
d	D	$	d	𝒟	𝔡	𝔇	D	δ	Δ	д	Д	■	–	↓	⫆	⊉	‡	▵	ℸ	đĐ	¯	d	04
e	E	–	e	ℰ	𝔢	𝔈	E	ϵ	E	е	Е	э	Э	⇓	⊇	⊉	∝	–	∍	∈	¯	e	05
f	F	–	f	ℱ	𝔣	𝔉	F	φ	Φ	ф	Ф	є	Є	⇋	⊃	⊅	⩽	⋑	≺	♀	¯	f	06
g	G	–	g	𝒢	𝔤	𝔊	G	γ	Γ	г	Г	■	–	⇌	⋙	⊁	>	⊳	≫	♃	▪	g	07
h	H	–	h	ℋ	𝔥	ℌ	H	η	H	х	Х	і	І	⇆	≖	⊀	ℏ	–	ℷ	♄	▪	h	10
i	I	–	i	ℐ	𝔦	ℑ	I	ι	I	и	И	й	Й	⇄	∝	⊁	∫	ƒ	∮	ı	·	i	11
j	J	–	j	𝒥	𝔧	𝔍	J	ψ	Ψ	я	Я	ь	Ь	↩	⊩	⊊	æÆ	⋔	∯	ȷ	·	j	12
k	K	–	k	𝒦	𝔨	𝔎	K	κ	K	к	К	■	–	↼	⊫	⊋	œŒ	–	∰	♮	·	k	13
l	L	–	l	ℒ	𝔩	𝔏	L	λ	Λ	л	Л	■	–	←	⋘	↚	<	⋖	≪	łŁ		l	14
m	M	–	m	ℳ	𝔪	𝔐	M	μ	M	м	М	■	–	⇐	⊪	⇍	∓	⋉	♭	♂	¯	m	15
n	N	#	n	𝒩	𝔫	𝔑	N	ν	N	н	Н	■	–	↔	≤	⪇	≠	⋊	♯	#	˜	n	16
o	O	0	o	𝒪	𝔬	𝔒	O	o	O	о	О	■	–	↝	≥	⪈	⩾	⋈	≻	∅	▪	o	17
p	P	£	p	𝒫	𝔭	𝔓	P	π	Π	п	П	■	–	↪	⊥	∄	∏	–	∐	℘	¯	p	20
q	Q	Ø	q	𝒬	𝔮	𝔔	Q	θ	Θ	ч	Ч	■	–	⇀	‖	∦	‖	⩔	⫴	∵		q	21
r	R	–	r	ℛ	𝔯	ℜ	R	ρ	P	р	Р	■	–	→	≜	↛	√	⩕	®	∴	.	r	22
s	S	–	s	𝒮	𝔰	𝔖	S	σ	Σ	с	С	■	–	⇒	≑	⇏	∑	⩖	■	ß	⁄	s	23
t	T	˜	t	𝒯	𝔱	𝔗	T	τ	T	т	Т	■	–	↦	≏	↚	~	⩓	⌣	⊤	/	t	24
u	U	–	u	𝒰	𝔲	𝔘	U	ω	Ω	у	У	ю	Ю	↑	≎	≉	¶	⋓	⋃	∪		u	25
v	V	|	v	𝒱	𝔳	𝔙	V	∂	∇	в	В	■	–	⇑	∣	∤	|	⩒	⋁	∨	,	v	26
w	W	–	w	𝒲	𝔴	𝔚	W	ϑ	Θ	щ	Щ	ъ	Ъ	↿	≦	≰	£	⋒	⋂	∩	˜	w	27
x	X	–	x	𝒳	𝔵	𝔛	X	ξ	Ξ	ш	Ш	■	–	↾	≧	≱	×	⋌	⊠	⊗	˝	x	30
y	Y	ˆ	y	𝒴	𝔶	𝔜	Y	υ	ϒ	ы	Ы	■	–	⇛	⩽	⋦	×	⩑	⋀	∧	¯	y	31
z	Z	@	z	𝒵	𝔷	ℨ	Z	ζ	Z	з	З	ж	Ж	⇚	⩾	⋧	@	–	⧈	⊙	ˇ	z	32
0	0	0	0	0	–	–	0	–	–	0	0	■	–	↕	≡	⊮	•	–	●	○	˚	0	33
1	1	–	1	1	–	–	1	–	–	1	1	■	–	⇕	≃	≄	–	–	◆	◇	·	1	34
2	2	–	2	2	–	–	2	ς	–	2	2	■	–	↔	≊	⊯	—	–	♥	øØ	¨	2	35
3	3	–	3	3	–	–	3	ε	–	3	3	■	–	⇔	≬	⇎	°	°	∘	♆	'	3	36
4	4	★	4	4	–	–	4	φ	–	4	4	■	–	↖	⪅	⪉	*	*	*	∠	‘	4	37
5	5	–	5	5	–	–	5	F	–	5	5	■	–	↘	⪆	⪊	†	–	♠	∡	ˆ	5	40
6	6	–	6	6	–	–	6	ϖ	–	6	6	■	–	↗	⊲	∌	‘	≺	♣	∢	˘	6	41
7	7	–	7	7	–	–	7	–	–	7	7	■	–	↙	⊳	∉	’	’	⨿	⋮	¯	7	42
8	8	–	8	8	–	–	8	χ	–	8	8	■	–	⇈	⋚	≮	″	″	⋚	⪋	¸	8	43
9	9	–	9	9	–	–	9	ϱ	–	9	9	■	–	⇊	⋛	≯	‴	‴	⋛	⪌	_	9	44
.	.	–	.	.	–	–	.	–	–	.	.	■	–	⇉	≾	⪵	:	·	▲	△		.	45
,	,	–	,	,	–	–	,	–	–	,	,	■	–	⇇	≿	⪶	;	–	▼	▽		,	46
;	;	–	;	;	–	–	;	–	–	;	;	■	–	ˇ	⪷	⋨	✠	=	◀	◁		;	47
:	:	–	:	:	–	–	:	–	–	:	:	■	–	ˆ	⪸	⋩	÷	=	▶	▷		:	50
(	(	–	(	(	–	–	–	–	–	(	(	■	–	↶	≺	⋠	[	⌣	⊔	⊻		(	51
)	)	–	)	)	–	–	–	–	–	)	)	■	–	↷	≻	⋡	]	⌢	⊓	⊼		)	52
‘	‘	\	‘	‘	–	–	‘	–	–	‘	‘	■	–	–	≼	⋢	╲	–	╲	⟨		‘	53
’	’	!	’	’	–	–	’	–	–	’	’	■	–	–	≽	⋣	╱	–	✓	⟩		’	54
-	-	_	-	-	–	–	-	–	–	-	-	■	–	–	≅	≇	–	∸	⊟	⊖		-	55
[	[	<	[	[	–	–	[	–	–	«	«	■	–	∲	⊏	⊊	└	–	⟨	«		[	56
]	]	>	]	]	–	–	]	–	–	»	»	■	–	∳	⊐	⊋	┘	–	⟩	»		]	57
?	?	–	?	?	–	–	?	–	–	?	?	■	–	↣	⊑	⊀	¿	–	–	★		?	60
!	!	–	!	!	–	–	!	–	–	!	!	■	–	↢	⊒	⋪	¡	≡	≣	≀		!	61
"	"	–										■	–	–	≍	⋫	¬	–	‱	$		"	62
{	{	–	{	{	–	–	–	–	–	э	Э	■	–	↞	⋞	⊈	⋮	⊣	⌞	⌜		{	64
}	}	–	}	}	–	–	–	–	–	і	І	■	–	↠	⋟	⊉	⋮	⊢	⌟	⌝		}	65
+	@	–	+	+	–	–	–	–	–	й	Й	■	–	–	⊢	⊬	±	+	⊞	⊕		+	67
/	g	%	/	/	–	–	–	–	–	ь	Ь	■	–	–	⊨	⊭	%	⫫	‰	∖		/	70
=	)	–	=	=	–	–	–	–	–					⤅	≍	≢	≡	=	=	≈		=	71

Comments on Jim Roesser's Memo

Don Knuth

Here are a few comments from the (admittedly biased) author of TEX.

If TEX has been based on "at least two orders of magnitude less material" than STI, then STI must have been based on many millions of pages, not just "many thousands," because TEX has been based on experience with many tens of thousands of pages, covering a very wide variety of material.

It is quite true that the old version of TEX produced bad spacing on fractions, and "a sub y over 2" is one of the many bad examples. But TEX82 does hundreds of things better than TEX80, and fractions is one of the things it does right; that formula no longer needs any corrections. On the other hand, I believe that no fully automatic system will always produce the best results, so there should be easily understandable ways to tune up formulas in the small percentage of cases where a person wants to do it.

Most of the other criticisms of TEX in Mr. Roesser's memo are quite valid criticisms of the old prototype version. But I'm confident that the new version resolves the problems; I've had valuable input from so many people, I'm willing to believe that a mature piece of software has been developed. Time will tell; I have been wrong before.

I completely agree that keystroke minimization is not extremely important to a production typist; rhythm is far more important. It's not very often that I find it better to use macros in individual formulas, except for things like accented letters. So when Mike Spivak reported to me that somebody in his course had been unable to see why his examples of macros were so great, I knew that the (unnamed) critic was a person of wide experience. Macros aren't for everybody, especially not macros that have to be made up on the fly. However, I suspect maybe one paper in four or five will involve a couple macros that will improve a production typist's speed and consistency; so I have recommended a quick glance through the paper to see if there are any obvious candidates for such simplifications.

I do believe that the new TEX will require no maintenance, but that's only because it is a general substructure on which you have to hook front ends and back ends. The front ends and back ends should, naturally, change as better ideas are found; but I see nothing terribly naïve about the utility of a stable, powerful, machine-independent, and well-checked-out "fixed point" in the middle. Indeed, the stability of the new TEX should simplify all other kinds of maintenance.

I also agree that mnemonics are not best for everyone. Again, however, that's not a TEX question; it goes into the front end. What I've tried to do is provide a solid tool for typesetting, but of course I have not resolved all the problems. I hope that when people see the new system they will find that I have solved some problems reasonably well.

My main worry right now is that too many people are still using the horrible old TEX; how can we stamp it out?

* * * * * * * * * * *

J. R. Roesser's Memo on TEX Meeting

David Fuchs

A reading of J. R. Roesser's memo indicates that he has not fully understood much of TEX. His criticisms of TEX seem to be based on misconceptions, stemming either from misunderstanding how certain features work or from misunderstanding how they effect the efficiency and usability of TEX.

For instance, consider "high point 2". What he shows is indeed the example used in the short course to motivate the idea of user-defined macros in TEX. It was not necessarily meant to be the best example in the world, nor, I'm sure, did Mike mean to imply that a production typist in any way "must" use a macro to produce such a fraction. Indeed, as Mr. Roesser points out, an experienced keyboarder might well decide to enter such fractions without the aid of any macros. Nothing in the TEX system forces the user either way in this regard.

Some of the other comments are a bit mysterious. For instance, "random comment" number 2 complains (I believe) that TEX automatically handles the extra space that should appear at the end of a sentence. There is nothing to complain about here, since 1) it doesn't make the program any less efficient, 2) it makes TEX a little easier to use, and 3) if you don't like this feature, you can easily turn it off any or all of the time. This feature is also addressed in section IV, point 3, where it says "an unnecessary complication for the keyboarding." This incorrectly implies that the TEX user has to do something special at the end of the sentence; in fact, the extra spacing is automatic (as is extra spacing after commas, semicolons, etc.).

"High point 1" is presented out of context and is thus misleading. Clearly, different people have different ideas about math composition. I doubt that anyone claims that any system can please all the people all of the time. I suspect I'd get general agreement that no system can even please a single person all of the time. That's why it is important to provide escape mechanisms that can be used to adjust things to the user's taste.

TEX also adjusts to the user's taste in mnemonic versus non-mnemonic input. It is hard to understand Mr. Roesser's remarks regarding whether mnemonic input is any better or worse than non-mnemonic. Although one widely used TEX format defines mnemonic control sequences such as `\alpha`, TEX is flexible enough to also allow the user to say `@12` or `>AB` or anything else that the user would prefer. Moreover, this capability can be used with no performance penalty. While in a production environment, short codes may be preferable, can you imagine telling mathematicians, engineers or scientists that they must say '`*gq`' instead of '`\theta`', '`*n7`' instead of '`\notin`'? TEX users can do either, or both!

On the other hand, any large system will always have some of its own lingo for experienced users, whether it looks like '`\botsmash`' or '`*XY`'. In the particular instance referred to in "high point 1", it turned out that `\vphantom` and `\botsmash` had recently been discussed, and thus Mike was answering the question in a reasonable way. If a user doesn't like the spacing around a fraction bar, there are a number of other ways to deal with it, such as with `\lower` and `\raise`.

I've never heard any complaints about TEX's "intercharacter spacing". Perhaps Mr. Roesser was looking at some old output, or output produced on a low-resolution printer, or output from a buggy device driver. Perhaps not, in which case I'd like to hear some more specifics so that I can comment intelligently. (An example of a case that shows poor "intercharacter spacing" would help.) In all the feedback we've received from the folks at AMS who have been involved with TEX since the beginning, they have reported no such problems.

Even if there were something wrong, TEX's "intercharacter spacing" can't be flawed beyond all hope, since the whole system is table-driven with regard to all spacing; simply correcting the tables should fix any problem. In fact, this flexibility allows TEX to use any font for any device from any manufacturer, provided that the user is able to give TEX the character width information that any front-end system requires to do line breaking. A number of installations are using TEX quite happily with non-Metafont-generated fonts in text. Any problems with spacing in these cases is strictly due to deficiencies in the design of the typeface and the side-bearing values provided by the manufacturer. The problem of using different fonts in math material is admittedly more difficult because more than just the width of each character must be supplied; but again it is not insurmountable.

