



# TUGBOAT

THE TEX USERS GROUP NEWS LETTER

October 1980

VOLUME 1, NUMBER 1,  
PROVIDENCE RHODE ISLAND

USA

## **TUG Steering Committee**

**Donald Knuth, Grand Wizard of  $\TeX$ -arcana**

**Richard Palais, Chairman of the Steering Committee**

**Robert Morris, Secretary and Wizard of Macros**

**Samuel Whidden, Treasurer**

**Robert Welland, TUGboat Editor**

**Luis Trabb-Pardo, Vice Grand Wizard**

**Ignacio Zabala, Wizard of  $\TeX$ -in-Pascal**

**David Fuchs, Wizard of I/O and SAIL**

**Barbara Beeton, Wizard of Format Modules**

**Nicholas Allen, Wizard of TOPS-10**

**Arnold Pizer, Wizard of TOPS-10**

**Patrick Milligan, Wizard of TOPS-20**

**Michael Spivak, Wizard of  $AMS-\TeX$**

**Richard Zippel, Wizard of ITS**

**Hermann Zapf, Wizard of Fonts**

**Charles Howerton, Liaison with the  
National Bureau of Standards**

**Richard Friday, Liaison with Digital Equipment Corporation**

**Michael Bennett, Foreign Distribution of  $\TeX$**

The subject of mathematical printing has never been methodically treated, and many details are left to the compositor which should be attended to by the mathematician. Until some mathematician shall turn printer, or some printer mathematician, it is hardly to be hoped that this subject will be properly treated.

Augustus de Morgan  
*Penny Cyclopaedia* (1842),  
on 'Symbols'

# TUGBOAT

THE  $\text{\TeX}$  USERS GROUP NEWSLETTER  
EDITOR ROBERT WELLAND

VOLUME 1, NUMBER 1  
PROVIDENCE

RHODE ISLAND

OCTOBER 1980  
U.S.A.

Well ... the original version of  $\TeX$  was written in SAIL, so doesn't that mean the TUG Newsletter should be called TUGboat? Hmmm!

\* \* \* \* \*

## EDITOR'S COMMENTS

Robert Welland

The  $\TeX$  Users Group (TUG) met at Stanford University in February of 1980 (see Robert Morris' minutes of this meeting, p. 12) and among other things decided that the group would publish a newsletter to assist the distribution of  $\TeX$ pertise. Sam Whidden of the AMS volunteered the services of the Providence group to help make this decision a reality.

$\TeX$ ,  $\text{AMS-}\TeX$  and **METAFONT** are software tools which will make the processing of scientific documents less painful, less expensive and more rapid. Authors working at editing terminals will find correcting easier and they will be spared much of the pain of proofreading (see Richard Palais' "Message From The Chairman" on p. 3). Ellen Swanson's article "Publishing &  $\TeX$ " on p. 7 gives us a glimpse of the steps involved in getting a paper from the author's desk into print and indicates the benefits and savings the AMS expects when  $\TeX$  software tools become widely used.

TUG, the  $\TeX$  Users Group, like other software users groups, was formed so that its members could share with one another the skills they have developed using  $\TeX$ .

TUGboat, an energy-efficient conveyance powered by SAIL and a newly-found Pascal driver, is the name of the TUG newsletter and will be used to ship a cargo of information between its members. Among other things, it will provide the information necessary to support the implementation of  $\TeX$  and  $\text{AMS-}\TeX$  software. This support burden has been shouldered by Don Knuth and his colleagues at the Stanford University Computing Center. To the greatest extent possible, we must now carry this load so that these very kind people can return to their principal task of developing Computer Science.

This first issue of TUGboat contains several introductory articles, which we hope will inform and inspire our non-TUG friends in the AMS to board TUGboat so that they too can begin to experience the JOY of  $\TeX$ . This issue also contains several technical articles (David Fuchs' article on DVI files, p. 17, and Terry Winograd and Bill Paxton's Index Creation Package, Appendix A) which non- $\TeX$ perts will find difficult to understand.

This brings us to a problem which the newsletter must confront. For some time to come, the mathematics community will contain people not conversant with  $\TeX$ , whom we at TUG will want to join our ranks; it will contain  $\TeX$ perts at various levels of development and systems experts charged with the job of bringing up  $\TeX$  on a host of machines having many different operating systems and driving a large array of output devices.

This audience is too broad; so in the future we will assume that our readers have at least the acquaintance with  $\TeX$  which will come from reading this issue of TUGboat and from reading Don Knuth's excellent book  $\TeX$  and **METAFONT: New Directions in Typesetting**. With this background, a reader will be able to understand nearly everything in TUGboat with the exception of the parts directed to the systems programmers. The average  $\TeX$  user will be able to get by quite well without understanding these specialized parts.

TUG has been in existence for a short period of time and so its needs are only partially delineated and for this reason this is also the case for TUGboat.

Our ship has compartments in its hold for:

(a) *General Delivery.*

This compartment will carry expository articles, minutes of TUG meetings and other material not devoted to specific technical issues.

(b) *Interface Software.*

This compartment will carry descriptions of interface software. It will tell who is developing what, list what interface software exists, where it can be obtained and give reports from various test sites.

(c) *Warnings and Limitations.*

Barbara Beeton of the AMS suggested this compartment and makes its first entry. We were tempted to call it Murphy's room because it will carry descriptions of collections of steps which, if carried out in order, will produce undesirable results.

(d) *Macros.*

This compartment will carry descriptions of macros; it will also carry descriptions of various helpful programs related to using  $\TeX$ .

(e) *Bugs.*

The name of this compartment is short and self explanatory. We hope, unrealistically, that the contributions to it will be few and far between. There have been almost no bugs reported recently in the SAIL version of  $\TeX$ . However, we will undoubtedly experience difficulties with the new software. When bugs are found, please send the information to the newsletter so that it can be published immediately.

(f) *Questions and Answers.*

This compartment will contain questions which

are of general interest and hopefully eventually their answers.

(g) *Letters.*

The content of this compartment is obvious.

(h) *Miscellaneous.*

All odd-shaped packages which do not fit into any of the above compartments will be placed in this one.

How well TUGboat sails will depend on the support it receives from its crew. At this stage it is an experimental ship. The Steering Committee of TUG and the editor encourage you to make contributions, comments and criticisms. We believe that, with this collective hand on the TUGboat wheel, everyone will have a pleasant trouble-free trip into the friendly land of  $\TeX$ arcana.

Robert Welland  
Northwestern University  
Evanston IL 60201

\* \* \* \* \*

General Delivery

\* \* \* \* \*

MESSAGE FROM THE CHAIRMAN

Richard S. Palais

An Apology ...

The organizational meeting of TUG took place last February and at that time the first mailing of the newsletter was at least informally promised within a couple of months. And here it is October. Perhaps the major cause for delay was the failure of the Chairman of TUG to get his share of the writing of the newsletter done in a timely fashion. It would be easy enough to blame this on preoccupation with "other matters", but that only pushes back the question to why the newsletter was not given higher priority—and the answer quite honestly was a feeling that the initial newsletter should not predate by too long the actual release of the Pascal version of  $\TeX$ . The latter happily now really does seem imminent (for more on  $\TeX$ -in-Pascal see the report on page 18). Finally then it is perhaps as good a place here as any to say a little about why the public release of Pascal  $\TeX$  has been delayed.

The most important cause seems to have been a technical one, the lack of a complete, standardized definition of Pascal. The resultant need to take careful account of possible compiler inconsistencies showed up (and aborted) the initial attempt at Pascalization. The second version of  $\TeX$ -in-Pascal

has made a careful segregation of the universal, machine-independent features from those parts that must be implemented separately for each different machine architecture family and operating system, and this approach now seems to have been quite successful. This illustrates a point worth noting. If the same problem had arisen while  $\TeX$  was being developed under contractual time constraints by a commercial software house, in all probability a quick technical fix would have been attempted. In fact of course,  $\TeX$  was developed at one of the world's foremost centers of artificial intelligence research by an outstanding master of the art of computer programming and his students. The goal was not only the best possible typesetting system but also a design and implementation that would exemplify and indeed be a paradigm of the state of the art in the development and documentation of a complex software system meant to run in many diverse architectures and environments. The original SAIL implementation of  $\TeX$  has proved to be an unusually bug-free and stable system that has now been running very successfully for well over a year on five operating systems (TOPS-10, TOPS-20, SU-AI, ITS, Xerox PARC) driving as many different output devices (XGP, Versatec and Varian electrostatic printers, Alphatype CRS typesetter, Xerox Dover printer). The Pascal version has now passed most of its basic tests and there is every reason to expect of it the same high quality performance demonstrated by its predecessor.

... and an Appreciation

I am sure many early  $\TeX$  users have had the experience of watching with surprised delight as their first pages came off the printer and thinking, "Did I really set that myself!" And on reflecting, we knew that while we did, and while we had reason to be proud of our new skill, in a very deep sense Don Knuth had been and would continue to be there at our side through his algorithms, his carefully designed user interface, and delightfully written  $\TeX$  manual. I am sure that kind of silent, smiling appreciation is all the thanks Don is looking for, but I would be remiss in my role as TUG chairman if I did not express in a more public way the tremendous debt of gratitude we all owe to him for the tireless and selfless effort he has put into the task of making  $\TeX$  a freely available system. Despite his consummate programming skill, creating software is not what Don considers a high priority use of his time. Research and writing papers in mathematics and computer science is much more challenging for him, but he even rations the time spent on this in

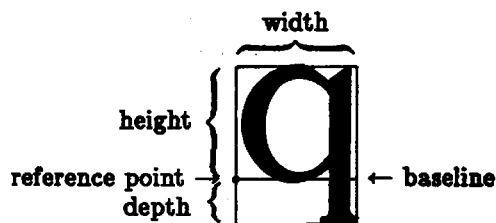
order to have time left for what he considers his primary life work, the completion of his monumental *Art of Computer Programming*. Don originally created  $\TeX$  and **METAFONT** to insure that the later volumes of *ACP* would conform to the same high standards of typography that he demanded of volumes one to three. He had not foreseen the enthusiasm that greeted the early, primitive versions of his typesetting programs, but happily for all of us Don was willing to take up the challenge this presented, and now the fruits of his efforts as well as the efforts of his many students are becoming available for us all to enjoy.

### An Introduction to $\TeX$ and **METAFONT**

This is the first issue of TUGboat, the newsletter of TUG (the  $\TeX$  Users Group). Presumably most readers will be TUG members and familiar with  $\TeX$ , so for the most part the newsletter will assume a reasonably high level of knowledge of  $\TeX$  basics. However, this issue of the newsletter will also be sent out to those with little or no knowledge of  $\TeX$  who write to TUG asking for basic information. For the benefit of these readers we shall give below an elementary description of  $\TeX$  and **METAFONT** together with pointers to the major source of detailed information about them.

$\TeX$  is a "typesetting" program. Like most computer programs it transforms information supplied to it (an "input file") into an "output file". In the case of  $\TeX$  the input file consists of the text to be typeset together with certain local formatting hints (e.g. `\par` to indicate a new paragraph) and occasional local font information (e.g. `\bf` or `\it` to indicate that what follows should be either boldface or italic, respectively). Note however that serious global questions of formatting and font choice, such as page width and length, interline spacing, just where on a page to place titles, author names, page numbers, chapter headings, footnotes, etc., and in what fonts these should be set are specified in a parameter file usually defined by a book or journal design expert, so that the average  $\TeX$  user rarely if ever has to deal with these finicky details. The input file is usually prepared at a computer terminal using an editing program, though not necessarily using the same computer that will process it with  $\TeX$ . The  $\TeX$  program itself can run on any computer with a Pascal compiler and sufficient memory. (Actually, "sufficient" here means pretty big;  $\TeX$  probably won't run very efficiently in much under 300K bytes, although if you have a virtual memory system and plenty of time you might get by with 128K. Anyway, don't expect to use  $\TeX$  at home on your TRS-80

soon!) Running on a mainframe computer,  $\TeX$  will prepare its output file (a so-called DVI or "device-independent" file) at the rate of about one printed page worth every few seconds. Before describing the DVI file in any detail it is necessary to explain a little about how  $\TeX$  conceives of characters and fonts of characters. To  $\TeX$  each character is merely a rectangular box with a horizontal "baseline" through it. The box is completely described by its height and depth (distance from baseline to top and bottom of the box) and its width. The position of a character box on a page is specified by giving the  $x$  and  $y$  coordinates of its "reference point" relative to the upper left-hand corner of the page (the reference point is the intersection of the baseline with the left-hand edge of the box).



$\TeX$  does not "know" the shape of a character that will eventually go inside the box (in fact the shape needn't even fit inside the box). In any given job,  $\TeX$  is able to work with 64 different fonts of 128 characters each. Aside from the size of each character box,  $\TeX$  has other information available to it concerning a font: so-called kerning, ligature and italic correction information. Using this information, the formatting and font information in the parameter file, and some rather sophisticated algorithms for justification and page makeup,  $\TeX$  lines up the character boxes from the input text into properly justified paragraphed horizontal rows ("lines of type") and arranges these into pages.  $\TeX$  finishes up by describing each page it has made up as a sequence of "records", one record for each character on each page. The record for each character consists of its font number (0-63), its place in the font (0-127), and its  $x$  and  $y$  coordinates on the page. The DVI file is just the collection of these records coded according to a specified protocol.

But where, you ask, is the printed page? The answer is that once the DVI file is in hand, producing the actual hard copy is a relatively simple matter, although one that depends very sensitively on the nature of the particular output device that will be used. For each output device that is to be interfaced to  $\TeX$  a software "driver" must be written that will transform the device-independent DVI file into

a device-specific format, and then actually give instructions that get the ink onto the page. Normally this driver program is not run on the mainframe hosting  $\TeX$  but rather on a small microcomputer either off-line (perhaps physically inside the output device) or else as a peripheral to the host mainframe and attached to it over a serial line. Such drivers have already been successfully written for a variety of devices including electrostatic printers (Versatec, Varian), CRT typesetter (Alphatype), laser printer (Xerox, Canon), and even experimentally for matrix and daisy-wheel printers. Experience seems to indicate that developing such a  $\TeX$  interface for a new output device takes on the order of one man-month.

But isn't a new output device going to require that all the fonts  $\TeX$  will use on that device be recreated according to the protocols which that device understands? And isn't that very expensive? This is where **METAFONT** comes in. **METAFONT** is both a language for the creation and specification of type faces and a program for creating the computer files necessary to describe a font of type to  $\TeX$  and to a  $\TeX$ -output device interface. Of course, for any new output device a (relatively simple) program must be written to interface **METAFONT** to that device. But once this is done the large and growing library of fonts in **METAFONT** format become immediately available to the new device.

The basic source of complete documentation on  $\TeX$  and **METAFONT** is  $\TeX$  and **METAFONT: New Directions in Typesetting** by Donald E. Knuth, published jointly by the American Mathematical Society and Digital Press. Send orders to:

Digital Press  
Dept. AMS, Educational Services  
Digital Equipment Corporation  
12 A Esquire Road  
North Billerica, MA 01862

The price is \$12.00 in the U.S.A. on prepaid orders, with individual members of the AMS entitled to a 50% discount. The book is in three parts. Part I is a reprint of Knuth's 1978 Gibbs Lecture to the AMS. In it he gives some of the history of mathematics in the service of typesetting, and then goes on to describe some of his design philosophy and some of the mathematical problems he met in the development of  $\TeX$  and **METAFONT**. Parts II and III are respectively the  $\TeX$  and **METAFONT** manuals.

Of course there is a great deal more in the way of documentation to  $\TeX$ : program documentation and installation guides, font catalogues, "header" files to format specific types of documents appropriately, macro packages to create indexes or to ease the labor of setting complex mathematics. But these will be

pointed to elsewhere in this and future editions of the TUG newsletter. That indeed is one of the main purposes of TUGboat.

Well, so much for the broad outline of the  $\TeX$ -**METAFONT** system. But to round out the picture and give a more complete perspective I feel that there are further comments that should be added.

(1)  $\TeX$  was designed not just as a typesetting system, but as one with a particularly good built-in facility for setting mathematical text ("penalty copy" in the words of the old hand compositors). What makes mathematics so hard to set is not the large number of special symbols, but all the superscripts and subscripts, all the equations that must be aligned, and all the many and varied complex two-dimensional diagrams. Even the problem of designing a good mnemonic language for specifying the various intricacies of mathematical text is non-trivial. The solutions that are evolving out of  $\TeX$  and systems built on  $\TeX$  could by their wide acceptance develop into a *de facto* standard for the "linearization of mathematics".

(2)  $\TeX$  has built into it a very powerful "macro facility". What this means at a very simple level is that a user can give a name to a complex and frequently occurring chunk of text to avoid having to constantly retype it. At a more sophisticated level it means that individuals or groups of individuals can easily customize  $\TeX$  to best suit their special needs *without* changing the basic computer code. The importance of this latter point makes it worth stressing.  $\TeX$  itself has proved to be an exceptionally robust and bug-free program. The intention is to keep it very stable, making only the most essential changes to the program, and these only very infrequently. But experience has shown that any multi-user, general program must either be easily modified or die. Since  $\TeX$  is in the public domain, there is no way to prevent it being modified to suit the varied tasks of its users. If these modifications are made at source code level the result will soon be a chaos of mutually incompatible programs that would make maintenance an impossible nightmare. Moreover, a great advantage would be lost. It would not be possible to prepare and test a  $\TeX$  input file at one site and then send it by tape or computer net to be output at another site. (To help forestall this disease the AMS, with Knuth's permission, has applied for trademark protection for the  $\TeX$  logo and will prevent its use to describe any unauthorized modification of  $\TeX$ .) Fortunately the  $\TeX$  macro facility permits users to treat  $\TeX$  as though it were written in a so-called "threaded" programming language, and to modify it to their hearts' content

without introducing any incompatibilities between their underlying implementation of  $\text{\TeX}$  and implementations at other sites. All these users have to do if they want to interchange documents written in  $\text{\TeX}$  input format is to be sure that together with their input files they send along their macro definition files. A number of quite sophisticated macro packages have already been written, and one is described in detail elsewhere in this newsletter. The American Mathematical Society is preparing a customized version of  $\text{\TeX}$  called  $\text{\AMS-TeX}$  to simplify the production of the Society's journals and also to permit research mathematicians, if they wish, to actually set their own papers as preprints exactly as they will appear in final published form. Michael Spivak, one of the main designers of  $\text{\AMS-TeX}$ , and the author of its manual, *The Joy of  $\text{\TeX}$* , reports on it later in the newsletter.

(3) All of this doesn't answer one of the main questions a potential  $\text{\TeX}$  user will be interested in. How difficult is it to learn to use  $\text{\TeX}$  and then to actually use it to create, say, a technical paper? All my evidence is anecdotal, coming from my own personal experience or that of my friends, and it all says that the learning experience is remarkably easy, pleasant and fast. Moreover, most authors get very excited about their new skill and find they quickly develop a much more highly critical eye for typographical niceties. One anecdote is too good not to relate. G. G. Emch, a physicist at Rochester, approached Arnie Pizer of the Rochester mathematics department to help him  $\text{\TeX}$  a paper he needed in a hurry. Pizer was the driving force behind getting  $\text{\TeX}$  running on the TOPS-10 system at the University of Rochester and is very  $\text{\TeX}$ -knowledgeable. His physicist friend, on the other hand, had never even used a computer terminal himself before. Arnie spent one day teaching him how to use the terminal, the operating system, the editor, the basics of  $\text{\TeX}$  and the experimental version of  $\text{\AMS-TeX}$ . Two days later, believe it or not, Dr. Emch had typeset a highly creditable four-page paper by himself and flew off with it to a conference in Europe.

#### $\text{\TeX}$ as more than a Production Typesetter

$\text{\TeX}$  is such a pleasantly useful daily tool for production typesetting that it is easy to lose sight of some of the more long-range possibilities it opens up. I would like to comment here on those I consider most important.

Anyone who looks at the process for producing scientific journals (and most other books and magazines for that matter) must be struck by

the tremendous wastefulness of human time and effort it entails. After the author in collaboration with technical typist, 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 (from an information content point of view) finally back in the form in which it was sent to be composed. This activity of adding errors and then removing them is actually responsible for half the cost of producing the journal. That is, if authors could present the journal with acceptable camera-ready copy, the cost of journals could literally be cut in half! (Some journals for this reason have taken to using the typescript as camera copy—but for many of us that would never be acceptable.) Now  $\text{\TeX}$  actually gives us the possibility of realizing this economy and at the same time putting the responsibility for beautiful composition where it belongs: with the author. Almost all of the copy-editor's function can be automated into so-called "header" files which contain all of the journal-specific design, font and format information. This is not to say that copy-editing does not call for the exercising of taste and judgment. It does, but this part of the job can easily be transferred to the author too. The ultimate goal of the American Mathematical Society's  $\text{\AMS-TeX}$  system is to make it possible for willing authors to avoid the outdated foolishness described above and typeset their papers themselves (perhaps with the aid of a technical typist) at the preprint stage.

If we go ahead a little further in time we can foresee a development which I call the all-electronic, save-the-forest library. It is rapidly becoming the case that many (perhaps most!) articles in a particular library copy of a scientific journal are never read. Those that are read are apt to be photocopied out of the library copy to be read at leisure at home. The costs of printing, binding and mailing the journal make up the other half of the journal's cost mentioned above. Why not save that cost too by simply having the journal in magnetic storage at some (or several) central locations. It will soon be cheap and convenient to peruse such a magnetic journal on a computer terminal in the comfort of one's home or office. And of course if a hard copy is desired, it can be produced when and where needed in moments. Now a little thought will show that (because



of differences between terminals and the exorbitant memory it would require) it will not be feasible to store the journal as a high-resolution raster image. The natural form of storage will be a source file for TeX or some similar typesetting program, or perhaps as the DVI output file of such a program.

Finally, there is a use to which TeX or some system growing out of TeX could be put which is in the nature of a spinoff. There is rapidly coming into existence a large number of on-line bibliographic data bases. These data bases must of necessity be stored in a linear character-by-character manner. For ordinary text this poses little problem. But when there are mathematical formulas involved there are enormous problems caused by the lack of any standard linearized method to describe what is often really two-dimensional text. Of course, if TeX gains sufficient acceptance it could become a *de facto* standard for linearized mathematics. Indeed, it is possible that ultimately the complete AMS-TeX version of the Mathematical Reviews input files will be available on-line in commercial data base retrieval systems.