About Mr. Roesser's comments concerning how large a sample of material has been evaluated for use with TEX, I submit that the AMS folks have seen more in the way of unusual copy than everyone else put together, and I'm not aware of any complaints they have about anything that TEX simply won't do. Also, I'll wager that there are currently more TEX users in the world than STI users, and we haven't heard anything along these lines from any of them, either.

Authors using TEX, Scribe, and other systems have successfully prepared their own copy. I will not comment on whether or not the compositor introduces significant numbers of errors, since the AMS knows more about this than I. It is important to realize, however, that just because an author has provided computer-readable input, does not mean that the AMS is bound to use it verbatim if it does not meet their stringent standards. Obviously, the AMS should expect authors to use the AMS-TEX package correctly, in which case the number of corrections that must be done will be no more than had the author not provided computer-readable input. If the author's tape is so bad that less work would be required to re-input it than to correct it, then once again, the AMS is no worse off than it was before. I must point out, contrary to Mr. Roesser's experience, that we have run off thousands of pages of phototypeset TEX output prepared by various outside authors, and we are quite pleased with most of the results.

It is important to realize that authors need not all go off doing things in non-standard ways using their own macros. A large number of papers can use plain TEX as-is, without any author-written macros whatsoever. Most of the rest can rely on the AMS-TEX package to fill in the remaining gaps. Only in unusual cases should the author need to worry about doing any macro writing at all; and even then probably a few macros would be plenty for any paper. The user need never even know the distinction between the facilities provided by TEX and those handled by AMS-TEX; indeed, much of what is documented as being part of "plain TEX" in the user's manual is actually done with a standard set of macros that are considered part of the basic system.

Because TEX was designed from the beginning to be a host for such packages, it is easy to see that AMS-TEX is not an "adjunct program", but an integral part of one of the many applications for which TEX can be used. TEX macro packages are the basis of the systems's great flexibility. TEX is just like Fortran in that it is a compiler that can be used to create programs that are useful to many people. No manufacturer of large mainframes would presume to sell a computer with a bunch of programs bundled in, and claim that those programs should be good enough for all possible uses that the purchaser could possibly want. Rather, manufacturers take pains to produce efficient compilers for languages like Fortran, Cobol, Pascal, etc., so that the users can do their own things when necessary. TEX macro packages are typographic in nature, but just like any other computer language, there are simple programs that can be written by novices, and more complex programs that require more experience to write, while ordinary folks simply use the programs created by others.

When authors are presented with a system tailored to their needs (such as AMS-TEX for mathematicians and LaTEX and Fácil for Scribe-like applications) they actually avoid doing anything incompatible so that they can use the handy features of those systems. Our experience with the old version of TEX is that most of our local users used the Fácil package because it was tailored to the sort of document they were producing. The only users who wrote large numbers of their own macros were those who required a much different document design than is provided by any of the existing packages. For the new TEX, the LaTEX package promises to be the most popular for non-math applications.

Mr. Roesser's "digression" implies that a good typesetting system has all of its features hardwired in. If STI wants a new feature, they put it in the code. If anyone else needs a new feature, they can try to talk the STI poeple into putting it in for them, with the risk that they'll be told that since it hasn't been needed in the last 12 years, it must be a bad idea. If STI should go under, then everyone is out of luck. Mr. Roesser must not have been paying much attention to Leslie Lamport's talk, if he couldn't find any examples of good uses of TEX's flexibility. I also can't imagine what he's thinking when he says that the flexibility lowers efficiency. The fallacy of his position is easily seen in his statement "We have tried to consistently incorporate into our program a complete set of capabilities." Even twelve years is not enough for this never-ending task. No program can be all things to all people. Only a flexible program with a TEX-like approach can possibly survive in anything more than a small niche.

The text editor EMACS, in use at MIT, Stanford, AMS, and many other locations, is an example of a large software system built from a primitive but powerful base. (EMACS sits on top of TECO, and its name is a short form of "TECO Macros".) The unsurpassed flexibility of this system can be seen from the fact that it can handle new applications that weren't even conceived of at the time the system was designed. For instance, there now is a "TEX-mode" available in EMACS that makes entry of TEX material easier through automatic matching of braces, etc. The flexibility of being able to layer macro packages on top of TEX is one of its strongest features. Not only is this not inefficient (indeed, it even saves memory space), it conforms with the current notions of good programming practice generally accepted by the computer community: Modular, layered software, table driven wherever possible. Because TEX can be built upon, features can be added without changing the underlying code. For instance, the basic table formatting capabilities of TEX are indeed "primitive", but they are also very powerful. They are designed to provide a groundwork on which virtually any possible table can be built.

Using the layered approach for TEX has many advantages, including: 1) Users who need a new feature will not be (rightfully) frightened away by the prospect of having to alter a large, existing program. 2) Adding a feature does not mean becoming incompatible with other installations (the TEX program remains the same, other people's TEX files will still work no matter what any user does). 3) Because TEX itself need not being modified to add new features, it is more stable and is much less likely to have any new bugs introduced into it. (Any programmer will tell you that adding a feature to a large program is fairly likely to add new bugs, especially compared to the likelihood of finding a bug in a program that hasn't been changed in months and has no known bugs.) The very flexibility that Mr. Roesser criticizes allows sophisticated users to tailor the TEX system to their own needs without in any way impacting on compatibility and maintainability!

Mr. Roesser has some unfounded fears about the maintainability of TEX. It is quite clear from prior experience that any bugs in TEX will be fixed by Prof. Knuth. It is in his own interest to do so, since he uses the system for his own books. The original SAIL version of TEX had its share of bugs over the years, and all of those were fixed here at Stanford. I see no reason why the same will not hold true for the new TEX. In fact, there are no known bugs in the SAIL version of TEX, and that's after a period of only a few years of its being in general use. Since the new TEX has a very similar program structure to the old, there is good reason to believe that it will take even less time to reach a mature level. It is already in active use at a number of installations, and the rate of bug reports is very low. To date, everyone who has found any bug has reported it to the TEX project, and there have been no problems due to incompatibilities from disparate bug fixes.

Well, what if Prof. Knuth is hit by a train? Bugs in TEX will still not be a problem, because of the unprecedented internal system documentation. All of our source code is available to interested parties, and in fact some of the bug reports we have received have been of the form "in module vvv line www, you should change xxx to yyy, to avoid a bug when zzz". Of course, the average user should never have to look at the program itself, but it is quite a relief to know that it's there if needed. In fact, a number of large companies are using TEX in-house and TEX82 can't be available soon enough for them. They are not worried about support because they have looked at the code of TEX82 and are confident that it is maintainable, flexible, and efficient. Initially they became interested in TEX because of Prof. Knuth's reputation in the computer community, and they are happy with our track record to date for fixing bugs, but they also know that their own programmers are quite capable of delving into the system in a pinch.

At least four companies are coming out with TEX-based products, including such large firms as Hewlett-Packard and such up-and-coming concerns as Intergraph. They know they can support a program like TEX on their own, or they would not be spending the large amount of money necessary to bring these products to market. Right now, anyone who wants traditional support for TEX can buy an HP9836 computer with the TEX package, and they'll get it from HP. Note that TEX82 isn't even officially released yet; I expect more companies to pick it up as a product in the future.

Because TEX is in the public domain, anyone who wishes to write an output device driver is free to do so. For instance, Imagen, Symbolics, QMS, IBM, Xerox, etc., have produced TEX-compatible laser printers without having to make any kind of deal with anyone. And if they hadn't, there's nothing to keep any of their users from doing so. Thus, TEX users need not fret as to what will happen if the company that supplies their composition software fails to support a device they'd like to have. Even if California should fall into the sea, TEX users know that they'll be able to use next year's new technology output peripherals (some non-Stanford TEX installations already are able to see their TEX output on high-resolution terminal screens). Likewise, they know that they'll be able to use TEX on the next generation of computer hardware, and that they'll be able to transfer their documents to any other machine, be it IBM mainframe, DEC VAX or 20, PRIME, Data General, Cray 1, etc. Even the new breed of personal computer is able to run TEX. TEX is portable between computers, operating systems, and output devices.

TEX is being used by hundreds of "occasional authors". It will soon be in production use typesetting more pages each year than STI. Already the LATEX package makes TEX close enough to Scribe that it will capture many potential Scribe users who can't afford that system. This also represents a much bigger market than STI can reach. The TROFF typesetting language on Bell's UNIX system also represents a large number of typeset pages annually (both "occasional author" and "production typesetter"), and TEX will be making large inroads in this market, too, because of its higher quality output and increased flexibility. Even if TEX didn't exist today, the success of both TROFF and Scribe contradicts Mr. Roesser's skepticism about about author-prepared documents.

Soon, virtually every department in every university and corporation, and even many homes, will have a computer that can quickly and effectively run TEX. With the coming boom in computer networking, nationally accessible data bases, and the growing capabilities of inexpensive computers with hi-res graphic screens, TEX is in a unique position to become a de-facto standard for document formatting. Because of its capability, high performance, portability, reliability, high quality, superb documentation, and low cost, I believe it will.

* * * * * * * * * * *

Remarks on the STI report

Michael Spivak

Here are some remarks on the STI report.

Much is made in the report of the dichotomy between author produced manuscripts and commercially produced ones. It seems to me that this misses an important point: it ignores the people who really do all the work!

Mathematicians simply produce *handwritten* manuscripts, and these manuscripts certainly aren't going to be sent off to a journal!—whether that journal sets type by linotype or by computer. Instead, the manuscript first has to be given to a technical typist, who goes through an elaborate ritual with a Selectric—turning the platen up and down by a half space to get super and sub scripts (after all these years Selectric typewriters don't even have special keys to do this!), popping special type-balls in for the math symbols, lining up equations manually, etc. And what happens to this laboriously produced typewritten manuscript? It goes to the computer typesetter, who proceeds to undo all the work that the typist has done—putting in special control codes to indicate that the type is set as a super or sub script, other control codes for special symbols, still other control codes to align equations, etc! Thus all the labor is done twice—actually much more than twice, since it takes much longer to perform this dance on the Selectric than it does to produce a computer file for TEX (or presumably STI, for that matter). Obviously the typist might as well produce the end product in the first place! I suspect that within 5 years 99% of mathematics papers will be produced this way.

Although the question of "a sub y over 2" is now moot (I certainly flubbed it with my off-the-top-of-my-head answer, though I also mentioned later that this may have improved in the new TEX), there is still one important point to be made about a complicated answer like the one I gave: If such a construction actually is needed more than once a year, then one would simply make a macro to do it once and for all (AMS-TEX's `\frac` could have been permanently changed if there had been a problem).

Perhaps in a commercial production setting it is more efficent to maintain the rhythm than to minimize keystrokes, but I suspect that this isn't true in the sort of environment to be found in a mathematics department, where typists are doing many different jobs, like typing business letters and other daily work in alternation with mathematics papers, and where the author can indicate the oft-recurring symbols when he gives the typist the paper. For my own part I have often found that using macros saves me a great deal of time. Even more important, however, the use of macros has often made the typing a lot more *pleasant*. I might add that I found macros a lot less appealing when I used `troff`, which has macros, but not macros with arguments.

On spaces as delimiters: I think that it is certainly pleasant to be able to have multi-symbol control sequence names for things like `\align`, `\matrix`, etc. No one is going to mind that they take several key strokes or that a space is needed after them, because what comes next—the actual alignment, or matrix, etc.—is such a chore that the typing is nothing but a welcome breather. But I also agree that there are certain situations where the type of coding scheme used by STI might be appealing. I certainly wouldn't want to use the actual coding scheme used by STI, but, to take a reasonable example, let's consider Greek letters. Obviously no one who has to key a lot of Greek symbols is going to stick with prolix control sequences like `\alpha`, `\beta`, ... One solution is to define `\a`, say, to be `\alpha`, `\b` to be `\beta`, etc. But this uses up almost all the convenient single-letter control sequences in one fell swoop, and besides, lots of people would like something like \g to stand for "Greek", with `\g a` standing for `\alpha`, `\g b` standing for `\beta`, etc. Similarly `\b a` might give a bold **a**, etc. The Greek control sequence \g could not be defined by the casual user (and the correspondence is to some extent arbitrary, though perhaps there is an industry standard), but that's hardly a problem—any TEXnician could handle the job, and it only has to be done once. The real problem is that here the spaces truly are a nuisance—it would certainly be much nicer if one could type `\ga`, `\gb`, etc. (without having to worry about the spaces after the `a` and `b`!). True, if we used something like `\1` instead of `\g`, then `\1a`, `\1b`, ... would be perfectly acceptable, but `\1` is horribly non-mnemonic. I have to admit to wishing now and then that there could be control sequences of the second kind, not requiring any delimiting spaces, but with a letter instead of a non-letter.

Although it's usually too much to expect to have one's cake and eat it too, the fact of the matter is that TEX easily allows us to add, in addition to the usual control sequences, a whole new class of codes that works exactly like this, using some other character besides \ as the escape character. **@** is an obvious choice in AMS-TEX because **\@** is used for a printed @, while **@** by itself actually has no special function—it was reserved for some future extension to AMS-TEX. Here's how it would work.

In addition to control sequences, AMS-TEX could have "@-codes". These would be pairs **@a**, **@b**, **@+**, etc., consisting of **@** followed by a single character, which could be either a letter or a non-letter. Each such @-code would work exactly like a control sequence (possibly with arguments), except that no delimiting space would be needed *even when the character after the* **@** *is a letter*; thus, every @-code would be like a control sequence of the second kind. Such an @-code would be defined with `\atdefine`. For example, if you said

```
\atdefine@b#1{{\bf#1}}
```

then @bb would expand to {\bf b}, so

```
$$@bx↑a+@by↑b$$
```

would yield

$$\mathbf{x}^a+\mathbf{y}^b$$

The @-code @b would be entirely independent of the control sequence \b (so you'd also effectively get double the number of one letter control sequences). If sufficiently many people are interested in such a feature it will be added to AMS-TeX.

[For people familiar with TeX itself, the necessary definitions are very simple. @ is already an active character in AMS-TeX (at the moment all this control sequence does is to issue an error message saying that \@ should be used for a printed @). We simply have to change the definition to

```
\def@#1{\csname at#1\endcsname}
```

Thus @a expands to \ata; this has already been made into a token, so there is no need to have a delimiting space after it. The definition of \atdefine is simply

```
\def\atdefine@#1{\expandafter
          \def\csname at#1\endcsname}
```

These are naïve versions, of course. For example, if @a hasn't been defined, the use of @a will give a TeX error message saying that \ata is undefined, instead of an AMS-TeX error message that @a isn't defined.]

There aren't four ways to key parens in AMS-TeX, there are six! (I didn't even bother to mention \Big and \Bigg in the lecture.) Actually, of course, there is only one for most things, \left(...\right), or *any other* choice of delimiters. (I like to introduce the macros \(and \) for \left(and \right) and similarly for \[and \]—this saves lots of time.) I guess I didn't emphasize sufficiently that the other non-automatic sizing macros are used only in those special cases where automatic sizing doesn't work. Specifically, \big is for things like

$$\bigl(x-f(x)\bigr)\bigl(x+f(y)\bigr)$$

which almost always appear in journals in the less pleasing form

$$(x-f(x))(x+f(y)),$$

and \bigg is for things like

$$\biggl(\sum_{i=1}^{n} a_i\biggr)$$

which looks better than

$$\left(\sum_{i=1}^{n} a_i\right).$$

Macro packages are probably the only sort of "maintenance" that has any meaning for TeX, beyond bug fixes. Of course, it is too early to tell how this will work out, but I'm not at all sure that a money-making organization is going to do any better than the motley crew responsible so far for providing the TeX community with useful macros. Within a few weeks after the meeting AMS-TeX was available fully commented. There are still four major projects envisioned for AMS-TeX: interfacing with LaTeX, macros for commutative diagrams, macros for tables, and macros for special kinds of matrices (block matrices, divided into pieces with rules or dashed or dotted lines, etc.). I don't expect to have all of this done for another six months to a year, though of course some one else may just do it sooner out of impatience. How long would it take a commercial organization to do this?

LaTeX is already available, and since SCRIBE alone costs $????, one wonders just how much this macro package would cost if it were sold by a commercial organization. And how long it would take to produce!

"The idea that a desirable flexibility is achieved by allowing each author to define his own input language will prevent TeX from meeting its goals. ... " I really don't understand this at all! Presumably many different groups of people will be using widely varying macro packages for different sorts of technical work. But it is certainly to be hoped that mathematicians will be using some standard package (AMS-TeX or something better). Surely no one expects the casual TeX user to define a fancy macro like \align. The standard package had better have all the macros of this sort that any one will need, in other words, it had better be a "complete input system". But it will surely be a great savings if authors define simple macros to deal with their special complicated oft-recurring symbols, and I can't see how the inclusion of such definitions in a file is going to lead to any sort of incompatibility.

"Table capabilities are primitive". I would say rather that TeX's *primitive* table capabilities are enormous, it's the present macro capability that is primitive (non-existent in AMS-TeX). We'll just have to wait and see.

"User manuals are only usable by programmers." In my prejudiced opinion "Joy of TeX" isn't that bad!

* * * * * * * * * * *

A Response to Mr. Roesser's Memo to the STI Staff

Richard S. Palais

Personally I feel Jim Roesser should be warmly thanked by the TEX community for sharing his criticisms with us. He makes some interesting and valid points, and the debate and self-evaluation his remarks will elicit can only be advantageous for the healthy development of the TEX system.

Unfortunately, Mr. Roesser's memo also contains a great many misunderstandings of the nature and the goals of TEX, as well as some factual errors. I had started the difficult job of writing a point by point answer to the memo when I received a copy of the brilliant response written by David Fuchs. He has said nearly everything I had hoped to, and has said it better and with more authority than I could have, so I am left with the much easier task of responding only to point 3 of Section V of the STI memo, which attacks some of my remarks in the "Message from the Chairman" in the first issue of TUGboat.

The paragraph Mr. Roesser objects to is one in which I was commenting on the incredible wastefulness involved in the keyboarding of the typescript of a scientific paper (essentially the same point that Mike Spivak makes in the second paragraph of his reply to Mr. Roesser). Let me be more specific about this process. After papers are received by the editorial department of a journal, the following expense-producing steps intervene before the issue they constitute gets into the subscriber's hands: logging in, copyediting, keyboarding, debugging input file, production of galleys, initial proofreading of galleys, mailing galleys to and from the author for his proofreading, entering keyboarding corrections, production of galleys, checking of corrections, page makeup, issue makeup, [at this point the final "camera copy" of the journal issue finally exists], printing, binding, mailing. It was the first twelve steps, producing the camera copy, that I referred to (with humorous intent) as "This activity of adding errors and then removing them" and which I claimed was responsible for half of the subscription costs of a typical journal. Of course I was *not* literally referring to the costs involved in actually making the few incorrect key-strokes and then correcting them. The point is that even if a particular paper is keyboarded with *no* errors one must go through this whole costly sequence of steps anyway. This should have been clear from my next sentence after the segment Mr. Roesser quoted, "That is, if authors could present the journal with acceptable camera-ready copy, the cost of journals could literally be cut in half!" If Mr. Roesser believes this is nonsense let me assure him it is not. These figures were not pulled out of the air! As chairman of the AMS Committee on Composition Technology, I wrote a report for my fellow members of the Board of Trustees, explaining the reasons I felt the AMS should welcome and support the development of TEX. As part of the research for this report I asked the AMS journal production staff to do a careful cost accounting of the various steps in the production of a journal. The results were quite interesting. For a journal which publishes approximately 2,000 pages per year, the cost of getting the journal into the subscriber's hands was at that time very nearly seventy-five dollars per page (i.e., a total of $150,000). Of this amount the first twelve items in the above list accounted for almost exactly half, while printing, binding, and mailing accounted for the other half. The reader who would like to see a little more detail concerning these costs should read Ellen Swanson's article "Publishing & TEX", which followed mine in TUGboat Vol. 1, No. 1, and which is cited in Section II of Jim Roesser's critique.

By the way, I asked Ellen Swanson to make another survey for me to see if, as Mr. Roesser suggested, I had exaggerated the number of errors introduced by re-keyboarding. In a sample of eighteen papers keyboarded using STI by experienced AMS keyboarders the results were as follows. Two papers contained systematic errors and contained respectively 20 errors per page and 9 errors per page. The other sixteen papers averaged only 2.2 errors per page. However since an average mathematics paper tends to run from ten to thirty pages, my estimate of "dozens or even hundreds" of keyboarding errors per paper is one I am still comfortable with.

I feel there is more to be said about the relative strengths and weakness of TEX and STI and I would like to conclude with a few general remarks in this direction. STI was designed with a specific purpose in mind, to serve as the software system to support the production shop of a service bureau that does computer composition of technical text. Since the other elements of the production systems were pretty much known it was possible to optimize the program by hard wiring into it many details, such as the specific output device that would be used. Similarly, the input language could be optimized for highly trained keyboarders working on a specific type of terminal. Granting the efficiencies inherent in such an approach, one should also be aware of the built-in deficiencies. Such an approach requires a constant maintenance battle to avoid creeping obsolescence, it is very ill-adapted to other classes of users, particularly casual and intermittent ones, and it is virtually useless to attempt to adapt it to other purposes of mathematical communication besides typesetting. TEX on the other hand has hard wired into it only a carefully designed skeleton of powerful logical primitives for the linear description of the essentially two dimensional text of mathematics. In addition, it has a wonderfully supple macro language which permits TEXnicians to put muscle, flesh, and nerves on this skeleton in many very different ways so as to optimize TEX either for their own personal idiosyncracies or to benefit various classes of potential users. The genius of this approach, which probably was not fully evident even to Professor Knuth initially, is that it provides a safe answer to the nagging question of how to permit users to enhance, develop, and maintain TEX without introducing inconsistencies and incompatibilities. As long as the maintenance is done at the macro level, the basic TEX coding remains sacrosanct and anybody's TEX input files accompanied by their macro packages will run on everybody elses configuration. This I believe is the answer to the worries expressed by Mr. Roesser at the end of his Section III about "who will maintain TEX". As one example of what would be possible using the macro language, one could write an STI-TEX package which would accept STI input files and output equivalent TEX input files.

This brings me to my final point. While STI is certainly justified in developing the most efficient program for its purpose, the AMS as one of the primary organizations representing research mathematicians has a duty to consider the long range demands of mathematical communication in a broad context and extended time frame. In the next decade many of us see two important and related roles that probably only TEX can play well. First, as the (MC68000 based) second generation of personal computers starts making the current address space limited machines obsolete, we expect to see TEX in more and more mathematicians' home computers. Secondly, as these mathematicians become familiar with some successor version of the AMS-TEX input language, it will become the de facto standard for the linearization of mathematical text and will provide the natural answer to a serious and vexing problem — how does one store mathematical text, with its two dimensional structure, in the huge ASCII data bases that are beginning to play such an important role in the storage of scientific information?

* * * * * * * * * * * *

Comments on Jim Roesser's Memo

Barbara Beeton

Jim Roesser's memo raises several points which represent legitimate and long-standing differences of opinion among persons engaged in computer-aided composition. The Science Typographers (STI) program has been in use at AMS since 1975, and it has competently prepared copy for numerous Society publications, including the most recent *Mathematical Reviews* cumulative index, a substantial document of nearly 9,000 pages. STI has responded frequently to AMS requests for enhancements, and is in the process of installing pagination capability as a result of such a request. This having been said, I wish to state that I have not yet seen a "perfect" composition input language; neither the STI language nor TEX fulfills this dream, but each has substantial strengths in the particular areas for which it was designed.

There are two areas in which I believe TEX is exceptionally strong and flexible. The first is its ability to provide a framework for data, identifying various parts of it logically, in such a way that it can be used over and over again, arranged in different ways and presented in different physical forms *with no change needed in the data itself.* This is based on the capability of defining complex macros external to the data file, and does presume that an expert macro writer is involved in the design and execution of a project. With practice and discipline, it becomes quite easy to design a multi-use file that can not only be typeset for publication, but can also be used as a permanent, screen-oriented file that can easily be maintained and used for reference between published editions.

This leads to the second area of strength — the fact that a TEX file, if well-designed and constructed, is quite comprehensible to someone with professional interest in the content of the file, but little or no background in typesetting. Minimal reference is needed to tables of symbols if input codes are intelligently assigned. It is undoubtedly true that a production keyboarder can input compact, non-redundant codes more rapidly than mnemonics. However, the fact that a mnemonically-encoded document can be read directly in non-typeset form by a wider audience than one coded according to an STI-like scheme is likely to prove important in the context of "communications", or electronic output. A full description of the TEX language is (or will shortly be) available on your bookstore shelf; I am not aware of a publicly available description of the STI coding system that would permit a random user to decipher, say, the file used to generate the next *Mathematical Reviews* annual index. This is not far-fetched — *Mathematical Reviews* bibliographic and review text data are now available on-line from several commercial retrieval services; to make these data even moderately intelligible in that medium they are laboriously translated from STI-coded files into barely-adequate mnemonic form, a process that would be quite unnecessary were the input to be created initially in TEX. Perhaps this point will become moot when graphics terminals are generally available at very modest cost; but that will not happen tomorrow, and TEX is available now.

My view is undoubtedly biased by the type of material that the AMS has been using TEX for most heavily — administrative publications (such as our membership list and Catalog of Publications), journals with very tight publication schedules, internal documents such as bibliographic file cards and forms for use in the Mathematical Reviews office — all kinds of material that contains enough special symbols that typewriters can't easily provide, but certainly not just the technical books and papers for which TEX was originally designed. The Society adopted TEX originally not only for its ability to handle mathematical composition, but also because (unlike the STI program) it could cope with multiple columns and other complex page layouts. For these uses, it (still TEX80) has proved an effective production tool. There is no reason to believe that TEX82 will be less effective.

One point neglected by Jim Roesser seems to me the weakest feature of TEX — the present lack of support for high-quality output by the manufacturers of typesetting equipment. Although the situation is improving, no typesetter manufacturer has thus far offered wholehearted support of the TEX concept of low-resolution proof copy that is fully compatible, without an additional composition run, with publication-quality final output. This, more than any other factor, will probably influence the decision of commercial publishers to use TEX or not.

* * * * * * * * * * *

A Reply to the Replies

by J. R. Roesser

To Donald Knuth

The STI program has typeset well over a million pages of math. All of these pages were set at high quality with competitive schedules and costs. If we consider only math pages (which is really the subject of my memo) then I believe that my "two orders of magnitude" is not totally unreasonable.

To Michael Spivak

1) Your prediction that "within 5 years 99% of mathematics papers" will be author generated is probably wrong. It is certainly true that each manuscript is keyed twice. However, the cost of keying is $2–2.50 per finished page (including corrections). Thus an author of a 20 page paper can save only $50 by providing machine readable copy. (This assumes that proofreading, etc. are still required—a reasonable assumption if quality is to be maintained.)

   In order to save this $50 the author must learn to properly key the manuscript (or have access to someone who can) and provide it in some machine readable format. The publisher must be able to read and process the material. If problems occur then someone with a higher salary than a keyboarder must solve them.