### The Role of TUG

A look at the TUG roster will show that a surprisingly large number of people have written in expressing an interest in TeX and asking to belong to TUG. What is the appropriate role for us to play? It was clear from the discussion at the organizational meeting that there was anything but unanimous agreement on this point. It would seem appropriate to me that we should wait about a year after the Pascal version of TeX is actually "in the field" and then have a second face-to-face meeting to discuss our experiences, problems, and needs. In the meantime, TUG will function as a clearing-house for information concerning TeX. Initially the main problems will involve getting Pascal TeX installed on various architectures and operating systems. Here it will be particularly important to avoid many different groups reinventing the wheel. Once the basic installation problems are settled, there will be a certain amount of elementary question-answering needed from TUG until expertise builds up in the field. Eventually this function of TUG should evolve into one of providing names of experts for TeX consulting and trouble-shooting, and TUG will no doubt gradually assume other functions such as a clearing-house for the exchange of documented TeX macro packages. Ultimately TUG will become exactly what we its members make it. And that is what we can look forward to discussing at our next

meeting. Until then,

Happy TeX-ing  
Richard S. Palais

\* \* \* \* \*

### PUBLISHING & TeX Ellen E. Swanson

A published book or journal originates as an idea in the mind of the author, is put into manuscript form, and sent to an editor for review. Upon acceptance for publication it is copy edited, set into type, proofread, corrected, paged, and finally assembled and sent for printing and binding. Many hours and dollars are spent on each page and chapter of every published book or journal. If the material is scientific, or mathematical, the copy editing and composition is generally more expensive than for ordinary text. New technology utilized over the past few years has reduced the expensive composition costs. Tight budgets have resulted in some economies being implemented in the editorial aspects, for instance by eliminating a proofreading check. The purpose of this article is to acquaint the author with the main steps of the publication process and then to indicate how TeX may help eliminate some of the publication costs.

Current systems using computer-assisted composition have reduced the cost of composition dramatically, but not for editorial functions such as copy editing or proofreading. Curiously enough the cost of composing a page of mathematics at the American Mathematical Society is about the same in 1980 as it was in 1969; new technology has negated the inflation factor. On the other hand, the hours spent in copy editing, proofreading and other editorial tasks have not been reduced significantly, with the result that these costs per page have more than doubled.

The use of TeX could provide savings for the publisher in both composition and editorial functions. To understand more fully how this saving can be accomplished there follows a list of the steps involved in publishing and an indication of the time and/or dollars that are spent. Then it will be pointed out where TeX can produce savings. For the sake of simplicity in this paper it will be assumed that an article is being written for publication in a journal, and that the journal is a scientific one containing some mathematics. Essentially the same process is used for publication of a book. Obviously, if the publication does not contain mathematics, some of

the steps will be simpler and any references to mathematics can be ignored.

**THE MANUSCRIPT.** The first step in the publication process is that the author conceives an idea, and puts it on paper by handwriting or typing. The original manuscript is doubtless revised once, twice or even more times before the author is happy with the result. Then it must be sent to an editor, who may in turn send it to another one or two persons (often called referees) for their recommendation. Let us assume it is accepted for publication.

**ACCEPTANCE TO PRINTED COPY.** The editor usually sends a manuscript to the editorial office of the journal for processing. This office is usually involved in all of the steps from the time the manuscript arrives to when pages are sent to the printer ready for negatives to be made for offset printing.

**Logging in.** The manuscript has to be acknowledged and other procedures instituted such as setting up files, and obtaining copyright permissions.

**Copy editing.** A manuscript should be read line by line to be sure that it is legible, that the grammar and spelling are correct, and that the author has maintained a consistent style that conforms to that of the journal in which the article is being published. For a scientific paper editors also check that notation is available and clear, and in mathematics papers that the displays are in a form suitable for keyboarding and printing. Artwork must be in a form required for printing and reduction factors determined, if they are needed. Time spent copy editing is important; it saves money in the long run because it cuts down on the number of changes and corrections required on the proof.

**Composition.** The kind of composition used today is almost as varied as the number of publishers. Whatever the method, the manuscript must be keyboarded and then camera copy produced with the use of some kind of hardware. There is usually a check for errors and corrections must be keyboarded; proof must be produced and then sent to the editorial office and to the author.

**Proofreading.** This proof must be read carefully, whether by an editorial assistant or by the author; a more accurate publication is produced if it is proofread by both. If mathematics is involved, bad breaks at the end of lines, or poorly set up displays, need to be marked for rearrangement.

**Corrections.** All corrections have to be coordinated and returned to composition for keyboarding. Corrected proof is then returned to the editorial office for checking. If the luck is bad, there will be additional errors and the correction process will have

to be repeated.

**Page makeup.** Fully corrected proof is ready for its final paging. If it is in galleys, they must be cut into page lengths. All pages must be checked for proper length and for bad breaks between pages. Then any changes in paging must be both executed and checked. Running heads and page numbers must be inserted.

**Makeup of an issue.** Once the pages are ready the book must be carefully put together. Covers, title pages, information pages, prefaces, contents, and indexes must be coordinated; articles or chapters have to be ordered, and running heads and page numbers checked.

**To the printer.** Checks must be made that the pages are correctly ordered, that the instructions to the printer are clear, both on the proof and in the accompanying letter. Binding instructions must also be included.

**COSTS OF PUBLICATION.** It is rather difficult to put costs into dollars because of differences in salary and in the overhead rate for various publishers. Some of the cost items below are, therefore, given in hours rather than in dollars. It should be remembered, however, that, if you know the salary of a keyboarder or copy editor is \$6.00 an hour, it cannot be assumed that the cost of a half hour of work is just \$3.00. In addition to salary there are benefits, rent, and other types of overhead. The actual "rate" is apt to be two or three times the hourly wage. I remember a young woman, who was working for me as an editorial assistant, becoming very irate when she saw the budget of the project on which she was working—it seemed we were not paying her the amount of money that was budgeted for the work she was doing. Here is a list of the various steps with comments regarding time and costs. These figures assume a page has a printing area of about 5"×8" and 6 lines of 10 point type per inch.

1. **The manuscript.** Only an author can know the time and effort that is entailed in the writing and an editor the hours involved in reviewing the paper.

2. **Logging in and other clerical functions.** 30 minutes to 2 hours per paper.

3. **Copy editing.** 10 to 15 minutes per page. It depends greatly on the condition of the manuscript, whether or not rewrite is necessary, and the type of material being published.

4. **Composition.** 30 to 60 minutes per page. Depends greatly on the content, that is whether it is strictly words or if there is scientific notation involved. Computer or other hardware charges are involved in addition to personnel.

5. **Proofreading.** 6 to 20 minutes per page.

6. *Keyboarding corrections.* 3 to 10 minutes per page.

7. *Checking corrections.* 3 to 6 minutes per page.

8. *Page makeup.* 3 to 12 minutes per page. Depends a great deal on whether it is straight text or whether displays, figures or tables make page breaks difficult.

9. *Makeup of an issue.* 1 to 3 minutes per page.

In all, these minutes add up to anywhere from  $1\frac{1}{4}$  to nearly 3 hours per page. In general, one can assume that editorial functions (Steps 2, 3, 5, 7, 9) take about an hour a page, while keyboarding, page makeup and other composition related functions (Steps 4, 6, 8) tend to take about one hour and a half of personnel time.

At the American Mathematical Society we do all our own editorial work and we do our own composition by a computer-assisted system, but use an outside phototypesetter. The editorial functions take approximately one hour. Composition costs at the present time are \$18.00 to \$24.00 per page, including both department and Society overhead; composition takes about  $1\frac{1}{4}$  hours of personnel time plus computer and outside service charges of about \$4.00.

**ROLE OF T<sub>E</sub>X.** T<sub>E</sub>X can be useful to both the author and the publisher. If an author learns the T<sub>E</sub>X codes, or has a secretary who knows them, a manuscript can be keyboarded by use of T<sub>E</sub>X and revisions can be made while sitting at a terminal—only the lines being revised need to be rekeyboarded.

For the publisher there are two main ways in which T<sub>E</sub>X can cut costs.

(a) If a paper were submitted in the traditional manner some saving would result from using T<sub>E</sub>X for composition. The manuscript would be copy edited, keyboarded and all the other steps outlined above executed. However, page makeup would be streamlined because T<sub>E</sub>X not only can automatically divide a manuscript into pages, but has a built-in mechanism for lengthening or shortening pages to correct lengths. If page breaks between lines are not acceptable, they can be changed by a keyboarding direction rather than by hand stripping. These shortcuts could easily save a dollar or more per page.

(b) If an author were to produce the manuscript on the T<sub>E</sub>X system and submit a magnetic tape to a publisher who has facilities for converting the tape to typeset copy, the savings would be much more dramatic. Personnel for copy editing, composition, proofreading, and keyboarding of corrections (Steps 3, 4, 5, 6) would be almost entirely eliminated. If copy editing changes were needed to conform to house style, these could be made directly on the computer terminal and would thus be mini-

mal. Proof would be submitted to authors in pages set automatically by T<sub>E</sub>X. However, if the author makes changes or corrections, the cost would increase because it would result in extra keyboarding and checking; in addition the whole paper would need to be set in type again, or patches set and stripped onto the original pages by hand.

A saving of 80% of both composition and editorial costs could easily result if a paper were submitted on a T<sub>E</sub>X magnetic tape. If the T<sub>E</sub>X tape needs copy editing and/or corrections, the saving would likely still be 50% to 75% because copy editing, keyboarding and proofreading would be cut drastically.

It is unrealistic to assume that all articles in a publication will be submitted in T<sub>E</sub>X. Not all authors have the temperament or desire to learn the T<sub>E</sub>X codes and not all university departments will have a T<sub>E</sub>X expert. However, there would still be a considerable saving in the cost of publication if only part of the papers for a book or journal were submitted by authors on magnetic tape produced by the T<sub>E</sub>X system.

#### BIBLIOGRAPHY

1. *A manual of style*, 12th rev. ed., University of Chicago Press, Chicago and London, 1969, 546 pp.
2. Marshall Lee, *Bookmaking: The illustrated guide to design/production editing*, 2nd ed., R. R. Bowker, New York, 1979, 485 pp.
3. M. E. Skellin, R. M. Gay et al., *Words into type*, 3rd ed., Appleton-Century-Crofts, New York, 1974, 585 pp.
4. Ellen Swanson, *Mathematics into type*, rev. ed., American Mathematical Society, Providence, R. I., 1979, 90 pp.
5. Donald E. Knuth, *T<sub>E</sub>X and METAFONT: New directions in typesetting*, American Mathematical Society and Digital Press, Bedford, Massachusetts, 1979, 306 pp.

Ellen E. Swanson  
American Mathematical Society

August 13, 1980

## AMS-TEX: "A Very Friendly Product"

MICHAEL D. SPIVAK

This is a very brief introduction to AMS-TEX, the specialized version of TEX that is being created by the AMS. It is rather different from most of the other articles in this first newsletter. To read it, you don't have to know anything about TEX, nor do you need to know anything about computers, except how to make a computer file, typing on the keyboard of a terminal. Indeed, the whole point of AMS-TEX is that it will allow you to produce mathematics as beautiful as that in any journal without requiring any knowledge about computers or typesetting.

Suppose, to begin with, that you just want to produce ordinary text. To do this, you basically just have to be able to type. In fact, many of the typist's ordinary petty concerns are irrelevant, since TEX will take care of many details for you. For example, it doesn't matter how long you make each line (TEX will automatically arrange the lines into "justified" text, with all lines the same length), or how many spaces you leave after commas and periods (TEX will use printers' conventions for the spacing after punctuation); you can even leave extra spaces between words (TEX ignores the extra spaces).