   Clearly author input will increase. Not, however, to 99%. (Note: Since this was written I have read the reply from R. S. Palais. Please see his discussion of author-prepared copy and my reply.)

2) You say that only the macro packages must be maintained. Quite true. That is the problem.

To David Fuchs

1) The purpose of my memo was to inform the STI staff about TeX. The fact that so much time was spent by Mr. Spivak on that particular discussion of macros is an indication of the level reached in the TeX development.

2) The extra space at the end of sentences is not in general use by publishers. It complicates things because when in use one must remember to key special spaces after initials and abbreviations.

3) If you are interested I will send you a copy of the test which we use for intercharacter spacing. If TeX gives equally good results I will withdraw the objection.

   Your statements about a lack of adverse comment from AMS are easily explained when one realizes that their math production is on the STI program.

4) "... can you imagine telling a mathematician or ... to use `*gq` instead of `\theta`; `*n7` instead of `\notin`?" Yes. As a physicist I find it simple and logically complete. The asterisk in all cases indicates a change of font. The `g` tells that the new font is Greek (or the `n` (for not) indicates a "canceled" symbol font). Finally the `q` or `7` refers to a specific character. If I forget the code I can look at a single sheet where all symbols are grouped in a logical way.

   Thus we have a coding system which is equally usable by typist or scientist. To see the problem with mnemonics look at the six-page list from the AMS.

   If the technical societies and publishers would jointly decide on an input language for phototypesetting, STI would gladly follow along. I believe that good communication depends on a common language (or perhaps a small number of intertranslatable languages). Remember that files must be edited and prepared for data bases.

   In my opinion the impediment to good communication is the idea that each author should be free to design his own input language. A good language (macro package) requires time and control to provide efficiency and completeness. (Obviously it will never be totally complete.) The beauty of the STI approach is that our language is now complete enough that we are able to satisfy virtually all of the authors with which we deal.

5) You refer to "features deep down in the STI program". I am sorry if I gave you that impression. (Again, the staff at STI for whom the memo was written know that we can alter the program at will to satisfy authors—though only when we have the publisher's approval.) The difference in practice between changing TeX macros and changing "features deep in the STI program" is mostly one of semantics.

In both systems there is a core of processing logic (TEX, for example) and a layer (macros) which is altered as new features are added. This is just as true for STI as for TEX even though we have not formally separated our program. I submit that it is just as easy for us to add new features as it is for "macro wizards" to add new macros. We have chosen not to make this available to everyone because it would then be impossible to maintain a common input language. Instead we try to incorporate all reasonable sugestions in a consistent way.

The flexibility I object to is the uncontrolled ability of every author to change input language or style at will. Changes in input language make communication difficult and data bases impossible. Random changes in style lowers journal quality.

6) In your reference to author-prepared copy you make the point that AMS could decide in each case to use or not to use such copy. The very act of making that decision costs time (money).

7) As to your assumption that TEX will take over the world. I wish that I were still new enough in this game to think that way.

To Barbara Beeton

1) Please note that STI is not only capable of reusing data in different formats, but we do so on a regular basis.

2) Using data bases to drive CRTs for math is an unresolved problem. However, I submit that STI is better positioned than TEX. If necessary it is quite simple to construct a table to convert from STI input codes to mnemonics (*if* that is what is required). One then has the double advantage of efficient keying and whatever one wishes to see on the screen.

   TEX, on the other hand, will require a separate translator for each variation of input language. (See where D. Fuchs explains that one could define alpha in many different ways.)

3) Granted STI does not yet have full page makeup capabilities. We are making good progress in that direction.

To Richard S. Palais

1) Of the 12 steps that constitute half the cost, only one (the initial keyboarding) is eliminated by the use of author-generated copy *as such*. Although no doubt money and time could be saved by not logging in submitted articles, not copyediting the file, not proofreading output, etc., the result would be a serious decline in quality. If this compromise is indeed acceptable, it can be achieved without

new machinery, merely by ceasing to worry about lost copy, spelling, grammar, mathematical style, etc.

2) To achieve a 50% decrease in costs would require that 100% of authors supply camera copy. Very unlikely.

3) As to the number of errors per page introduced by the typesetter, STI follows accepted industry standards, which are no more than two errors per galley. Since a galley is about one and a half pages, this means about one and a third errors per page. Conversely we as a typesetter correct more than that number of author errors. Note that at the present time STI is regularly setting about 48 journals. We would not have this work if our error rate were much higher. Also, AMS is using the STI system without our error-checking program.

4) Almost without exception your comments about the STI system are grossly incorrect. The ideas behind the STI system were developed between 1969 and 1971 by Roger, Joe Fineman, and me while I was an editor of *The Physical Review*. The purpose was to provide a typesetting system which could (a) efficiently typeset the journal, (b) provide a means by which authors could generate their own camera-ready copy, and (c) produce files for data bases. At that time we already were using all of the arguments about double keying and how the APS should reduce page charges for all papers which authors ran through the anticipated typesetting system. I can show you a 1970 application for NSF funds where we argue (a) that all software should be hardware independent because of anticipated improvements in hardware and (b) data files must be compatible with data bases.

   The STI program has been developed with total consistency to these ideas. Unfortunately we were so far ahead of our customers that it has been necessary to wait for the opportunity to put some of these ideas into practice. Thus STI has long been able to do the things that you say we cannot. It is sad that there has not been greater understanding of our program. If AMS had been willing to approach all of these new developments within the STI framework much time and money would have been saved.

4) Finally, your last paragraph. Here is one of the conflicts built into TEX. If authors are encouraged to use whatever language they choose, how does anyone read the data base?

Jim Roesser's Memo and Responses – Wrapup

I suspect that most comments expressed on both sides will not convince users of either TeX or the STI program who are already convinced that their own preferred method is best. As I said before, many of these differences of opinion are valid, and depend almost entirely on the task to which a system is to be applied. TeX genuinely does give typesetting power directly to authors, who can make good use of it in generating theses, lecture notes and preprints; anyone who has labored over a mimeograph stencil, trying to make sense of an author's inscrutable handwriting, can only be grateful for the advent of readily-available computing power and intelligible printouts.

It would be quite rash for journals not to make rules for authors who intend to submit computer-readable manuscripts. The use of widely-available, powerful macro packages such as AMS-TeX and LaTeX will certainly be more attractive to most authors than the idea of writing their own macros. Of course, there are always hackers who are discouraged by *nothing* (after all, macro writing is *fun*!), but I suspect that, in practice, there will be relatively little tinkering, and considerable reliance on existing packages.

Some features of the STI system are very attactive in a production environment: (1) A very wide variety of fonts is available, potentially any font in the typesetter manufacturer's library; those manufacturers will have to become persuaded that TeX users constitute a lucrative market before such a situation will exist for TeX. (2) STI pagination, although designed for manual intervention after all line and paragraph decisions are complete, does allow a page to be made shorter or longer than the norm to avoid very short final pages; it is not particularly easy in TeX to force one excess line onto the previous page. (3) The STI program usually keeps going, regardless of the severity of input errors, allowing output to be generated for proofing; with TeX, a keyboarder may tempted to stop for every error, and reprocess the file from the beginning. We still have insufficient experience with TeX82's error interrupts to know whether the number of runs per file will be decreased substantially from our TeX80 experience.

Regarding the keyboarding costs cited in Jim's response to Mike Spivak: if composition (which also includes such things as computer time and overhead) were really available that cheaply, AMS composition would most likely be sent out, rather than being done in-house. We are interested in any system that keeps costs down, provides flexibility in creating and reusing files, and permits files to be shared with other organizations without massive reprocessing. We have been satisfied with the cost of preparing and processing single-use files using the STI system; that system does not yet provide sufficient flexibility for sharing and reuse, although that power could probably be developed (and we shall not hesitate to offer suggestions for doing so). We have been waiting several years for TeX to stabilize to the point where only a single conversion will be needed to bring us up-to-date; that time is now upon us, and we estimate that a good six months or more of work will be required. We wouldn't undertake the task unless we thought the results would pay for the effort. That's the bottom line for any publisher, and only time will tell.

Readers of this commentary are invited to send in their thoughts. Anything received will be forwarded to all parties, and a summary published in the next issue. Barbara Beeton

* * * * * * * * * * *

Late-Breaking News

* * * * * * * * * * *

The following article by Mike Spivak gives details of how to use the TeX82 version of the AMS-TeX macro package, as he presented them at the Beginners' Course held in conjunction with the July TUG meeting. This article wasn't really late, but is too large to fit in any other column.

Mike's article is published here by permission of the American Mathematical Society; it will also be published as a supplement to the reprint of *The Joy of TeX*, version 0, which still describes AMS-TeX for TeX80. *Joy* for AMS-TeX82 will not be ready for at least several months.

In the meantime, Mike has requested that any comments on AMS-TeX82 be sent to him in writing, c/o the American Mathematical Society, P.O. Box 6248, Providence, RI 02940. Please make a distinction between TeX and AMS-TeX questions, asking your local TeXnician for assistance first, if possible, and send only AMS-TeX questions to Mike.

The real news is, this article is the first output from TeX82 at the AMS to be published in TUGboat. It was manufactured using macros and an AMS-TeX manual 'style' file created by Mike, with a bit of tinkering to apply the TUGboat running heads. With luck, time and persistence, it may actually be possible for the next issue of TUGboat (that's in 1984) to be prepared with TeX82.

Barbara Beeton

SUMMARY OF $\mathcal{AMS}$-TeX

MICHAEL SPIVAK
September 13, 1983

This is a brief summary of $\mathcal{AMS}$-TeX, as it is now written for TeX82. The old "Joy of TeX" manual is woefully out of date, but I will not be able even to begin on the new version until November, so this summary has to be used as a supplement. It probably won't be of much help for learning $\mathcal{AMS}$-TeX from scratch, but people who are already familiar with the old $\mathcal{AMS}$-TeX can use it to see what new things are available, and what changes have been made. (Some of the more esoteric features of $\mathcal{AMS}$-TeX aren't mentioned here, and will have to wait for the manual.)

BASICS

Control Sequences. Control sequences are of two types. The first, like `\par`, consists of \ followed by any number of *letters*. The second, like `\$`, consists of \ followed by just one *non-letter*. A control sequence of the first kind is ended by any non-letter, including a space, and a space after such a control sequence is *not* considered as a space within the text. When an interword space does have to be indicated after such a control sequence, you can use `\␣`, where ␣ indicates a typed space; this is a control sequence of the second type. A control sequence of the second kind needs no special ending, since it is only one character long. A space after such a control sequence *does* count as a space in the text. Multiple spaces count as one space, so `\$␣␣` gives only one space after the printed $. A special exception is made for the control sequence `\␣`; here a space after the control sequence *is* ignored, so that `\␣␣` still gives one space (this of course complies with the general rule that multiple spaces count as one space). Also, spaces in math mode still don't count, so it doesn't matter about spaces after control sequences of this sort.

Document Formatting. Your file should begin

```
\input amstex
\documentstyle{amsppt}
...
\document
```

with the line

```
\enddocument
```

at the very end of the file. (If you are using a program `amstex`, which consists of `tex` with the file `amstex.tex` preloaded, then you shouldn't have the initial line `\input amstex`.)

The control sequence `\documentstyle` is supposed to allow you to select the particular format, but at the moment the only style available is the `amsppt` (AMS preprint) style. When you use this style you will get the question

```
Do you want output? (y or n, follow answer by return)
```

on your terminal. If you type anything other than `y` or `n` (or `Y` or `N`), followed by ⟨carriage-return⟩, the question will be repeated.

The other things that go at the beginning of the paper, between `\documentstyle` and `\document`, will be covered later.

DEFINING CONTROL SEQUENCES

To define your own control sequence `\cs` you type

```
\define\cs{...}
```

Typeset by $\mathcal{AMS}$-TeX

`\define` will check for you whether `\cs` already has a meaning, and if so it will issue an error message. It's actually rather improbable that you will stumble upon one of TeX's control sequences, since most of them have long and unlikely names. Most of the control sequences consisting of \ followed by a single non-letter do have meanings, but `\0`, `\1`, ...,`\9` and `\.` don't, and they are convenient ones to choose, especially for temporary definitions in a complicated math formula (definitions made inside `$` signs disappear at the closing `$` sign). If you want to redefine one of your own control sequences, you can use `\redefine`. You might even want to `\redefine` some of TeX's control sequences, provided you can be reasonably sure that they are not used in some hidden way somewhere else. For example, we'll see later that `\b` and `\d` are used for certain accents under letters, and you might want to use them for something else, say for `\beta` (β), and `\delta` (δ). So you would say

```
\redefine\b{\beta}
\redefine\d{\delta}
```

(using `\define` would give an error message). If it turns out that you also need the original `\d` (which gives an under-dot accent), you can use `\predefine`:

```
\predefine\underdot{\d}
\redefine\d{\delta}
```

Then you can type '`\underdot x`' to get 'x̣' if you ever need it. The only important thing is that the `\predefine` comes before the `\redefine` for `\d`.

When you are defining control sequences with arguments,

```
\define\cs#1#2{...}
```

you must be careful not to have spaces in the expression `#1#2{`. But you *don't* have to worry about spaces when using `\cs`. TeX will ignore spaces between arguments.

Ordinary Text

Control Sequences for Special Symbols. If your keyboard is missing certain symbols you might be able to make do with

`\plus`	+
`\equal`	=
`\less`	<
`\more`	>
`\lq`	‘
`\rq`	’
`\vert`	|

(If you need `\vert` for the |, then you will also need `\Vert`, for \|.)