You also don't have to worry about how big an indentation to leave at the beginning of a paragraph. In fact, any indentation you leave is irrelevant, since TEX ignores extra spaces; instead, you can tell TEX that a paragraph has ended by leaving a blank line (which you get by pressing the "carriage-return" key twice in a row); TEX then starts a new paragraph with the next text.

Another way you can tell TEX that a paragraph has ended is to type `\par`. The special symbol `\`, which appears on computer terminal keyboards, though not on ordinary typewriters, is used at the beginning of TEX's "control sequences". These, like `\par`, are special instructions to TEX, rather than material to be typeset. For example, `\it` tells TEX to set text in *italics* and `\bf` tells it to set text **boldface**. Control sequences are also used to name symbols that don't appear on the keyboard. For example, `\pi` stands for the symbol  $\pi$ .

This sounds pretty easy, but you may be worrying about other things. How do you know when you get to the end of a page? How do you center the title? How do you put the author's name under it? Etc. Well, the computer file for this article begins

```
\input preprn
\title \amstex: ``A Very Friendly Product``\endtitle
\author Michael D. Spivak \endauthor
```

The first line requested  $\text{\TeX}$  to print the article in a standard "preprint" style, in which the typeface, the width and length of a page, etc., are all determined. After this, there wasn't much to worry about— $\text{\TeX}$  figured out where to start new pages, etc. (If you don't like the typeface or page size, there's an easy way to change that, too. Moreover, you can just as easily request the format of some journal.) The next two lines of input should be nearly self-explanatory. They told  $\text{\TeX}$  what the title and author's name were— $\text{\TeX}$  then set these things correctly by itself (the special control sequence  $\text{\backslash amstex}$  was made to produce the output " $\text{\AMS-TeX}$ "). The only slightly surprising feature might be the quotation marks, which were built up from the single quote marks "''" and "''"; this procedure was built into  $\text{\TeX}$  because most computer terminals don't have left and right double quotes. There are a dozen or so special rules of this sort that you need to learn, but once you have learned them you can produce almost any sort of text, including footnotes and special symbols and diacritical marks ( $\text{\AE}$ ,  $\text{\acute{o}}$ , etc.).

Now how about mathematical text? Well, if you want to get an equation like  $y = x + 1$  within text, you simply type  $\text{\$}$  signs on either side of it. Then the  $y$  and  $x$  get set in italics, the spacing around the  $=$  and  $+$  signs are just right, etc. (Any spaces that you type in the input for the equation are completely irrelevant, so you can type equations as squashed together or as spread out as you like.) On the other hand, if you want the equation

$$y = x + 1$$

displayed, you just put  $\text{\$\$}$  on either side of it. Of course, sometimes you need more complicated things, like

$$\begin{aligned} y &= x + 1 \\ &= v + 2. \end{aligned}$$

For this you simply type

```
 $\text{\$\$}\backslash align y \&=x+1\backslash\backslash$   
 $\text{\&=v+2.\end{align} \math{\$}$ 
```

The  $\text{\&}$  signs tell  $\text{\TeX}$  which symbols get lined up, and the  $\text{\backslash\backslash}$  separates the lines. In practice, you'd probably type something like  $\text{\dots y\&=x+1\backslash\backslash\&=v+2\dots}$  all on one line; breaking the input into two lines makes it look nicer, but is irrelevant to the way  $\text{\TeX}$  sets it.

All this may sound too good to be true, and it probably is—the present (incomplete) version of the  $\text{\AMS-TeX}$  manual runs over 100 pages! The only way to really find out is to get a copy of the  $\text{\AMS-TeX}$  manual and read it—the first publishable version (somewhat incomplete, but workable) shall be out soon. Since  $\text{\AMS-TeX}$  is still in the process of being designed, and is meant to make life easier for you (the mathematician and/or technical typist), suggestions for improvements will be welcome.

## MINUTES OF THE FIRST TUG MEETING

Robert Morris

Below follows what I heard at the first  $\text{\TeX}$  Users Group meeting in Palo Alto, February 22, 1980. If I've abused the speakers I hope they will write to the newsletter to correct me. Occasionally I have inserted notes cross-referencing other speakers or sources. These are in italics with my initials, ...ram, appended.

A rather broad spectrum of  $\text{\TeX}$  expertise was assumed on the part of the listeners, even within the remarks of a given speaker. I have made no particular attempt to sort this out. Thus  $\text{\TeX}$ pers will be bored by parts of some paragraphs and novices mystified by others. Sorry.

coffee(slc)

People introduced themselves. About 50 were in attendance. Miscellaneous remarks were made, all of which were amplified later except the location of the  $\text{\TeX}$  and METAFONT files for those desiring to get them from SU-AI via the Arpanet. To do this, get the files FILES.INF[TEX,DEK] FILES.INF[MF,DEK] and FILES.INF[FNT,DEK] for  $\text{\TeX}$ , METAFONT, and the SU  $\text{\TeX}$  font files respectively. Each of these tells which files you must further import. In order to avoid pain induced by the SAIL character set, be sure to FTP them in ASCII mode. Failure to do this will cause grief at least about the right brace character, which is rather critical to  $\text{\TeX}$ . Anyone without Arpanet access can get the SAIL sources described above by inquiring of

Dr. Luis Trabb Pardo  
Dept. of Computer Science  
Stanford University  
Stanford CA 94305

Sources obtained in this way will not have the SAIL character set problem because they are generated on TOPS-20 at SCORE. At the moment, only SAIL sources are released. See below about Pascal.

SPEAKER: DONALD KNUTH,  
STANFORD UNIVERSITY.

Don Knuth's opening remarks.

Don opened with some observations about the things he's learned from the  $\text{\TeX}$  project. He did not anticipate the full power of  $\text{\TeX}$  until he had used it extensively himself. For example, until he made macros to pretty-print Pascal programs, he had not thought much about text which was ragged on both left and right. And, early on he thought an interactive  $\text{\TeX}$  would be useful, but finds now that  $\text{\TeX}$  users internalize what  $\text{\TeX}$  will do to such an extent that they usually know what  $\text{\TeX}$  is going to

do about their input and so have no pressing need to see it displayed on a screen immediately after the input is finished. Knuth's conversations with other typographical software users confirm this, e.g., with people setting classified ads in a newspaper.

$\text{\TeX}$  is rather stable now: No reported bugs from October to December, one in January, and one rumored yesterday. (*The SAIL version has been in use at Stanford since August 1978 and at other sites since September 1978. ...ram*)

Radical changes in  $\text{\TeX}$  are to be discouraged, because they might destroy the stability of the system and the interchangeability of  $\text{\TeX}$  input files. An attempt will be made to keep a single common denominator  $\text{\TeX}$  in circulation. Don feels reliability is far more important than allowing everyone to add "missing features" to  $\text{\TeX}$ . When the Pascal version is released there may be some last minute changes of this radical nature, but they should be in all released versions of  $\text{\TeX}$ .

One flaw in this regard was the use of floating point arithmetic in  $\text{\TeX}$ . The first Pascal release will use floating point, as does the SAIL version, with the attendant risk of machine dependencies due to rounding (e.g. in extreme cases, a paragraph might have 11 lines in one implementation and 10 in another if two line breakings had very similar "badness" value). The second Pascal release will use only fixed point arithmetic.

**METAFONT.**

METAFONT has a few dozen users. Don took 3-4 months to create about 80 fonts necessary to set his books. The eminent type designer Hermann Zapf visited, learned METAFONT, taught Don some type design, and helped beautify those fonts. The fonts are available for use with  $\text{\TeX}$ . A Stanford C.S. Report "The Computer Modern Family of Fonts" is forthcoming. (*Don explained in a TUG Steering Committee meeting the next day, that one of the accomplishments of Zapf's visit was to write METAFONT programs incorporating some of Zapf's craft, especially techniques involving changes of pen pressure during letter construction. ...ram*)

Output devices.

The Alphatype CRS is fully functional under  $\text{\TeX}$ . It sets about 3 inches/sec per baseline, requiring about 3-4 minutes per page of moderately complex math. Don rewrote the Alphatype internal software for  $\text{\TeX}$  suitability when using METAFONT fonts.

In general, output on most devices is straightforward in principle. (*See report of David Fuchs' talk below. ...ram*) This includes known phototypesetters and even vector devices, although more software would be involved.

**Audience requests.**

What follows are Don's responses to questions from the audience about specific topics.

**General Organisation of TeX.**

TeX consists of 5 modules:

(1) TEXSYS, about 10% of the code. Contains the storage management and error recovery routines. For complicated TeXing it uses around 30K words (36-bit), but for simple things as little as 8K has been used.

(2) TEXSYN, about 30% of the code. Processes macros, scans input tokens, and otherwise handles TeX syntax.

(3) TEXSEM, about 50% of the code. Responsible for switching modes (math, display, vertical, horizontal), contains the page and paragraph builders, math setting routines, rules. This module has as its main documentation the source code itself. (But see report below on the Pascal version. ...ram)

(4) TEXOUT, about 10% of the code. The output modules, responsible for recursively processing a list of boxes and making a list of output instructions for the device driver.

(5) TEXEXT, presently 0% of the code. The TeX extension module is the place the user defined extensions to TeX are placed. Examples of such will be supplied with the Pascal version, because an indexing and cross-referencing facility has been implemented that way for the time being. Also, researchers at XEROX have implemented color descriptions as an extension. Hooks are placed in TEXSYS to trap the TEXEXT. The feature is powerful and dangerous, providing the user with the ability to clobber any internal TeX data structure. (Readers should not confuse this feature with the macro feature. The TeX extension feature is for the addition of things which TeX can not do, whereas the macro feature parameterises or otherwise makes easy the things that require many or arcane TeX commands to do. Perhaps aside from the indexing feature, most TeX users will never encounter TeX extensions. ...ram)

**Indexing and document management features.**

The initial problem with document management facilities was TeX's inability to make its activities known to anything but its output file. This was the difficulty of interfacing it to CMU Scribe, for example, or other software which needs to know what's going on while TeX is working. Although not all cases are covered, the TEXEXT solution indicated above seems to address many of the resulting problems. The index output thus generated may need some sorting or other trivial post-processing. (See report below of Richard Zippel's MIT document

preparation macro package for non-TEXEXT solution to some of these problems. ...ram)

**The macro facility.**

The macro facility is designed to allow users who know nothing about typography to set papers well. With it, one source can produce many different output forms. This approach is similar to IBM's General Markup Language, and is being pursued by the American Mathematical Society.

**Last minute changes.**

A few are contemplated for inclusion in the Pascal version. The only one posing any incompatibility with existing TeX input files is the improved syntax of font definitions, which presently is similar to that for font changes, being distinguished by context. The new syntax would require existing input files to have their font definition statements changed. The ability to set equation numbers at the left may also be added.

**TeX as a rough draft refinement tool.**

TeX provides the same flexibility and ease of change that any text maintenance use of the computer does.

**TeX as a general typesetting tool.**

TeX was originally designed by Don to set the *Art of Computer Programming*. However, many of his solutions turned out to be appropriate to many other forms of typesetting. The line breaking and paragraph building mechanism is particularly successful, producing text with astonishingly few hyphenations. Widow avoidance is less successful: in mathematics, adding lead to a display is feasible to defeat widows, but in straight text non-uniform interline spacing is frowned upon. Rivers (continuous streams of white space dribbling down a page due to accidents of interword spacing) is not specifically addressed in TeX but seems to be a rare occurrence. Most of Don's effort in design was focused on mathematics spacing.