More important, the following symbols are special in AMS-TeX:

```
\   {   }   $   &   #   %   "   @   ~   ^   _
```

Some of these have printed versions that can appear as symbols in ordinary text, and you get them by typing the obvious things:

`\{`	{
`\}`	}
`\$`	$
`\&`	&
`\#`	#
`\%`	%
`\@`	@

(You can also use `\_` to get a printed _, for things like *first_letter*, which computer scientists like to use.)

Paragraphing and special text symbols. New paragraphs are indicated by either a blank line, or the specific control sequence \par. Double quotes “ and ” are produced simply by typing two single quotes in a row: `` and ''. If you don't have ` or ' you can use \lq and \rq. These work even in pairs—\lq \lq gives “. Hyphens, en-dashes and em-dashes should be distinguished as follows

hyphen (-)	-
en-dash (–)	--
em-dash (—)	---

A hyphen in math mode (between $ signs) becomes a minus sign −.

Italic type is indicated by \it and **boldface** type by \bf, while \sl gives type that is *slanted*, and \rm indicates a return to ordinary (roman) type. You should add the “italic correction” \/ after something in italic or slanted type unless it is followed by a period or a comma:

```
{\it Italic\/} type ... type that is {\sl slanted}, and ...
```

Accents. Here are the common accents:

\\`o	ò
\\'o	ó
\\^o	ô
\\"o	ö
\\~o	õ
\\=o	ō
\\.o	ȯ
\\v o	ǒ
\\u o	ŏ
\\H o	ő (long Hungarian umlaut)
\\t oo	o͡o (tie-after accent)
\\c o	o̧ (cedilla)
\\d o	ọ
\\b o	o̲

You can also type \\`{o} or \v{o}, etc., but you should accent only single letters.

And here are the special letters:

\oe, \OE	œ, Œ
\ae, \AE	æ, Æ
\aa, \AA	å, Å
\o, \O	ø, Ø
\l, \L	ł, Ł
\ss	ß

Also, for accents over ‘i’ and ‘j’ one needs the dotless ‘ı’ and ‘ȷ’, which you obtain by typing \i and \j.

Miscellaneous considerations for ordinary type.

1) TEX puts more space after the ends of sentences than between words, so it needs to know which periods represent ends of sentences. Periods after upper-case letters are not considered to be ends of sentences, since they are usually initials. But periods after abbreviations like “Mr.” cause problems. So you can use \␣ to indicate an ordinary interword space: \Mr.\ Jones. Actually, in this case it would be better to type \Mr.@Jones, using AMS-TEX's “tie” ~, which indicates an ordinary interword space and also tells TEX to try not to break the line at that point.

On those rare occasions when you need a period after a capital letter, and it really is the end of a sentence, use \period:

```
Supported by the NSF \period And numerous others.
```

3) To get a footnote at some point,[1] you just type

```
... at some point,
\footnote {Here is a footnote}
you just type
```

and sure enough it will appear, at the bottom of the page. The particular style you are using will determine what symbols to use and whether numbering is consecutive or begins anew on each page. In the **amsppt** style, superscript numbers are used, and they begin anew on each page. On rare occasions the numbering won't be right, because TeX will have decided that a footnote should have number 3, say, before realizing that it will actually wind up on the next page and should therefore have the number 1. On such occasions you can say

```
\adjustfootnote{-2}
```

right before the footnote, to get the numbering decreased by 2 as needed.

Notice that the space or ⟨carriage-return⟩ after the final } of the `\footnote` is what provides the space before the next word "you". If you don't want the space[2], you just omit it:

```
the space \footnote {...},
you just omit it:
```

4) Use `\dots` for the three dots ... that indicate an ellipsis:

```
the three dots \dots that indicate an ellipsis:
```

5) A % on a line of input causes TeX to ignore everything on the line from that % to the end, *including* the ⟨carriage-return⟩ at the end. So

```
... by Schwartz's Lemma%  Is this how you spell Schwartz?
we have
```

prints out

by Schwartz's Lemmawe have

which isn't quite what you had in mind when you wrote that reminder to yourself; be sure to type

```
... by Schwartz's Lemma %Is this how you spell Schwartz?
we have
```

to get the space after "Lemma", since the ⟨carriage-return⟩ that normally gives it has been obliterated by the %. You can take advantage of this behavior of % to break a long word near the end of a line right in the middle.

If you want to "comment out" a large portion of text, instead of putting % in front of each line, you can type

```
\comment
...
...
\endcomment
```

and everything between `\comment` and `\endcomment` will be ignored. But the `\endcomment` must be on a line by itself (although it can be preceded by blank spaces).

[1] Here is a footnote

[2] When TeX set the footnote at the top of this page it was still working on the previous page (since it didn't know yet just where the best page break would be). Luckily, that footnote still got the number 1, because there were no footnotes on the previous page. But after TeX had completed the previous page, it started numbering footnotes over again, so *this* footnote originally got the number 1 also! I had to put `\adjustfootnote{1}` right before it.

6) For the proper spacing between quotes within quotes, use `\qspace`:

```
‘‘They call this ‘typesetting’\qspace’’ he sneered.
```

This will work for ’qspace” or ”’, and ‘“ or “‘.

7) Use `\dash` for an em-dash (—) that can go at the beginning of a line, if you really need it to avoid a bad line break (normally, TeX will only put an em-dash at the end of a line, not at the beginning of one). Similarly, use `\slash` instead of / in a word like `and/or` if you want to allow the “and/” to be at the end of one line, with the “or” at the beginning of the next.

8) To get an paragraph to be unindented, use `\noindent` at the beginning of it.

9) Between paragraphs you can use `\smallvspace`, `\medvspace` and `\bigvspace` to get a small, medium or big amount of vertical space; there is a `\bigvspace` before this `\noindent`'ed paragraph, a `\medvspace` before the previous one, and a `\smallvspace` before that one.

10) Between paragraphs if you type `\midspace{⟨dim⟩}` for some “dimension” ⟨`dim`⟩ you will get a space of that dimension, if it will fit on the page; otherwise it will be on the top of the next page. A “dimension” is a number followed by a unit like `in` (inches), `cm` (centimeters), `pt` (point) or `pc` (pica), to name the ones most likely to be used. If you type `\topspace` instead of `\midspace` the space will be at the top of the current page if it fits, or at the top of the next page.

11) `\linebreak` breaks a line right where it is typed, without spreading the line out. `\newline` spreads the line out.

`\pagebreak` breaks the page immediately after the present line is completed. `\newpage` does the same, but fills up the remainder of the page with blank space, instead of spreading out the lines if the page is short.

12) Use `\TeX` to get the TeX logo, and `\AmSTeX` to get the AMS-TeX logo.

More About Document Formatting

Topmatter. At the beginning of your file, right after the `\documentstyle` line, there will normally be

```
\topmatter
...
...
\endtopmatter
```

for the “topmatter”, like the title, author, etc. (some of which may actually end up at the bottom of the title page, or even at the end of the paper). Here are the things you can have (the order is irrelevant):

```
\title...\endtitle
\author...\endauthor
\affil...\endaffil
\address{...}
\date{...}
\thanks{...}
\keywords{...}
\subjclass{...}
\abstract{...}
```

The first three use a different syntax because each of them can be made multi-line, by inserting `\\` between the lines. `\address` can be used several times in a row, for different authors, say. In the `amsppt` style, the addresses appear at the end, along with the `\keywords`, prefaced by “*Keywords.*” and the `\subjclass`, prefaced by “1980 *Mathematics subject classifications.*” The abstract begins with the word “ABSTRACT.”, so this shouldn't be typed in.

Sometimes acknowledgements and affiliations are handled as footnotes on the title and/or author, instead of using `\thanks` and `\affil...\endaffil`. That's no problem—`\footnote` can be used in `\title...\endtitle` and `\author...\endauthor` (and most other places, as well, although the footnote may disappear if you bury it really deeply within some complicated construction—don't worry about it until it happens).

There's also `\TITLE...\endTITLE` and `\AUTHOR...\endAUTHOR` to automatically print the title and author in capital letters. Normally, these will be used by various journal styles, when it is desirable to have the title and author appearing in the file the same way that it may appear in some index, say. In fact, various journal styles will probably just have `\title` mean `\TITLE`, so that there really isn't any need to use `\TITLE` yourself.

There are a couple of other neat little things that you can do near the beginning of the file. First of all, you can say `\guidelines` before `\document` to get horizontal lines at the top and bottom of the page—these can be useful when your output come out on a roll of paper, and you can't figure out just where to cut it. The guidelines are normally about two lines above and below the actual page. You can say `\guidelinegap`⟨`dim`⟩ to specify some other dimension ⟨`dim`⟩ for the gap.

In a few places the **amsppt** style assumes that your paper is in English, for it begins the abstract with the word "ABSTRACT.", puts the logo "Typeset by $\mathcal{AMS}$-TEX" at the bottom of the first page, labels the bibliography (which we'll get to later) as "REFERENCES", and prefaces the `\keywords` and `\subjclass` with appropriate English words. If you put `\german` before the `\document`, however, all of these will come out in German. Actually, at the moment only "ABSTRACT." will be changed, to "ZUSAMMENFASSUNG.", since I don't know what the proper German for the other things is. But this will be changed as soon as someone tells me, and other languages will be added as requested.

Incidentally, if you can't stand having the "Typeset by $\mathcal{AMS}$-TEX" logo at the bottom, no matter what language it is in, put `\nologo` at the beginning.

The paper proper. The heading for this section was made by typing

```
\heading More About\\Document Formatting\endheading
```

with `\\` indicating linebreaks, as usual. `\subheading{...}` provides the boldface flush-left headings like '**The paper proper.**' (the period is supplied automatically); this can't be multi-line, so the syntax is different.

To state theorems, lemmas, and other proclamations, you just say

```
\proclaim{Theorem 4} Blah, blah, blah.
\finishproclaim
```

and you will get

THEOREM 4. *Blah, blah, blah.*

Notice that the punctuation after the label of the theorem comes for free. All `\finishproclaim` really does is switch back to regular type, and perhaps provide a little extra spacing, so if you forget it the paper will look strange, but it won't be catastrophic. We use `\finishproclaim` instead of `\endproclaim` because the `\end`... syntax is reserved for things with `\\` in them; but amstex recognizes `\endproclaim` as a synonym for `\endproclaim`, in case you forget.

For the proofs, use

```
\demo{Proof} Here is the proof.
\finishdemo
```

Once again, omitting `\finishdemo` will not be catastrophic—all it will lose is the special spacing after the end of the proof.

Mathematicians frequently list things like this:

(a) The first thing on the list. We will make this very long so that you can see what happens when the condition is more than one line long,
(b) The second thing,
(c) The final thing.

You get this by typing

```
\conditions
(a) : The first thing on the list.  We will make this very long so that you
can see what happens when the condition is more than one line long,\\
(b): The second thing,\\
(c): The final thing.\endconditions
```

As usual, \\ indicates the line breaks in this construction: the colon separates the label for the condition from the condition itself. In the amsppt style the label gets printed in \rm, unless some other font is explicitly given, while the condition is printed in whatever font is being used at the time (so that \conditions in a \proclaim will get printed in \sl). When you need "runin" conditions like (a) The first,
(b) The second,
(c) The last.
just type \runin right before the \conditions.

By the way, sometimes you may not want to start a paragraph after the statement of a theorem, but you may find that \finishproclaim does it for you, even though you didn't leave a blank line. The same problem will arise with \finishdemo and \endconditions. You will then need to use \noindent; in later versions of $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX this annoyance will go away.

There are \proclaimnp and \demonp, if you need to have the \proclaim or \demo supply No Punctuation.

Finally, there's \refto. \refto{10} gives [**10**], while \refto{10, Theorem 4} gives [**10**, Theorem 4], with only the stuff before the comma in boldface.

BIBLIOGRAPHY

When you are ready to type the bibliography, you first type \Refs, which produces the heading REFERENCES, or whatever the particular style uses, and sets things up for typing individual references. A typical individual reference would be

```
\ref \no 9 \by S. S. Chern \pages 947--055
\paper Integral formulas for hypersurfaces in Euclidean space and
  their applications to uniqueness theorems
\yr 1959 \vol 8
\jour J. Math. Mech.
\endref
```

which in the amsppt style will produce

9. S. S. Chern, *Integral formulas for hypersurfaces in Euclidean space and their applications to uniqueness theorems*, J. Math. Mech. **8** (1959), 947–055.

Notice that there is no need to specify the various pieces of the reference in the order that they will be printed, nor do you have to worry about the proper font or details like enclosing parentheses; all this is done by the style file. Notice also that we don't have \␣ after Math. and Mech.; within \Refs all spaces will count as usual interword spaces, even those after periods (if you do need the usual space after a period, for example if a title consists of two sentences, you can always use \period, which has already been mentioned in another context).

There is no specific amount of information that you have to include in each \ref...\endref. The style file will do the reasonable thing with the information you give. For example, if you leave out the \vol, then it just won't get printed, but if you leave out \jour, then \vol and \yr will be ignored, even if you put them in.

Other things you can have within \ref...\endref are \toappear, which typesets "(to appear)" and \issue, which might be needed for some sort of special issue. If the next reference is by the same author, you can type \bysame instead of \by ...; in the amsppt style this will produce a line just the length of the author's name in the previous reference instead of repeating it.

Instead of \no for a number you can use \key for some other sort of "key", like \key[{\bf C1}] to get "[**C1**]" instead of a number in front of the reference. Both \key and \no can be omitted.

If the paper is just one page long, you might use `\page` instead of `\pages`; this will make sure that the single page gets "p. " printed in front of it to avoid confusion.