**Chemical formulas and other scientific typesetting.**

In principle, TeX should be good for most of this work. Don suggests that duplication of effort be avoided by discouraging exploration of non-math typesetting until the American Mathematical Society experience is greater. Some of their solutions should apply immediately to physics and chemistry publishing.

**SPEAKER: LUIS TRABB PARDO,  
STANFORD UNIVERSITY.**

**The Pascal Version.**

Luis sketched the problems associated with the Pascal project and what the final product will

look like. The big question: when will it be released. The answer, not before it has been run successfully not only on the DECSYSTEM 20, but also been transported to an altogether different architecture, the IBM 370. Both tasks are moving along and within sight of completion, but Luis is (*understandably!* ...ram) reluctant to name a date out of concern for being unable to meet any particular deadline. (*If you press the Stanford team privately they will admit that they hope release is months away.* ...ram) Both Knuth and Luis are greatly concerned that a high quality product as bug free as possible be produced. The requirement that the parallel 370 development be complete will help insure that no machine dependent traps will be sprung on those who attempt to move T<sub>E</sub>X. The installation of the 370 version will, moreover, be done by people who are not T<sub>E</sub>X wizards, thereby further simulating the environment released versions will run in. Installation on CDC, VAX, and Univac machines is also in development. Finding suitable Pascal compilers is a big stumbling block. Pascal does not seem to be amenable to large portable software projects because of many unspecified things even in "standard" Pascal.

#### The structure of the Pascal Version.

Pascal T<sub>E</sub>X as released by Stanford will consist of two pieces: the T<sub>E</sub>X module and the system dependent module. The former, comprising 90% of the code, should need almost no local tuning save for specifying a few parameters, such as word size. Further, this should be valid for an entire family, and not site specific. The system dependent module contains the code which allows T<sub>E</sub>X to communicate with the host operating system. It deals, for example, with the file system and the input character set. This is where most of the implementor's work will lie.

The Stanford team has been striving to make the Pascal T<sub>E</sub>X a model of transportable software. To this end, it is to be released in the form of a "documented description" and a support package. The documented description is a topdown description of how T<sub>E</sub>X works. The support package contains two programs for dealing with it. TEXDOC is a program which produces a typeset description of the fully commented T<sub>E</sub>X program. A paper copy of that will also be supplied. UNDOC is a program which converts the documented description into a Pascal program. Usually humans will never read this, studying instead the TEXDOCed version. The support package also includes some sample device driver software, a set of 200 dot/inch fonts and a sample TEXEXT extension.

#### What the implementor must do.

To implement T<sub>E</sub>X do this:

(1) First find a suitable compiler. (*Recipe for rabbit stew: first catch a rabbit!* ...ram) This is not entirely trivial because of the lack of Pascal standards. In fact it was the major delaying factor so far in the project. (*I assume a future issue of the newsletter will have an article detailing just what makes a compiler suitable.* ...ram)

(2) Redefine some of the data allocation policies and a few of the low level macros.

These tasks are constant for an entire architecture and need not be repeated by each site.

(3) Modify certain system dependent calls, typically the names of T<sub>E</sub>X calls to the host operating system. This may be quite site dependent. Luis estimated it might take one month of a system programmer's time for each operating system, but no special knowledge of T<sub>E</sub>X would be needed.

#### Resources.

How greedy is T<sub>E</sub>X? It occupies 75K 36-bit words on a DEC 20 and requires 2-3 CPU seconds per page to set mathematics. Something similar ought to hold on an IBM 370/138. Some investigations into making it work in smaller environments are taking place. Luis believes that such efforts should evolve in the following directions (*see the talk of Bob McClure below* ...ram):

(1) Overlay parts of the program and reduce the size of some of the data structures, e.g., reduce the number of fonts which can be handled at once (currently 64) and the complexity of math which can be set. Such efforts could result in 50% memory reduction.

(2) Make a multipass T<sub>E</sub>X. Natural separation: (a) macro expansion; (b) line breaking and box making; (c) page breaking and output. This might force abandoning some of the interaction between these tasks, e.g., macro definitions in output routines, but it would probably leave a highly useful T<sub>E</sub>X anyway.

(3) Memory management. For portability, T<sub>E</sub>X makes no assumptions about the ability of the host to manage memory, rather simply asking to be given a big block then left alone. An implementation which used the host's memory management might not need such a big chunk.

**SPEAKER: BOB McCLURE, PRIVATE CONSULTANT.**

#### T<sub>E</sub>X in C.

Bob is working on a C version, using many of the smallifying techniques suggested above by Luis. The first target is a Z8000 system running version 7 UNIX, but the programming is not making



use of UNIX features tempting as some of them might be. The aim is to have a TeX which will run in 48Kbytes of memory, thereby fitting on 16-bit machines. Heavy virtualizing of some of the tasks is envisioned. Roughly, the vast resources get stretched out in time and in space (on a disk). Bob isn't prepared to guess about speed of the resulting product which will be proprietary, but for sale at "a reasonable price". (*I hope a rumor mongering section of the newsletter will be devoted to others' efforts at making small TeX, or better still that those doing so will write brief descriptions for us. ...ram*)

**SPEAKER: DAVID FUCHS, STANFORD UNIVERSITY.**

#### Output devices.

David spoke on the structure of TeX output and the technical requirements of device drivers. Since he is devoting an article in this issue to it, I omit details. Roughly, implementors need to write a spooler and a program to convert the TeX output to a form suitable for the output device. This involves decisions about how much of the task to leave to the host, which in turn depends on the communications bandwidth between host and output device. The details thus depend on the host-device combination and may be somewhat site dependent.

**SPEAKER: RICHARD ZIFFEL, MIT.**

Richard spoke about a loosely organized document macro package available at MIT. It has many of the features one wants in document preparation and has some overlap with AMS-TeX. It is reported separately below.

**SPEAKER: MIKE SPTVAK, PUBLISHER OF PERISH PRESS.**

#### AMS-TeX.

Mike is a developer of the AMS version of TeX and the author of "The Joy of TeX". The purpose of these two ventures is outlined by Mike elsewhere in this issue.

**SPEAKER: DICK PALAIS, BRANDEIS UNIVERSITY.**

The final session of the afternoon was chaired by Dick. It consisted of a discussion of existing, contemplated, and desired output devices and a discussion of the TeX Users Group.

#### Output Devices.

Presently running with TeX: Versatec 3211; Varian 9000; Alphatype CRS.

Under development or consideration: Canon laser printer.

Desired: Mergenthaler Linotron 202, 2000 and VIP; GSI C/A/T; APS-5; Compugraphic 8600 and UNISSETTER; AMI; Sanders infinite matrix printer; vector devices in general.

Dick Friday (Digital Equipment) said he had experimented with driving a Diablo and found it not very difficult.

Rumors were mentioned about dramatically falling prices to be announced this June at the National Computer Conference. Others were mentioned that the internal coding of Mergenthaler devices were now sufficiently understood that they might be useful. (*The general problem with phototypesetters is finding out exactly what information they need so that TeX can provide it. ...ram*)

#### TUG organization.

A Steering Committee was elected by acclamation. Its first meeting is reported below. General discussion to guide it centered on the function of a users group. Among the roles proposed were the following (*When it met the next day, the Steering Committee took a much narrower view than the spectrum represented here. ...ram*): organize TeX research; organize "Birds of a Feather" sessions; organize a newsletter; supervise or perform TeX software support; distribute TeX; formulate a test suite to validate programs claiming to implement TeX.

Several people suggested seeking low cost commercial support along the lines of IMSL. Most agreed that an institutional membership fee on the order of \$100 would be acceptable. (*The Steering Committee subsequently decided that such a fee is premature. ...ram*) It was generally agreed that the Users Group would initially be concerned with short term problems so no attempt need be made to find solutions which would be cast in stone. Dick Palais told us that the AMS is applying for trademark protection of the TeX logo and would thus keep administrative control over what could be called TeX.

\* \* \* \* \*

### REPORT OF THE STEERING COMMITTEE MEETING OF FEBRUARY 23, 1980

The Committee met jointly with the AMS Font Subcommittee of the Committee on Composition Technology.

Richard Palais and Robert Morris agreed to be chair and secretary pro tem.

Robert Welland agreed to edit the newsletter. The first newsletter will have a report of the meeting and will be distributed free by the AMS upon inquiry about TeX. Subsequent newsletters will be by subscription only.

There will not be institutional membership until TUG has to play a larger role than circulation of information. Individual membership will be \$10 annually and will cover newsletter expenses. All memberships and fees will be reconsidered at the next Steering Committee meeting.

Richard Friday will work up a proposal for a validation suite for programs whose authors desire to call them  $\TeX$ . Presumably, the Users Group would pass recommendations to the AMS Board of Trustees about a given request to use the  $\TeX$  logo.

The role of the Users Group in distribution will be re-examined after the Pascal release is made. In about six months the Steering Committee will decide when to meet again. At this time it is hoped that Pascal distribution will be under way from Stanford and alternate development sites.

Respectfully submitted,

Robert Morris,  
UMASS/Boston

\* \* \* \* \*

### Interface Software

\* \* \* \* \*

## THE STATUS OF THE PASCAL IMPLEMENTATION OF $\TeX$

September 9, 1980

Ignacio Zabala  
Luis Trabb-Pardo

This document (PTEX.TXT[TEX,IAZ]%SAIL) is intended as a public, up-to-date report on the status of the PASCAL implementation of  $\TeX$ . A file PTEX.BBD[TEX,IAZ]%SAIL is maintained that contains all the mail and new events related to PTEX.

#### SYSTEM ORGANIZATION:

The  $\TeX$ -PASCAL system consists mainly of three modules:

- the TEXPRE module implements the pre-processor that generates the data structures employed by TEX.
- the SYSDEP module contains routines that are very much dependent on the particular host system. It is used both by TEXPRE and TEX.
- the main TEX module.

#### COMPATIBILITIES AND INCOMPATIBILITIES:

Neither TEX nor TEXPRE should need modification at any installation, but, surely, SYSDEP must be adjusted for each host site. The three modules are programmed in PASCAL though some

installations may find it convenient to reprogram SYSDEP in assembler language for the sake of execution speed.

As the default case in the CASE statement is a non-standard feature of PASCAL, it has been given different names by different compilers. Compile time initialization is not standard either, but available under different names in most compilers. These are the only reasons why TEX and TEXPRE may have to be modified, and the modifications are straightforward.

#### PTEX INSTALLATIONS:

$\TeX$ -PASCAL has been running in the PDP-10 (SAIL) machine in our CS Dept., here at Stanford, since April 1980. Since then, all changes made by Don Knuth in the SAIL program have been incorporated immediately into the PASCAL program, which has also undergone modifications as more information was obtained on the characteristics of widely used PASCAL compilers. The program has been in-house tested. It has already processed the whole  $\TeX$  manual and several chapters of Knuth's *Art of Computer Programming*.

Stanford's CIT has an IBM machine of the 370 family. Eagle Berns is in charge of the installation of  $\TeX$  there. He has obtained a copy of the new IBM PASCAL VS compiler, and has tried to compile PTEX with it. As of September 9, both TEXPRE and SYSDEP (in PASCAL, only slightly modified) had compiled and run successfully to generate the table of data structures employed by the main TEX module. For these trials, Berns used 36 bpw font information files from SAIL. There was a problem with two procedures in the main TEX module which were still too large for the VS compiler. This module has been modified accordingly and we are waiting for feedback from Berns.