Finally, there is indeed a use for `\\` in `\ref...\endref`, though it's for special things, where two references by the same author are combined because they are closely related. For example,

```
\ref\no4 \by L. Auslander \paper On the Euler characteristic of compact
locally affine spaces \jour Comment. Math. Helv. \vol 35 \yr 1961
\pages 25--27\\ \paper II \jour Bull. Amer. Math. Soc.
\vol 67 \yr1961\pages 405--406 \endref
```

will give

4. L. Auslander, *On the Euler characteristic of compact locally affine spaces*, Comment. Math. Helv. **35** (1961), 25–27; *II*, Bull. Amer. Math. Soc. **67** (1961), 405–406.

For book references there are `\book`, `\publ` and `\publaddr`. For example,

```
\ref\no7\by H. Bass\book Algebraic $K$-theory\publ W. A. Benjamin
\publaddr New York\yr 1968\pages 15--19\endref
```

gives

7. H. Bass, "Algebraic K-theory", W. A. Benjamin, New York, 1968, pp. 15–19.

Notices that in this cases `\pages` gave "pp. " before the pages.

For a paper that appears in a book, rather than in a journal, just use `\paper` together with `\inbook`.

Finally, there are `\paperinfo`, `\bookinfo` and `\finalinfo` to put in extra information after the paper, after the book, or at the very end. Punctuation around `\paperinfo` and `\bookinfo` is handled automatically. Punctuation for `\finalinfo` has to be supplied.

Mathematics

Getting Into Mathematics. Math formulas in text are indicated by `$...$`; displayed math formulas by `$$...$$`. Within a math formula, spaces are completely irrelevant (except, of course, the spaces needed at the end of control sequences).

Special Symbols. Aside from the symbols <, >, +, = and |, which are probably on your keyboard, you will need control sequences for most of the other special symbols of mathematics. They are listed in Appendix F of The TeXbook. Some important ones are

`\le` or `\leq`	$\le$ ("less than or equal")
`\ge` or `\geq`	$\ge$ ("greater than or equal")
`\ne` or `\neq`	$\ne$ ("not equal")
`\in`	$\in$
`\notin`	$\notin$
`\infty`	∞

All the Greek letters have special control sequences to name them, like `\alpha` (α), `\beta` (β), `\gamma` (γ) and `\Gamma` (Γ). Several characters have variants:

```
$$(\epsilon,\phi, \theta,      \pi)$$
```

yields

$$(\epsilon,\phi,\theta,\pi)$$

while

```
$$(\varepsilon, \varphi, \vartheta,\varpi)$$
```

yields

$$(\varepsilon,\varphi,\vartheta,\varpi)$$

Switching Fonts. You can also switch to other fonts within a math formula, exactly as in text. If you type

```
$${\rm a + \bf b - c}/d$$
```

you will get

$$\mathrm{a} + \mathbf{b} - \mathbf{c}/d$$

Notice that, just as in text, font changes work for the whole formula, not just for the next letter. Only the "variables" change fonts; symbols like $+$ and $-$ stay the same.

There's a font called `\cal` that gives "calligraphic" characters:

```
$${\cal A} + B$$
```

gives

$$\mathcal{A} + B$$

This font should only be used for upper case letters. However, there's also a character ℓ, which you get with the control sequence `\ell`.

One other font change that's useful for math is `\mit`, the "math italic" font. This should be used only for capital Greek letters, to give things like $\mathit{\Gamma}$ (`$\mit \Gamma$`), as compared to the ordinary $\Gamma$, which you get simply by typing `$\Gamma$` (or even `$\rm \Gamma$`).

Bigger Parentheses, etc. In addition to the ordinary size parentheses that you get in math formulas, like $f(x+y)$, you can get slightly larger ones with `\bigl` and `\bigr`, which also apply to things like `[` and `]` and `\{` and `\}`:

<table>
<tr><td><code>\bigl(X \bigr)</code></td><td>$\bigl(X \bigr)$</td></tr>
<tr><td><code>\bigl[X \bigr]</code></td><td>$\bigl[X \bigr]$</td></tr>
<tr><td><code>\bigl\{X \bigr\}</code></td><td>$\bigl\{X \bigr\}$</td></tr>
<tr><td><code>\bigl|X\bigr|</code></td><td>$\bigl|X\bigr|$</td></tr>
<tr><td><code>\bigl\|X\bigr\|</code></td><td>$\bigl\|X\bigr\|$</td></tr>
</table>

These are useful for formulas like

$$\bigl(x - f(y)\bigr)\bigl(x + f(y)\bigr)$$

There are also `\biggl` and `\biggr` sizes, which fit nicely around two-line fractions:

`\biggl(\biggr)` $\biggl(\biggr)$

Superscripts and Subscripts.

Superscripts are indicated by `^`, subscripts by `_` (that's the "underscore" key, not the hyphen). You can also use `\sp` and `\sb`. The symbols `^` and `_` apply only to the next character, so you need braces if you want them to apply to more:

`$x^\alpha$`	x^α
`$x_{x+y}$`	x_{x+y}
`$x^{(x+y)}$`	$x^{(x+y)}$
`$x^{12}$`	x^{12}

Notice that `\alpha` is just one thing in TeX's mind, so no braces are needed there, but the (typeset) number 12 is two things.

For things like

$$a^{b^c}$$

just think of the formula as a mathematician does: the a has a superscript, and this superscript is—not b, but the whole formula b^c. So you type

```
$$a^{b^c}$$
```

You can have simultaneous superscripts and subscripts:

`$A_a^b$` A_a^b

Remember that `\prime` ($\prime$) is special, because you need to superscript it to get it the right size:

`$f^\prime$` $f^\prime$

`$f^{\prime\prime}$` $f^{\prime\prime}$

But any number of single quote marks (`'`) in a row in a math formula will translate into `^{\prime\prime...}` with the appropriate number of `\prime`'s:

`$f''_2$` f''_2

(You'll still need `\prime` for the occasional formula like

`$f^{\prime2}$` $f^{\prime2}$

which is the common way to typeset such things.)

Fractions, etc. To get a fraction like

$$\frac{a}{b}$$

you type

```
$$\frac ab$$
```

and to get

$$1+\frac{a+b}{c+d}$$

you type[1]

```
$$1+\frac {a+b}{c+d}$$
```

Notice that

```
$$\frac {\frac ab}{c+d}$$
```

gives

$$\frac{\tfrac{a}{b}}{c+d}$$

where the numerator is now in the size normally used for text. TeX uses four sizes for mathematics formulas: `\dsize` is the size used for display, `\tsize` the size for text, `\ssize` the size for superscripts, and `\sssize` the size for superscripts to superscripts (the letter c in a^{b^c}). You can get the fraction $\tfrac{a}{b}$ in the above formula bigger by typing

```
$$\frac {\dsize\frac ab}{c+d}$$
```

to get

$$\frac{\dfrac{a}{b}}{c+d}$$

The size change macros `\dsize`, etc., work just like font changes—they change sizes permanently within a formula, or in some subformula delimited by braces. These are the only macros that work this way.

An easier way to get the above formula is to use `\Frac`, which automatically makes the `\frac` into `\dsize`:

```
$$\frac {\Frac ab}{c+d}$$
```

[1] If you are enamored of AMS-TeX's old way of doing things, just use TeX's primitive `\over`; in the old AMS-TeX `\frac` was just a new name for `\over`, but `\over` seems to cause so much confusion that it's been replaced by a control sequence that works like all the others. Sometimes `\over` will save a pair of braces (though sometimes the new `\frac` will also), but you often have to think ahead and put in certain braces before their time has come, and it seems easier to do things the new way.

In addition to `\frac`, there's `\stack` and `\binom`:

`$$\stack ab$$`	$\begin{matrix}a\\b\end{matrix}$
`$$\binom {n+1}k$$`	$\binom{n+1}{k}$

`\binom` is common; `\stack` occurs in only a few special situations, that we'll see later. There's also `\Binom` to get the `\binom` in `\dsize` (but there's no special `\Stack`, since that's almost never wanted). Finally, there's `\thickfrac` for a fraction with a thick line (rarely used):

`$$\thickfrac {\Frac ab +\Frac cd}{C+D}$$`	$\dfrac{\frac{a}{b}+\frac{c}{d}}{C+D}$

Variable Size Symbols. You can underline and overline formulas:

`$$\underline{x+y+z}$$`	$\underline{x+y+z}$
`$$\overline{x+y+2^{\underline 2}}$$`	$\overline{x+y+2^{\underline{2}}}$

There are also `\overrightarrow`, a.k.a. `\overarrow`, `\overleftarrow`, `\overleftrightarrow`, `\underrightarrow`, `\underleftarrow` and `\underleftrightarrow`, all of which do the obvious thing.

`\sqrt` gives square roots automatically of the right size:

`$$\sqrt 2$$`	$\sqrt{2}$
`$$\sqrt{\frac ab+1}$$`	$\sqrt{\frac{a}{b}+1}$

Most important, you can get variable size parentheses, etc., using `\left` and `\right`:

`$$\left(\frac 1{1-x^2}\right)$$`	$\left(\frac{1}{1-x^2}\right)$
`$$\left[\frac 1{1-x^2}\right)$$`	$\left[\frac{1}{1-x^2}\right)$

Note that the `\left` and `\right` things don't have to match; you can even do things like `\left]`. But every `\left` needs a matching `\right`, so for special things like

$$\left.\frac{dy}{dx}\right|_{x=a}$$

there are `\left.` and `\right.` to give a `\left` or `\right` empty symbol:

```
$$\left.\frac {dy}{dx}\right|_{x=a}$$
```

Other possible delimiters are:

`|` (or `\vert`)	$\vert$
`|` (or `\Vert`)	$\Vert$
`\{`	$\{$
`\}`	$\}$
`\lfloor`	$\lfloor$
`\rfloor`	$\rfloor$
`\langle`	$\langle$

`\rangle`	$\rangle$
`\slash`	$/$

The last three are special, because they come in only a limited number of sizes; the others are, in the largest sizes, built up of extensible pieces:

$$\left(\rule{0pt}{4em}\right)$$

Large Operators. The next thing you want to know about are "large operators", which include things like $\int$ (`\int`) and $\sum$ (`\sum`), as well as related things to be found in Appendix F of The TEXbook. The important thing to remember about `\sum` is that when you type `\sum_{i=1}^na_i` between dollar signs it comes out as $\sum_{i=1}^n a_i$, as expected, but when you type the very same thing as a displayed equation it comes out as

$$\sum_{i=1}^n a_i$$

which is the way such formulas are usually displayed: Not only does the $\sum$ sign change size, but the sub- and super-script are set as "limits".

`\int`s are similar, except that they merely change sign; normally

`$$\int_a^b f(x)dx$$`

gives

$$\int_a^b f(x)dx$$

Some printers set things differently, so the format you are using may use "limits" for `\int`s, and not use them for `\sum`s. You can change things yourself in the **amsspt** style by typing

`\LimitsOnInts`

to get limits on `\int`s and

`\NoLimitsOnSums`

to avoid having limits on `\sum`s. (And, `\NoLimitsOnInts` and `\LimitsOnSums` get you back again.) But these should be used only for changing the **amsppt** style; most journals insist on their own styles, and their style files will then simply ignore these instructions

Occasionally, no matter what style is being used, you need to change the way an individual $\sum$ or $\int$ is set. This is easily done by following it with `\limits` or `\nolimits` to tell TEX that limits should or should not be used. This works even in text. For example, `$\sum\limits_{i=1}^Na_i$` gives $\sum\limits_{i=1}^N a_i$, as opposed to $\displaystyle\sum_{i=1}^N a_i$, which we got with `$\dsize\sum_{i=1}^Na_i$`.

Whatever convention is being used for `\sum` will also hold for most of the other large operators, like $\prod$ (`\prod`). But the symbol $\oint$ (`\oint`) will be treated like $\int$. The same is true for

`$$\intii$$`	$\displaystyle\iint$
`$$\intiii$$`	$\displaystyle\iiint$
`$$\intiv$$`	$\displaystyle\iiiint$

```
$$\intdotsint$$
```

$$\int\cdots\int$$

One of the things to note about `\sum` is that you usually *don't* want to type

```
$$\left(\sum_{i=1}^N a_i\right)$$
```

because it gives

$$\left(\sum_{i=1}^N a_i\right)$$

instead of the preferred

$$\biggl(\sum_{i=1}^N a_i\biggr)$$

This formula was typed using `\biggl` and `\biggr`.

One other thing you sometimes see is something like

$$\sum_{\substack{1\le i\le N\\ 1\le j\le M}} a_{ij}$$

Here we use `\stack`:

```
$$\sum_{\stack{\ssize1\le i\le N}{\ssize1\le j \le M}}a_{ij}$$
```

Notice that `\ssize` was needed for both `1\le i\le N` and `1\le j \le M`, since they would normally be in `\ssize`, since the whole stack is in `\ssize`.

Spacing. You seldom have to tell TEX to put in spacing in math formulas. But there is one case where you always should. The formula

$$\int_a^b f(x)dx$$

given above really isn't right because it's conventional to leave a certain amount of space before the dx. This space, which printers call a "thin space", is called `\,` by TEX. So you should type

```
$$\int_a^bf (x) \, dx$$
```

$$\int_a^b f(x)\,dx$$

The other case where you need to specify spacing is in a formula like

$$f(x,y)=0\qquad x,y>0$$

The space before the "side-condition" $x, y > 0$ is traditionally two "quads", where a quad is another standard printer's space. AMS-TEX has `\quad` for this amount of space and `\qquad` for the more usual `\quad\quad`, so the above formula was typed as

```
$$f(x,y)=0\qquad x,y>0$$
```

Ordinary type. Sometimes you have a formula like

$$f(x,y)=0\qquad \text{for all } x,y>0$$

that includes ordinary type. One way to get this is to switch to `\rm`:

```
$$f(x,y)=0\qquad{\rm for all\ }x, y > 0$$
```

Notice that we had to sequester the `\rm` so that it didn't affect the x and y. Notice also that we had to use `\␣`, since spaces are ignored in math. For such situations it is almost always better to use AMS-TeX's `\text{...}`, which puts you back into ordinary text:

```
$$f(x,y)=0\qquad\text{for all }x,y>0$$
```

Now all we need to remember is that space after the `all` (it doesn't have to be `\␣` since we're out of math when we're in `\text`). Even easier is to go into `\text` and then within `\text` switch back to math!

```
$$f(x,y)=0\qquad\text{for all $x,y>0$}$$
```

Just be sure you get the order of `$` signs and `}`s correct.