Charles Lawson is installing PTEX on the UNIVAC 1100 of the Jet Propulsion Lab at Caltech using the University of Wisconsin PASCAL compiler. He has made a good programming effort to get PTEX up. (For instance, he has coded the "environment" modules required by his compiler.) This installation seems to be already past the compilation phase.

At the University of Minnesota, Mike Frisch is installing PTEX on a CDC-Cyber.

George Otto is in charge of the installation at Wharton. The Moore school has a UNIVAC 90-VS/9 where they use the PASCAL-8000 compiler.

David Kashtan has compiled everything successfully on the VAX (VMS) at SRI.

Richard Friday has also compiled everything on a VAX (UNIX) at DEC.

**TEX-PASCAL** has been distributed to more centers: Stanford Linear Accelerator Center, University of Aarhus in Denmark, Universities of Milan and Pisa in Italy, University of Valencia in Spain, etc. The ones mentioned above have given the most feedback.

At this time, it seems that most pioneer installations are free of compilation problems and are now trying to obtain adequately interfaced output devices, together with fonts and font information files suitable for them.

The only fully operational PTEX system is still this at Stanford CSD, but we expect to be printing DVI files produced by the CIT installation very soon.

#### COMPILER ISSUES:

**TEX-PASCAL** was developed using the Hamburg PASCAL compiler for the PDP-10 by Kisicki and Nagel. Some compiler maintenance was needed during the debugging of PTEX. We have found this to be a rather powerful and permissive compiler.

There have only been three system requirements on PTEX hosts and these were explicit since the beginning of the project:

- The system must have enough addressable memory to store the large arrays employed by PTEX (about 128K words of 32 bits).
- The compiler should be able to really pack fields of a PACKED RECORD and overlap multiple variants of packed records. If this requisite is not satisfied, PTEX will require at least four times as much memory.
- The compiler should be able to handle large case statements (say over 64 actual cases in a range [-500..500]) and have a default case (this is non-standard in PASCAL but available in most compilers).

Additionally, PTEX requires an EXTERNAL (or separate) compilation facility. If no such thing is available, the SYSDEP module has to be inserted both in TEX and in TEXPRE by hand. Also, if there is no compile time variable initialization, the INITPROCEDURE appearing in the program has to be changed into an ordinary procedure.

We have fought not to add more requirements and have changed the program to facilitate the installation with simpler or more restrictive compilers. Encountered problems have been common to most pioneer installations:

- lines of code were too long
- octal constants were not accepted
- identifiers containing the underscore character were not accepted
- some identifiers were too long

- sometimes two different identifiers were equal in the first eight characters
- fields of packed records could not be passed as procedure arguments
- loop counters had to be local variables
- all declared labels had to be used
- use of GOTOs was restricted: not even allowed from the body of a procedure out to the block in which the procedure was declared
- there were discrepancies in the treatment of nested WITH statements
- the compiler lacked the standard MAX and MIN functions
- procedures had to be kept small (less than 400 statements)

The program has been modified to avoid them. Currently, the code is all uppercase in lines that are never longer than 72 characters. All identifiers are shorter than 16 characters and differ in the first 8 characters. Octal variables appear only in SYSDEP.

#### DISTRIBUTION:

Currently, **TEX-PASCAL** can be obtained from the **TEX** group at the CS Dept. at Stanford. Anyone asking for the system will get a tape containing the files **TEX.PAS**, **TEXPRE.PAS**, **SYSDEP.PAS**, **TEX.STR**, **TEXPRE.STR**, and **SYSDEP.STR**, which is about everything that is needed to have PTEX running. The distribution package also contains a short installation guide, a description of the DVI format of the output file of TEX, and extensively documented listings of TEX, TEXPRE and SYSDEP. (Of course, the ultimate documentation on TEX is the **TEX** manual.) All these files (not the listings) are available on-line in the directory (TEX.PASCAL)%SCORE, accessible via the ARPANET.

Fonts and font information files may also be provided on request (in the format employed here at SAIL). These files are very system-and-output-device-dependent and of restricted general value for that reason.

\* \* \* \* \*

## THE FORMAT OF **TEX**'S DVI FILES

David Fuchs

DVI files contain information about where characters go on pages. The format is such that there are those who claim that almost any reasonable device can be driven by a program that takes DVI files as input. In particular, DVI files can be sent to the Xerox Graphics Printer (XGP), Versatec, Canon or

Alphatype at the Stanford CS Dept., depending on what spooler it is passed to. The format follows.

The basic unit of information in a DVI file comes in an 8-bit chunk. Here at Stanford, they are packed four per word, in the lower-order 32 bits of each word, and the highest-order chunk is considered to be before the others, etc.

The DVI file contains a number of Pages followed by a Postamble. Each Page starts with a BOP command, has lots of other commands, and ends with an EOP command. Each EOP command is immediately followed by another BOP command, or the PST command, which means that there are no more Pages in the file, and the Postamble follows. See below for details on all the commands that occur in Pages, and what goes in the Postamble.

Each Page consists of a number of Commands that specify what characters should be typeset where. Who- or what-ever reads these Pages should have a Stack that can hold, say, 200 coordinates (i.e. integers) to be on the (very) safe side.

There is a notion of the "current position on the page", which is specified by its horizontal and vertical coordinates. Moving rightwards on a page is represented by an increase in the H-coordinate, while moving down is an increase in V, and the upper-left-hand corner of the page is 0,0 (i.e. it's slightly non-cartesian). Coordinates are given in *rsu's* (ridiculously small units), where  $1rsu = 1/2^{16}$  points. This is so that accumulated errors will be undetectable even in the worst imaginable case (a "box" many feet long). Whenever a character or rule is set, it gets put at the current position on the page. The current position on the page is changed by explicit move commands (their names begin with W, X, Y, and Z). It can also change as a side effect of setting a character or rule (the 0-127 and VERTRULE commands). The w-, x-, y-, and z-amounts are not locations, but distances (in *rsu's*). Some commands change their values, and some cause the current H- or V-coordinate to be incremented by one of their current values.

A lower-case character with a bracketed number following a command means that the command has a parameter that is that many bytes long. Thus, the BOP command, for instance, is 9 bytes long, the first byte of which has the decimal value 129, the second through fifth of which give the page number (high order byte first), and the sixth through ninth being another number which is explained below. These numbers are in two's complement, so they should be sign-extended on the left when they are read.

The commands are:

Command	Description
0 to 127	Set the appropriate character from the current font such that its reference point is at the current H,V location, and then increment the current H-coordinate by the character's width.
128 NOP	No-op, do nothing, ignore.
129 BOP n<4> p<4>	Beginning of page n, with pointer p to the BOP command of the <i>previous</i> page. By "pointer" is meant the relative byte number within the DVI file, where the first byte (the BOP of the first page) is byte number zero. (ex.: If the first page had only a BOP and EOP, the <i>third</i> page's pointer would be 9, because the BOP command takes bytes 0 to 7, the EOP is 8, so the <i>second</i> page's BOP is in byte 9. Get it?). The <i>first</i> page has a -1 for a pointer; the second, a zero. Start the H- and V-coordinates out at 0, as well as the w-, x-, y-, and z-amounts. The stack should be empty, and no characters will be set before a FONT(NUM) command occurs. Remember that n can be < 0, if the page was Roman Numbered. Also the pages need not come in the proper order in the file, depending on who's doing the <i>TeXing</i> .
130 EOP	The end of all commands for the page has been reached. The next page, or the postamble, starts in the next byte.
131 PST	The postamble starts here. See below for the full explanation of what goes in the postamble.
132 PUSH	Push the current values of the H- and V-coordinates, and the current w-, x-, y- and z-amounts onto the stack, but don't alter them (so an X0 after a PUSH will get to the same spot that it would have had, had it been given just before the PUSH).
133 POP	Pop the z-, y-, x-, and w-amounts, and the V- and H-coordinates off the stack.
134 VERTRULE h<4> w<4>	Same as HORZRULE, but also increment the current H-coordinate by w when done (even if $h \leq 0$ or $w \leq 0$ ).
135 HORZRULE h<4> w<4>	Typeset a rule of height h and width w, with its bottom left corner at the current H,V position. If $h \leq 0$ or $w \leq 0$ , no rule should be set.

Command	Description
136 HORZCHAR <i>c</i> <1>	Set character <i>c</i> just as above, but don't change the current value of the H-coordinate (or V-coordinate, either).
137 FONT <i>f</i> <4>	From now on, set characters from font number <i>f</i> . Note that this command is not currently used by TEX—it is only needed if <i>f</i> is greater than 63. See FONTNUM commands below.
144 X2 <i>m</i> <2>	Move right <i>m</i> <i>rsu</i> 's by adding <i>m</i> to the H-coordinate, and put <i>m</i> into the current x-amount. Note that <i>m</i> is in 2s complement, so this could actually be a move to the left.
143 X3 <i>m</i> <3>	As above.
142 X4 <i>m</i> <4>	As above.
145 X0	Move right the current x-amount (which can be negative, etc).
140 W2 <i>m</i> <2>	The same as the X commands (i.e. alters H-coordinate), but alter w-amount rather than x-amount, so that doing a W0 command can have different results than doing an X0 command.
139 W3 <i>m</i> <3>	As above.
138 W4 <i>m</i> <4>	As above.
141 W0	Move right the current w-amount.
148 Y2 <i>n</i> <2>	Same idea, but now it's "down" rather than "right", so the V-coordinate changes, as does the y-amount.
147 Y3 <i>n</i> <3>	As above.
146 Y4 <i>n</i> <4>	As above.
149 Y0	Guess.
152 Z2 <i>m</i> <2>	Another downer. Affects the V-coordinate and z-amount.
151 Z3 <i>m</i> <3>	
150 Z4 <i>m</i> <4>	
153 Z0	Guess again.
154 to 217 FONTNUM's	Make 0, 1, ..., 63 the current font.
218 to 255	are currently undefined and will not be output by T <sub>E</sub> X.

Pages need not be sequential by number, but any blank or non-existent page might not be represented, so page -5's pointer to the "previous page" might point to page 34, for instance (remember that TEX uses negative numbers for roman-numbered pages). The first page in the file has a "previous page" pointer of -1.

The postamble begins with a PST command, followed by four bytes of previous-page pointer to the last real page, followed by four bytes of the height of the tallest page (in *rsu*'s), followed by four bytes of the width of the widest. Next come some Font Definitions (maybe none, if you're an authoritarian), each of which has a Font ID in the first 4 bytes, followed by 4 bytes of Font Number, followed by any character not in the font name, followed by the Font Name, one character per byte for as many bytes as necessary, followed by that same character that was not in the Font Name (a quote is probably a good choice for such a character). The end of the font definitions is marked by an ID of -1 (which will not be followed by font number, etc). The four bytes following this phony ID are a pointer to the PST command (i.e. the beginning of the postamble), which is followed by a zero byte, which is followed by at least 4 bytes containing the number 223<sub>10</sub> (which is '337 octal). The reason for some of the above weirdness is twofold: We are producing DVI files with a Pascal program, and to avoid doing any non-serial I/O, the postamble pointer has to go at the end of the file. Of course, most programs that read these files need not be generally transportable, and can do a random seek to the end of the file, and then another to get right to the postamble. The fact that page-pointers point backwards is in the same spirit, but this also allows the file to be read in backwards-page-order efficiently. This, in turn, will allow for further efficiencies in communicating with your device, depending on how clever it (and you) is (are).

Stanford University

July 10, 1980.

\* \* \* \* \*

UNIVERSITY OF MINNESOTA  
CDC SITE REPORT  
Thea Hodge

We have succeeded in compiling T<sub>E</sub>X-in-PASCAL on our Cyber 172 but cannot yet run it. TEXP<sub>RE</sub>, which should generate the required table file, has some problem relative to our system. Michael Frisch, our manager of user libraries and graphics software, is working on that. We are awaiting

the font tables promised to us by I. J. Zabala of Stanford.

Luis and his staff have broken TeX into parts so that it is easier to compile, although the binary will still be very large, I believe. The parts are:

1. the system-dependent user library,
2. the rest of the TeX program,
3. tables and font files.

We are working on the input and output from several different points.

A. We have upgraded a Decwriter II (LA36) with a SELANAR Graphics II circuit board which allows us to have 4 character sets online at a time—standard, APL, Math-Greek as defined by SELANAR Corp., and any set we wish to down-load from a Cyber or from a Terak. We have encountered a few difficulties in the Terak-Decwriter interaction and will continue to work on these.

B. Presently, we can produce text with special characters as follows: enter and edit the text on a Terak (176 characters online and displayable on the screen), send the file asynchronously to a Cyber for formatting with Purdue's TXTFORM, send back the file for re-editing on the Terak, return final version to the Cyber where TXTPLOT generates the codes to produce medium-quality hard copy on the Varian printer-plotter (200 dots/inch). We are also working on a program in CDC Fortran 77 to translate a TROFF or TeX output file for the Varian.

C. We are working in the University's printing department, driving the Mergenthaler Linotron 202 with a Terak microcomputer. We have with partial success typeset a page of a textbook. We write a TROFF file to an 8-inch floppy on a PDP-11/40 and use this file on a Terak to drive the Linotron.

Throughout this memo I have used "we" as a collective pronoun for all of us involved in this project. In fact, this project is made up of many small projects. The important names are Michael Frisch, Prof. Steven Bruell, Peter Zechmeister, Mark Everett, and Jeffrey Woolsey. I will be happy to forward any questions you may have to the appropriate person.

I would like to hear from all CDC users about what you are doing in text processing. In particular, I want to hear about and would be glad to distribute information about your efforts and achievements in the area of I/O. If you know of other interested people at CDC sites, I would be glad to add their names to my TeX-related mailing list.

\* \* \* \* \*

## Warnings & Limitations

\* \* \* \* \*

### Troubles with Traces, and Other Oddities

One of the options of `\trace` (manual page 147),  $z = 2$ , is not available under TOPS-20 (DECSystem 20); if that option is used, input lines are displayed on your terminal, but a carriage return flushes them into oblivion instead of delivering them to TeX for processing.

Another thing to be careful of is that such commands as `\baselineskip`, if invoked within an `\hbox` par, have no effect whatever on the boxed text; they must be given prior to the box command to be effective. This is actually documented (manual page 128, `(glueparam)(glue)`, and perhaps on other pages), but it is easy to overlook.

Ligatures "ff", "ffi", "ffl", etc. interact with hyphenation in the following manner. Hyphenation never occurs between characters of a ligature in a word, even though such a word ("differential") might legitimately be hyphenated, and in fact might require hyphenation to avoid an overlong line. The solution for an overlong line is to use a discretionary hyphen, `\-`, but there's a catch: if other changes in the paragraph cause the forcibly hyphenated word to move from the end of a line, neither a hyphen nor a ligature will result (compare: "differential", "differential"), and the discretionary hyphen should be removed for most elegant results.

Barbara Beeton

\* \* \* \* \*

## Macros

\* \* \* \* \*

There are two reports on macro packages in this issue. One appears above: see Michael Spivak's "AMS-TeX—A Very Friendly Product" under General Delivery. (An order form for the "pre-preliminary" edition of the AMS-TeX manual is included in the package with this issue.) The second report, "An Indexing Facility for TeX", was submitted by Terry Winograd and Bill Paxton, and is attached as Appendix A.

\* \* \* \* \*

**Questions & Answers**

\* \* \* \* \*

Late Saturday afternoon at the TUG meeting at Stanford, someone had the excellent idea that we should form subcommittees based on the various types of mainframes. The meeting broke up into small collections of people, with the members of each group being people who work on common computers. Each of these groups chose a chairperson, but unfortunately no one made a list of these groups or of the chairpersons. Would these chairpersons please send their names to the TUGboat editor. Also please give us a TeX status report from your site.

\* \* \* \* \*

**Letters**

\* \* \* \* \*

A letter to Richard Palais from two satisfied users at the University of Rochester, Gérard Emch and Arnold Pizer, is reproduced on the next page.

\* \* \* \* \*

\* \* \* \* \*

**Miscellaneous**

\* \* \* \* \*

Two large items are enclosed as separate documents:

1. TeX Errata. This is the up-to-date list of corrections and changes to all versions of the TeX manual and to TeX itself, as compiled by the TeX group at Stanford.

2. TUG Mailing List. This list includes the names of all persons who had requested information on TeX and TUG as of October 24, 1980. Since membership dues are only being solicited with this issue of TUGboat, this list cannot be considered the official membership list. It does, however, show the types of equipment and potential applications of TeX in which most persons expressed interest, and can thus be used to identify other sites whose interests are the same as yours.

\* \* \* \* \*

## THE UNIVERSITY OF ROCHESTER

DEPARTMENT OF MATHEMATICS  
MATHEMATICAL SCIENCES BUILDING  
ROCHESTER, NEW YORK 14627

Phone 716-275-4411

September 25, 1980

Professor Richard Palais  
Department of Mathematics  
Brandeis University  
Waltham, MA 02154

Dear Dick,

This letter is to convey to you in concise form our impressions on the use of TEX and AMSTEX based on our implementation of two very different projects.

The first project was the setting of "Hecke Operators for  $\Gamma_0(N)$ ," a 26 page paper by Pizer, which involved fairly complicated formulae and constructions. Pizer spent three or four weeks on this, working mostly in the evenings. This included all the learning and experimentation to be expected when a new system is used for the first time on a self-taught basis.

The second project was much more modest in scope: a three-page announcement presented at the AMS Summer Research Institute, held from July 14 to August 2, 1980 at Kingston, Ontario. Working under rather intense time pressure, Emch, starting from scratch, managed to produce this paper in two days including one night spent on a first reading of Knuth's manual. It should be said that he benefitted from Pizer's guidance, on the spot availability, and experience with the system.

It is our conviction that the whole system, together with Knuth's manual, is an eminently usable tool in the hands of any mathematician, even one who would sit at a computer terminal for the first time. The task will be made even easier with the complete AMSTEX package.

It should be recognized that setting complicated alignments and displays (such as commutative diagrams for example) can be quite difficult, at least for novice users such as ourselves. However, as the goal of AMSTEX is to create "MACROS" that will allow one to easily set almost any construction that might appear in a mathematics paper, most of these difficulties should disappear when AMSTEX becomes fully operational.

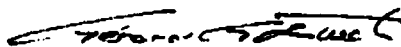
In closing we should say that both of us are very pleased with the results of our first experiences with TEX and AMSTEX



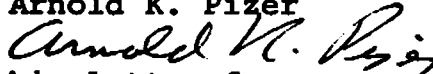
and are quite proud of the typographical appearance of our respective papers. We now hope that our secretaries too will be seduced by the "Joy of TEX".

Sincerely,

Gérard G. Emch



Arnold K. Pizer



P.S. Please feel free to use this letter for any propaganda purpose you might see fit.



## Appendix A

### An Indexing Facility for TEX Terry Winograd and Bill Paxton July 17, 1980

We have created a set of TEX macros and INTERLISP programs that generate an alphabetical index in various standard book formats. The index terms are sprinkled into the text, using a macro. As a side effect of compiling the file to produce pages, TEX creates an index file designed to be read by an INTERLISP program. This program can merge any number of index files and produce an alphabetized formatted index that can then be compiled by TEX to produce the final index pages.

It is easiest to understand this by looking at examples and seeing how they come out. First we will go through an extended example, then describe the features, then finally give the code. The sample index produced is:

Artificial intelligence, 70, 76, 82, 91, 526,  
529

Constantinople, 12ff. *See also* Istanbul.  
alien rule of, 20ff.  
Arab invasion of, 19, 31n.  
Crusades and, 22ff., 57  
founding of, 14, 16ff.  
Golden Age of, 17ff.  
Greek takeover of, 28-29  
Ottoman conquest of, 29-31  
Persian attack on, 18-19  
Venetian sack of, 31

Darwin, Charles, 82  
Darwin, Max, 82  
Death. *See* Eggs, Life.

Eggs  
fried, 530. *See also* Death.  
scrambled (yuk), 526  
Everything else, 70. *See also* Life.

Indexing, 70, 527-30  
cross, 76, 529  
strategies, 526  
typography, 527ff.  
strategies, 529  
Istanbul, 63. *See also* Constantinople.  
Life, 82-91

The output of the INTERLISP program is designed to provide a great deal of flexibility in formatting the final index. We have developed three different styles; if none of them suit you, there is a good chance you can produce something that will. In addition to the "entry-per-line" style shown above, we offer run-in (paragraph) style and a combined style in which subentries all begin a new line, preceded by an em dash, but sub-subentries are run-in. In general, the entry-per-line style is best if for a complex index, paragraph style is more economical in terms of space, and the combined style is a compromise between the other two. Here is the previous example using these alternative styles.

**Paragraph style**

Artificial intelligence, 70, 76, 82, 91, 526, 529

Constantinople, 12ff.: alien rule of, 20ff.; Arab invasion of, 19, 31n.; Crusades and, 22ff., 57; founding of, 14, 16ff.; Golden Age of, 17ff.; Greek takeover of, 28-29; Ottoman conquest of, 29-31; Persian attack on, 18-19; Venetian sack of, 31 *See also* Istanbul.

Darwin, Charles, 82

Darwin, Max, 82

Death. *See* Eggs, Life.

Eggs: fried, 530. (*See also* Death): scrambled (yuk), 526

Everything else, 70. *See also* Life.

Indexing, 70., 527-30: cross, 76, 529 (strategies, 526; typography, 527ff.); strategies, 529

Istanbul, 63. *See also* Constantinople.

Life, 82-91

**Combined style**

Artificial intelligence, 70, 76, 82, 91, 526, 529

Constantinople, 12ff. *See also* Istanbul.

—alien rule of, 20ff.

—Arab invasion of, 19, 31n.

—Crusades and, 22ff., 57

—founding of, 14, 16ff.

—Golden Age of, 17ff.

—Greek takeover of, 28-29

—Ottoman conquest of, 29-31

—Persian attack on, 18-19

—Venetian sack of, 31

Darwin, Charles, 82

Darwin, Max, 82

Death. *See* Eggs, Life.

Eggs

—fried, 530. *See also* Death.

—scrambled (yuk), 526

Everything else, 70. *See also* Life.

Indexing, 70., 527-30

—cross, 76, 529: strategies, 526; typography, 527ff.

—strategies, 529

Istanbul, 63. *See also* Constantinople.