Sometimes roman type is used in a different way, as in the formula

$$\sin(x+y) = \sin x\cos y + \cos x\sin y$$

Here sin stands for a special mathematical operator, and the spacing follows certain important rules; note, for example, that there is no space between sin and $(x+y)$, but there is space between sin and x. You don't have to worry about any of this, just use `\sin`:

```
$$\sin  (x+y) = \sin x \cos y + \cos x\sin y$$
```

Other familiar operators are

```
\arccos \cos  \csc \exp \ker     \limsup \min \sinh
\arcsin \cosh \deg \gcd \lg      \ln     \Pr  \sup
\arctan \cot  \det \hom \lim     \log    \sec \tan
\arg    \coth \dim \inf \liminf  \max    \sin \tanh
```

Some of these have "limits" just like $\sum$:

```
$$\max_{1\le n\le m}\log_2P_n$$
```

$$\max_{1\le n\le m}\log_2 P_n$$

The control sequence `\operatorname` produces new gadgets like `\sin`, and `\operatornamewithlimits` produces new gadgets like `\max`. If you say

```
\define\gadget{\operatorname{gadget}}
\define\Gadget{\operatornamewithlimits{Gadget}}
```

then you can type

```
$$\gadget^2(x+y)-\gadget x + \Gadget_{i<j}a_{ij}$$
```

to get

$$\operatorname{gadget}^2(x+y)-\operatorname{gadget} x + \operatorname*{Gadget}_{i<j}a_{ij}$$

Numbered equations. If you type

```
$$x=y\tag 3--1$$
```

you will get

(3–1) $x=y$

or

$$x=y \tag{3–1}$$

depending on the format style. The **amsppt** style puts tags on the left. Typing `\TagsOnRight` puts them on the right (and `\TagsOnLeft` get them back to the left again). Like the control sequences `\LimitsOnInts`, etc., `\TagsOnRight` and `\TagsOnLeft` will be ignored by the style files for journals.

Note that the parentheses, or whatever, around the tags are put in automatically by the style file. Note also that the tag was processed as text, rather than as mathematics—so the `--` gave an en-dash rather than two hyphens. For a tag like "2′" you will need to type

```
$$x=y\tag {$2'$}$$
```

(making sure to have the math part `$2'$` inside braces). However `\TagsAsMath` automatically treats all tags as math, so that `...\tag 2'$$` will give the tag (2′) [while `...\tag \text{2--3}$$` will give the tag (2–3)]. `\TagsAsText` gets you back to tags treated as text. Journal formats will not ignore the control sequences `\TagsAsMath` and `\TagsAsText`, since these do not actually effect the form of the output, merely the way that it is specified.

Finally, we ought to mention that a tag might end up on the line just before an equation (or on the line just after it, in the case of right-tagging), if the tag and the equation won't fit nicely on one line. But TEX does all this for you automatically, so you don't have to worry.

Aligned Equations. You get

$$\begin{aligned} x&=y=z\\ x^2&=y^2\\ x^3&\le y^3 \end{aligned}$$

by typing

```
$$
\align x&=y=z\\
 x^2 &=y^2\\
 x^3 &\le y^3
  \endalign
$$
```

The `&`'s come right before the symbols which get lined up, and the `\\`'s separate individual lines.

An `\align`'ed equation is a unit, so that you can get

$$\left\{\begin{aligned} a&=b\\ a^2&=b^2 \end{aligned}\right.\qquad\left\{\begin{aligned} c&=d\\ c^2&=d^2\\ c^3&=d^3 \end{aligned}\right.$$

by typing the following, which incidentally illustrates how a couple of on-the-spot `\define`'s can make life easier:

```
$$
\define\1{\align a&=b\\a^2&=b^2\endalign}
\define\2{\align c&=d\\c^2&=d^2\\c^3&=d^3\endalign}
\left\{ \1 \right.\qquad\left\{ \2 \right.
$$
```

Similarly, you can tag an `\align`. For example,

```
$$\align a&=b\\a^2&=b^2\\a^3&=b^3\endalign \tag 3-1$$
```

comes out as

$$\begin{aligned} a&=b\\ a^2&=b^2\\ a^3&=b^3 \end{aligned}\tag{3-1}$$

with the tag centered with respect to the `\align`, since it's the whole `\align` that's being tagged.

A long broken equation like

$$\begin{aligned}(a+b)(a-b) &= (a+b)a-(a+b)b\\ &= a^2+ab-ab-b^2\\ &= a^2-b^2\end{aligned}$$

can be gotten by typing

```
$$
\align (a+b)(a-b) &=(a+b)a - (a + b)b\\
   & = a^2 +ab-ab-b^2\\
  &=a^2-b^2
\endalign
$$
```

You will get the same result if you use `\broken`...`\endbroken`:

```
$$
\broken (a+b)(a-b) &=(a+b)a - (a + b)b\\
   & = a^2 +ab-ab-b^2\\
  &=a^2-b^2
\endbroken
$$
```

But things work out quite differently when you put a `\tag` on a `\broken` equation, like

```
$$\broken a&=b\\&=c\\&=d\endbroken \tag 1--2$$
```

Instead of having the tag centered on the equation you will get either

(1–2)

$$\begin{aligned}a &= b\\ &= c\\ &= d\end{aligned}$$

or

$$\begin{aligned}a &= b\\ &= c\\ &= d\end{aligned}$$

(1–2)

depending on the format.

[Some formats center tags on broken equations, just as if they were an ordinary `\align`. Typing `\CenteredTagsOnBrokens` will cause the `amsppt` style to do this, and `\TopOrBottomTagsOnBrokens` will get the original behavior. These control sequences are, of course, ignored by a journal format style. You can get an individual `\broken` to be tagged in the center (to save some space, perhaps) by using the construction `\cbroken`...`\endcbroken`.]

`\align` or `\broken` can also be used for broken equations like

$$\begin{aligned}Z &= (X+Y)(A+B+C+D+E+F\\ &\qquad +G+H+K+L+M+N)\end{aligned}$$

which we got by typing

```
$$\align Z&=(X+Y)(A+B+C+D+E+F\\
    &\qquad +G+H+K+L+M+N)\endalign
$$
```

Sometimes an equation is broken as follows:

$$\begin{multline*}A+B+C+D+E+F+G+H+I+J+K+L+M+N+P+Q+R+S+T+U\\ =A'+B'+C'+D'+E'+F'+G'\end{multline*}$$

with the first line beginning near the left margin, and the second line ending near the right margin. A long equation might even require three or more lines:

$$\begin{multline*}A+B+C+D+E+F+G+H+I+J+K+L+M+N\\ +A'+B'+C'+D'+E'+F'+G'+H'+I'+J'+K'+L'+M'+N'+P'\\ =P+Q+R+S\end{multline*}$$

You get such displays by using `\multline`:

```
$$
\multline A+B+C+D+E+F+G+H+I+J+K+L+M+N\\
  + A'+B'+C'+D'+E'+F'+G'+H'+I'+J'+K'+L'+M'+N'+P'\\
     =P+Q+R+S
\endmultline
$$
```

The lines between the first and last are centered, but you can shove any of these to the left or right by typing

`\shoveleft{`second line`}`, `\shoveright{`third line`}`, etc.

When a `\multline` is tagged the tag will either be at the beginning of the first line or the end of the last line. One caution here: If the tag and the first line together won't fit, `\multline` *won't* put the tag on a separate line, you'll have to do it by hand (by having an empty first equation). In version 2 of $\mathcal{A}\!\mathcal{M}\!\mathcal{S}$-TeX we hope to fix this problem.

Finally, every once in a while you will need a display consisting of a bunch of equations that are individually centered, instead of being aligned:

$$\begin{gathered}a=b+c\\ d=e\\ f+g=h\end{gathered}$$

You shouldn't type

```
$$a=b+c$$ $$d=e$$ $$f+g=h$$
```

because you'll get too much space between the equations. Instead use `\bunch`:

```
$$\bunch a=b+c\\
  d=e\\
   f+g=h\endbunch$$
```

Notice that there *must not* be any `&`'s here, since nothing is being aligned. Of course, you might well have `&`'s within some `\align` appearing within the `\bunch`. For example,

```
$$
\bunch a=b+c\\
\align d&=e\\&=f\endalign\\
 f+g=h\endbunch
$$
```

gives

$$\begin{gathered}a=b+c\\ \begin{aligned}d&=e\\ &=f\end{aligned}\\ f+g=h\end{gathered}$$

Aligned and tagged equations. For something like

$$\begin{array}{lr} (1) & \qquad\qquad a = b = c \\ & a^2 = b^2 \\ (2) & a^3 = c^3 \end{array}$$

you use \aligntag:

```
$$
\aligntag a&=b=c\tag 1\\
a^2&=b^2\\
a^3&=c^3\tag 2\endaligntag$$
```

Like \multline, \aligntag won't automatically put a tag on a separate line if necessary—you may get a tag overlapping an equation, without getting an Overfull box message! In version 2 this should be fixed.

You can have a \broken inside an \aligntag, with or without a tag, and *everything* will line up nicely. For example,

```
$$
\aligntag (a+b)(a+b)&=a^2+2ab+b^2,\tag 1\\
\broken (a+b)(a-b) &=(a+b)a - (a + b)b\\
   & = a^2 +ab-ab-b^2\\
  &=a^2-b^2
\endbroken\tag2\endaligntag$$
```

gives

$$\begin{array}{lr@{}l} (1) & \qquad\qquad (a+b)(a+b) &{}= a^2 + 2ab + b^2, \\ (2) & (a+b)(a-b) &{}= (a+b)a - (a+b)b \\ & &{}= a^2 + ab - ab - b^2 \\ & &{}= a^2 - b^2 \end{array}$$

with the = signs of the \broken lining up with the ones from the \align. (\broken wasn't designed to behave this way inside a bare \align since there's no point using \broken instead of \align if you don't have tags to worry about.)

Between lines of an \aligntag you can type \vspace{⟨dim⟩} to get extra vertical space of dimension ⟨dim⟩ between them. If you type \xtext{...} the extra text ... will appear on a separate line, starting at the left margin, without destroying the alignment of the other lines.

Although an \aligntag is usually treated as a unit, like an \align, you can put \allowbreak between lines to allow a page break, \goodbreak to indicate an especially nice place to break, and even \break to force a break.

There's also \bunchtag, which bears the same relation to \aligntag as \bunch bears to \align.

Matrices. A "matrix" is a square array like

$$\begin{matrix} a & b & c \\ a' & b' & c + c' \\ a'' & b'' & c + c' + c'' \end{matrix}$$

You get this by typing

```
$$\matrix
  a&b&c\\a'&b'&c+c'\\a''&b''&c+c'+c''\endmatrix$$
```

with \\ separating rows, as usual, and & separating elements of the various columns. The elements within each column are centered, and there is a quad of space between the columns. The number of columns will be the largest number specified in all the rows; rows with fewer specified columns will simply have a blank formula at the corresponding place. Thus,

```
$$\matrix
  a\\a'&b'\\a''&b''&c+c'+c''\endmatrix$$
```

gives

$$\begin{matrix} a \\ a' & b' \\ a'' & b'' & c+c'+c'' \end{matrix}$$

You can get a different format for a `\matrix` by using

```
\format ...\\
```

right after `\matrix`. For example, if you want the matrix

$$\begin{array}{crl} x & 1 & .1 \\ x+y & 11 & .11 \\ x+y+z & 111 & .111 \end{array}$$

with the first column centered, the second set flush right and the third set flush left, you type

```
$$\matrix \format \c & \quad\r & \quad\l\\
x&1&.1\\
x+y&11&.11\\
x+y+z&111&.111\endmatrix$$
```

In the format line `\format \c & \quad\r & \quad\l\\` the specifications for the columns are separated by `&`, just as the elements of each row are later; the `\c` indicates that the elements of the first column are centered, the `\quad\r` indicates that the elements of the second column are set flush right, with a `\quad` of space before the column, etc. Any other spacing, like `\,` or `\;` or `\qquad` can be used instead.

There is also a special way of abbreviating formats with many columns that are all treated the same. Suppose, for example, that we want all centered columns, but separated by a thin space `\,` instead of by a `\quad`. We can just use

```
\format \c && \,\c \\
```

The double `&&` means that the remaining specifications are repeated over and over, for as many columns as is necessary. Similarly,

```
\format \c && \quad\l & \quad\r \\
```

will center the first column, and then produce columns that are alternately set flush left and flush right, with all columns separated by a `\quad` of space. You mustn't have `&&` twice in a `\format...\\`.

If you have a special format that you use many times you unfortunately *can't* just say

```
\define\specialformat{\format ... \\}
```

and then save work by typing

```
\matrix\specialformat
  a & b & ...
   ...\endmatrix
```

`\matrix` has to see the explicit sequence `\format ... \\` if it is used. (However, you can say, for example,

```
\define\specialformat{\c&&\quad\l&\quad\r}
```

and then type

```
\matrix\format \specialformat \\
 ...
```

which is almost as good.)

In a `\matrix` the elements are set as math formulas in `\tsize`; use `\dsize` if you explicitly need it.

Usually a matrix will occur with parentheses or something else around it, like

$$\begin{pmatrix} a & b & c \\ a' & b' & c+c' \\ a'' & b'' & c+c'+c'' \end{pmatrix}$$

You can always put in the parentheses with `\left` and `\right`, but there's also `\matrixp` and `\endmatrixp` to do it:

```
$$\matrixp
  a&b&c\\a'&b'&c+c'\\a''&b''&c+c'+c''\endmatrixp$$
```

There's also `\matrixv` to give `\left|`, `\matrixvv` for `\left\|` and `\matrixb` for `\left[`, with corresponding `\endmatrix...` constructions. You can even mix them, to get one thing on the left and another on the right. But be sure not to begin with `\matrixp`, say, and end simply with `\endmatrix`, since this gives a `\left(` unbalanced by a `\right` (if you really want nothing on the right use `\endmatrix\right.`).

For a matrix like

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

you use `\hdots` for the horizontal dots, and `\vdots` for the vertical dots:

```
$$
\matrixp \format \c&&\;\c\\
a_{11}&a_{12}&\hdots&a_{1n}\\
a_{21}&a_{22}&\hdots&a_{2n}\\
\vdots&\vdots& &\vdots\\
a_{m1}&a_{m2}&\hdots&a_{mn}
\endmatrixp
$$
```

(Actually, at the moment you'll have to use `\ldots` (low dots), because I forgot to add the alternative name `\hdots`.)

You might want to get something like

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

To get the diagonal dots just use `\ddots`:

```
$$
\matrixp \format \c&&\;\c\\
a_{11}&a_{12}&\hdots&a_{1n}\\
a_{21}&a_{22}&\hdots&a_{2n}\\
\vdots&\vdots&\ddots &\vdots\\
a_{m1}&a_{m2}&\hdots&a_{mn}
\endmatrix
$$
```

If instead you want

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \hdotsfor{4} \\ \hdotsfor{4} \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

you type

```
$$
\matrixp \format \c&&\;\c\\
a_{11}&a_{12}&\hdots&a_{1n}\\
a_{21}&a_{22}&\hdots&a_{2n}\\
\dotsfor 4\\
\dotsfor 4\\
a_{m1}&a_{m2}&\hdots&a_{mn}
\endmatrixp
$$
```

You have to tell \dotsfor how many columns it should put dots in for (since \matrix doesn't even know for sure how many columns there are going to be until it has read the final row).

A construction like

$$f(x) = \begin{cases} x+1, & \text{for } x > 0 \\ x-1, & \text{for } x \le 0 \end{cases}$$

can be constructed from \matrix using \left\{ and \right. but there's also \cases to do it all for you:

```
$$
f(x)=\cases x+1, & \text{for $x>0$}\\
           x-1, & \text{for $x\le0$}\endcases
$$
```

A \matrix can be used inside another \matrix or anything similar. For example, you can get the following matrix of parenthesized matrices

$$\begin{matrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\[20pt] \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} & \qquad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \end{matrix}$$

by typing

```
$$
\def\1{\matrixp 1&0\\0&1\endmatrixp}
\def\2{\matrixp 0&1\\1&0\endmatrixp}
\def\3{\matrixp 1&1\\1&0\endmatrixp}
\def\4{\matrixp 1&0\\1&1\endmatrixp}
\matrix\format \c\qquad&\c\\
\1&\2\\
\vspace {20pt}
\3&\4
\endmatrix
$$
```

Here we've used \vspace to specify extra vertical space between lines.

Dots. For things like

$$x_1 + \cdots + \|(a, \ldots, b)\| + (y_1 y_2 \cdots y_n) + \int \cdots \int f(x,y)\,dx\,dy$$

just use `\dots` to get the dots:

```
$$x_1+\dots+\|(a,\dots, b)\|+ (y_1y_2\dots y_n) + \int\dots\int f(x,y)\,dx\,dy$$
```

`\dots` figures out what kind of dots to give on the basis of the next symbol, so it probably won't give you the right kind of dots if you type something like

```
$$x_1 + x+2 + x_3 +\dots$$
```

So as a last resort, use `\dotsb`, the dots $\cdots$, together with the proper spacing, that go between binary operators and relations, like $+$ and $=$; or `\dotsc`, the dots and spacing that go before a comma; or `\dotsj`, the dots between juxtaposed things; or `\dotsi`, the dots between integral signs. If you do something weird (even though perfectly legitimate) you may confuse `\dots` so thoroughly that you'll just end up with an incomprehensible error message. Resort to specific `\dots`... in that case also.

Accents. The accents in math require different names, and there are some more of them:

`$\dot x$`	$\dot x$
`$\dotii x$`	$\ddot x$
`$\dotiii x$`	$\dddot x$
`$\dotiv x$`	$\ddddot x$
`$\hat x$`	$\hat x$
`$\check x$`	$\check x$
`$\tilde x$`	$\tilde x$
`$\breve x$`	$\breve x$
`$\vec x$`	$\vec x$
`$\bar z$`	$\bar z$

The choice between $\bar z$ (`$\bar z$`) and $\overline z$ (`$\overline z$`) is one of taste.

There are also variable size versions of `\hat` and `\tilde`, called `\widehat` and `\widetilde`:

`$\widehat x,\widetilde x$`	$\widehat x, \widetilde x$
`$\widehat {xy},\widetilde {xy}$`	$\widehat{xy}, \widetilde{xy}$
`$\widehat {xyz},\widetilde {xyz}$`	$\widehat{xyz}, \widetilde{xyz}$

Another way of accenting a long formula is to put the accent as a superscript to the whole formula in parentheses:

$$(x_1 + x_2 + \cdots + x_n)^{\sim}$$

You can't type `^\tilde` because `\tilde` isn't a symbol, it's an instruction to put an accent on something. But there's `\tildesymbol` which is a symbol:

```
$$(x_1+x_2+\dots+x_n)^\tildesymbol$$
```

Similarly, there's `\hatsymbol`, `\vecsymbol`, `\barsymbol` and corresponding symbols for all the dot accents. There also should be `\checksymbol` and `\brevesymbol`, but I forgot to make them, so they'll come later.

Oversetting and Undersetting. To get something like

$$\underset{\alpha\beta}{\mathbf{M}}$$

type

```
$$\underset \alpha\beta \to {\bf M}$$
```

(The underset formula is between `\underset` and `\to`, so doesn't have to be in braces even if it's more than one symbol.) Similarly, you can `\overset` something.

You can also `\underbrace` or `\overbrace` something:

`\underbrace{x_1+x_2+\dots+x_n}` $\underbrace{x_1+x_2+\cdots+x_n}$

And to get something like

$$\underbrace{x+\cdots+x}_{n\ \mathrm{times}}$$

you don't have to `\underset` and `\underbrace`; you can simply use `\undersetbrace`:

```
$$
\undersetbrace n \rm\ times\to {x+\dots+x}
$$
```

Boxed formulas. `\boxed` makes a formula boxed. For example,

```
$$
\boxed{x\le y}\qquad\text{for all $x\in A$, $y\in B$}
$$
```

produces

$$\boxed{x\le y}\qquad\text{for all } x\in A,\ y\in B$$

Roots. To get

$$\sqrt[\alpha+\beta]{1+\frac{a}{b}}$$

you type

```
$$
\root \alpha + \beta \of {1+\frac ab}
$$
```

If you don't like the position of the root, you can `\uproot` it or `\rightroot` it. For example,

```
$$
\uproot 3\rightroot {-3}\root \alpha + \beta \of {1+\frac ab}
$$
```

gives

$$\sqrt[\uproot{3}\leftroot{-3}\alpha+\beta]{1+\frac{a}{b}}$$

The amount of `\uproot`'ing or `\rightroot`'ing is given in terms of an arbitrary unit, that you don't specify.

Phantoms, Smashes, Etc. As the old "Joy of TEX" points out, there are numerous nefarious tricks that one can play with `\phantom{...}`, which produces an "invisible symbol" that takes up just as much room as the formula .... Even more useful is `\vphantom{...}`, which takes up no horizontal space but sticks just as far above and below the line as ..., and there is also `\hphantom{...}`, which takes up no vertical space but as much horizontal space as .... You can do even more "eye-balling" tricks now that AMS-TEX has `\smash{...}`. This leaves the formula ... in place but makes TEX think that it doesn't stick above or below the line at all! And `\topsmash{...}` just smashes the part above the line, while `\botsmash{...}` smashes the part below. There's still `\shave`, a specialty discussed in the old "Joy of TEX", but it's much easier to use now: Just say

```
\shave{\sum_{i=1}^Na_i}
```

to shave off any excess space above the formula

$$\sum_{i=1}^N a_i$$

There's also `\topshave` and `\botshave`, as before.

Errata

Page 3, line –5, Change `\Mr.@Jones` to `\Mr.~Jones`.

Page 5, line 3, Change "This will work for 'qspace" or ... " to "This will work for '" or ... ".

Page 6, line 14, `\guidelinegap⟨dim⟩` needs braces: `\guidelinegap{⟨dim⟩}`.

Page 6, line –12, Replace "amstex" by the AMS-TEX logo.

Page 13, line 14, "`\ssize`, since ... " should be "`\sssize`, since ... ".

Page 16, lines –6–end of page,

$$\begin{aligned} Z &= (X+Y)(A+B+C+D+E+F \\ &\qquad +G+H+K+L+M+N) \end{aligned}$$

which we got by typing

```
$$\align Z&=(X+Y)(A+B+C+D+E+F\\
    &\qquad +G+H+K+L+M+N)\endalign
$$
```

is really supposed to be

$$\begin{aligned} Z=(X+Y)&(A+B+C+D+E+F \\ &\qquad +G+H+K+L+M+N) \end{aligned}$$

which we got by typing

```
$$\align Z=(X+Y)&(A+B+C+D+E+F\\
    &\qquad +G+H+K+L+M+N)\endalign
$$
```

* * * * * * * * * * *

Advertisements

* * * * * * * * * * *

* * * * * * * * * * *

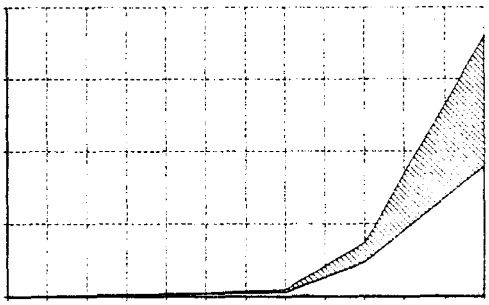
IMAGEN Corporation Intelligent Page Printer Systems

TEX Support

- Since 1981, and the introduction of IMAGEN's IMPRINT-10 non-impact page printer, we've supported the TEX formatting system.
- IMAGEN now supports TEX82, full graphics, and various other applications.
- The numerous TEX installations IMAGEN has across the country allows us to offer field tested and proven software for your TEX applications.

Quality Output

- IMAGEN now supports TEX (and other applications) on non-impact page printing systems varying in resolution from 240 to 480 dots per inch.
- This advertisement was typeset using the TEX formatting system and output on our 480 dot per inch printer.



Interested?

- We'd like the opportunity of discussing your specific applications with you in more detail. Simply fill out the coupon below and return it to IMAGEN at the address indicated. If you'd like, give us a call at (415) 960-0714.

Name:______________________________ Phone:____________________

Address:____

____

Host/Operating System:____

Current Output Device:____

Return to: IMAGEN Corporation, 2660 Marine Way, Mountain View, CA 94043

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 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 should be notified of new TEX releases. This should be the TEX user, not 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 48 hours. 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 preferably 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.

TEX82 ORDER FORM

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

TEX generic distribution tapes (PASCAL compiler required):

__________ ASCII format __________ EBCDIC format

TEX distribution tapes in special formats:

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

__________ DEC 20/Tops-20 Dumper format __________ IBM MVS format

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** **

__________ TEX82 – $20.00 __________ Test Manual – $8.00

__________ WEB – $10.00 __________ TEXware – $8.00

__________ TEXbook (draft) – $20.00

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

TEX Users Group **Membership/Order Form**

Request for Information

The TEX Users Group publishes a membership list containing information about the types of equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX 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 TEX 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. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX 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:
  TEX Users Group
  c/o American Mathematical Society
  P.O. Box 6248
  Providence, Rhode Island 02940, U.S.A.

Name: ______________________

Address: ______________________

____

____

____

Mail to (if different): ______________________

____

____

____

____

QTY	ITEM	AMOUNT
	1984 TUG Membership/TUGboat Subscription – North America New (first-time): [ ] $20.00 each Renewal: [ ] $30.00; [ ] $20.00 – reduced rate if renewed before January 31, 1984	
	1984 TUG Membership/TUGboat Subscription – Outside North America* New (first-time): [ ] $25.00 each Renewal: [ ] $35.00; [ ] $25.00 – reduced rate if renewed before January 31, 1984	
	TUGboat Back Issues, $15.00 per issue, circle issue(s) desired: 1980 (v. 1) #1; 1981 (v. 2) #1, #2, #3; 1982 (v. 3) #1, #2; 1983 (v. 4) #1, #2 (after 12/31/83)	
	The Joy of TEX (revised preliminary edition, 1982, with AMS-TEX82 supplement) @ $10.00 each	
	TEX Lectures on Tape (see cover 3, Vol. 4, No. 2)	
	Max Díaz's *Fácil TEX* (TEX80 macro package supplement; revised 10/81) @ $8.00 each	

*Beginning in 1984, 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 TEX:

My installation can offer the following software or technical support to TUG:

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

Date:

Status of TEX: [ ] Being installed
[ ] Up and running since
[ ] Under consideration

Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Approximate number of users:

Computer system(s):

Please answer the following questions regarding output devices used with TEX
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.