Life, 82-91

In addition to these major style variations, you can modify the TEX macros to control smaller details as well. For example, you can easily change the manner in which primary references are indicated from boldface to something else, or you can change the formatting for cross references to put them all in parentheses. Comments in the TEX macros should help you to make the necessary changes to get the style you want.

These examples are the result of merging two index files, one of which came from the source file on the following page, which was called TESTINDEX.TEX.

The macro '\<' does indexing. It takes two arguments, the first of which is a single character and the second of which is the entry. The character means:

```
.      ordinary reference to the page on which it appears
:      boldface form of . (for use in giving primary reference)
-      begin span reference (e.g. '23-47') on this page
=      boldface form of -
+      end span reference on this page
|      'ff' reference to this page (e.g., '56ff.')
```

use this to indicate the page on which a long discussion begins

```
!      boldface form of |
,      'n' reference to this page (e.g., '78n.')
```

use this with a page reference to a footnote

```
;      boldface form of ;
*      author reference (see details below) to this page
+      cross reference (see details below)
```

Each call to '\<' contains a sequence of terms, separated by semicolons and is terminated with a '>'. The exceptions are the author reference (which calls for precisely two terms--last and first names) and the cross reference (which has the sequence of terms followed by an '=' followed by the phrase to be used in the '(see ...)'). Calls to the index macro do not affect the regular TEX output file at all. This was done because the exact wording of the index term is often not identical to a sequence of characters appearing in the text. It was more uniform to treat the index and text as always distinct. Also, the system carries through whatever capitalization you use in the index terms. The example here uses what is more or less standard in publishing. In this example we have used the character macro feature (\chcode = 13) to allow the character '<' to stand for the sequence '\<'. If you want to use '<' normally, you can skip this and simply use '\<' for index items.

#### TEX SOURCE FILE "TESTINDEX.TEX"

```
\input <tex>basic
\input index
\def\setpage #1 {\par\vfill\ject\setcount0 #1\setcount7 #1}
\openIndex testindex

\setpage 12
Our saga begins in the ancient city of Constantinople. <|Constantinople>X + Constantinople = Istanbul>
\setpage 14
It was founded long ago. <.Constantinople;founding of>
\setpage 16
There are many things to say about its founding. <|Constantinople;founding of>
\setpage 17
The Golden Age of the city lasted for several hundred years. <|Constantinople;Golden Age of>
\setpage 18
The beginning of the end for the city was the fierce Persian attack. <-Constantinople;Persian attack on>
\setpage 19
After the Persian attack, there was an Arab invasion. <+Constantinople;Persian attack on>X.Constantinople;Arab invasion of>
```

```

\setpage 20
This marked the beginning of a period of alien rule. <|Constantinople;alien rule of>
\setpage 22
The Crusades had a major impact on Constantinople. <|Constantinople;Crusades and>
\setpage 28
The Greeks took over the city. <-Constantinople;Greek takeover of>
\setpage 29
The Ottoman conquest of the city soon followed. <+Constantinople;Greek takeover of><-
Constantinople;Ottoman conquest of>
\setpage 31
The Venetians sacked the city. <+Constantinople;Ottoman conquest of><.Constantinople;Venetian
sack of> In a footnote, we compare this to the earlier Arab invasion. <.Constantinople;Arab invasion
of>
\setpage 57
Once again the Crusades reached the city. <.Constantinople;Crusades and>
\setpage 63
The city became known as Istanbul. <.Istanbul><+ Istanbul = Constantinople>

\setpage 70
This is material on indexing <.Indexing> and artificial intelligence <.Artificial intelligence> and
everything else, <.Everything else> which is what life is all about <+ Everything else = Life>
\setpage 76
This is the main reference to cross indexing <:Indexing;cross> files. It also mentions AI <.Artificial
intelligence> and another reference to cross indexing <.Indexing;cross> happens to fall on the same page
so should not appear redundantly. If it had happened to fall across the page break I would want both
pages to have references.
\setpage 82
I begin discussing life <-Life> on this page, quoting from Charles Darwin. <*Darwin;Charles> and at
times from his brother Max <*Darwin;Max> who did research in AI. <.Artificial intelligence>
\setpage 91
Here ends our discussion of life <+Life> and AI <.Artificial intelligence> and other matters. It contains a
duplicate crossreference having to do with life and everything else. <+ Everything else = Life>
\setpage 94
It also has described death <+ Death = Life> to some extent.

\setpage 526
We also want to discuss scrambled eggs, <.Eggs;scrambled (yuk)> AI <.Artificial intelligence> and cross
indexing strategies. <.Indexing;cross;strategies>

\setpage 527
Now I begin the main discussion of indexing <= Indexing> with several pages on the typography of
cross indexes. <|Indexing;cross;typography>
\setpage 529
Here I mention cross indexing <.Indexing;cross> again, along with some general strategies
<.Indexing;strategies> useful in doing indices. I also both begin and end a discussion of AI. <-Artificial
intelligence> <+Artificial intelligence> This could happen with a begin and end that weren't separated
very far and might end up on either the same or adjacent pages. When they fall on the same page we
want a simple reference, not a span.
\setpage 530
This is the end of indexing, <+Indexing> which is more complex than fried eggs. <.Eggs;fried> Some
poets of the absurd have argued that death is really just an ultimate form of eggs. <+ Death = Eggs>
<+ Eggs;fried = Death>

\par\vfill\eject\end

% NOTE: the calls to \setpage are not normally used. They are
% included here to create an output that has lots of pages from
% a short test file and to include high page numbers. The system ordinarily
% works with the standard page breaking, indexing things by the page on

```

% which they appear.

When TEX compiles TESTINDEX.TEX, it produces the regular output TESTINDEX.PRESS, plus a file TESTINDEX.INDEX which contains:

<12;N;F;Constantinople>  
 <Istanbul;N;C;Constantinople>  
 <14;N;P;Constantinople;founding of>  
 <16;N;F;Constantinople;founding of>  
 <17;N;F;Constantinople;Golden Age of>  
 <18;N;S;Constantinople;Persian attack on>  
 <19;N;E;Constantinople;Persian attack on>  
 <19;N;P;Constantinople;Arab invasion of>  
 <20;N;F;Constantinople;alien rule of>  
 <22;N;F;Constantinople;Crusades and>  
 <28;N;S;Constantinople;Greek takeover of>  
 <29;N;E;Constantinople;Greek takeover of>  
 <29;N;S;Constantinople;Ottoman conquest of>  
 <31;N;E;Constantinople;Ottoman conquest of>  
 <31;N;P;Constantinople;Venetian sack of>  
 <31;N;N;Constantinople;Arab invasion of>  
 <57;N;P;Constantinople;Crusades and>  
 <63;N;P;Istanbul>  
 <Constantinople;N;C;Istanbul>  
 <70;N;P;Indexing>  
 <70;N;P;Artificial intelligence>  
 <70;N;P;Everything else>  
 <Life;N;C;Everything else>  
 <76;B;P;Indexing;cross>  
 <76;N;P;Artificial intelligence>  
 <76;N;P;Indexing;cross>  
 <82;N;S;Life>  
 <82;N;P;Darwin, Charles>  
 <82;N;P;Darwin, Max>  
 <82;N;P;Artificial intelligence>  
 <91;N;E;Life>  
 <91;N;P;Artificial intelligence>  
 <Life;N;C;Everything else>  
 <Life;N;C;Death>  
 <526;N;P;Eggs;scrambled (yuk)>  
 <526;N;P;Artificial intelligence>  
 <526;N;P;Indexing;cross;strategies>  
 <527;B;S;Indexing>  
 <527;N;F;Indexing;cross;typography>  
 <529;N;P;Indexing;cross>  
 <529;N;P;Indexing;strategies>  
 <529;N;S;Artificial intelligence>  
 <529;N;E;Artificial intelligence>  
 <530;N;E;Indexing>  
 <530;N;P;Eggs;fried>  
 <Eggs;N;C;Death>  
 <Death;N;C;Eggs;fried>

This file is then fed to LISP. The transcript in doing this was:

```

3+LOAD(TEXINDEX.COM]
compiled on 10-JUNE-80 11:57:41
FILE CREATED 10-JUNE-80 11:56:41
  
```

```

TEXINDEXCOMS
<PAXTON>TEXINDEX.COM;20
4←INITIALIZE]
NIL
5←READINDEX(TESTINDEX]
NIL
6←WRITEINDEX(TESTOUT]
NIL

```

The program has three user-accessible functions: INITIALIZE(), READINDEX(NAME) and WRITEINDEX(NAME). INITIALIZE is called once on starting it up, READINDEX can then be called any number of times, reading in (and merging) .INDEX files. Finally WRITEINDEX is called once to write a file with extension .TEX which can then be compiled by TEX to produce the index. The resulting file, TESTOUT.TEX contains:

```

\indexStart
\indexChar{A}
\indexEntry0{Artificial intelligence, 70, 76, 82, 91, 526, 529}{{{}-{}]}
\indexChar{C}
\indexEntry0{Constantinople, \indexFF 12.\pageNumDot}{{\indexAlso{Istanbul}}+{
\indexEntry1{alien rule of, \indexFF 20.\pageNumDot}{{{}-{}]}
\indexEntry1{Arab invasion of, 19, \indexN 31.\pageNumDot}{{{}-{}]}
\indexEntry1{Crusades and, \indexFF 22., 57}{{{}-{}]}
\indexEntry1{founding of, 14, \indexFF 16.\pageNumDot}{{{}-{}]}
\indexEntry1{Golden Age of, \indexFF 17.\pageNumDot}{{{}-{}]}
\indexEntry1{Greek takeover of, \indexSpan 28-29.}{{{}-{}]}
\indexEntry1{Ottoman conquest of, \indexSpan 29-31.}{{{}-{}]}
\indexEntry1{Persian attack on, \indexSpan 18-19.}{{{}-{}]}
\indexEntry1{Venetian sack of, 31}{{{}-{}]}
\indexChar{D}
\indexEntry0{Darwin, Charles, 82}{{{}-{}]}
\indexEntry0{Darwin, Max, 82}{{{}-{}]}
\indexEntry0{Death}{{\indexSee{Eggs, Life}}-{}]}
\indexChar{E}
\indexEntry0{Eggs}{{}+{
\indexEntry1{fried, 530}{{\indexAlso{Death}}-{}]}
\indexEntry1{scrambled (yuk), 526}{{{}-{}]}
\indexEntry0{Everything else, 70}{{\indexAlso{Life}}-{}]}
\indexChar{I}
\indexEntry0{Indexing, 70, \mainEntry{\indexSpan 527-30.}}{{}+{
\indexEntry1{cross, \mainEntry{76}, 529}{{}+{
\indexEntry2{strategies, 526}{{{}-{}]}
\indexEntry2{typography, \indexFF 527.\pageNumDot}{{{}-{}]}
\indexEntry1{strategies, 529}{{{}-{}]}
\indexEntry0{Istanbul, 63}{{\indexAlso{Constantinople}}-{}]}
\indexChar{L}
\indexEntry0{Life, \indexSpan 82-91.}{{{}-{}]}
\indexEnd

```

Finally, this is put back through TEX with the initialization to produce the index.



## COMMENTARY

Some of the non-obvious features are:

Page references appear in numerical order, with a simple reference preceding a span that begins on the same page. You can safely put in multiple references or begin-end pairs without knowing whether they will end up on the same page or adjacent pages. If there are multiple references to the same page, only one will appear. It will be bold if any of them were. If a span begins and ends on the same page a simple page reference appears instead. Unmatched begins and ends will be noticed by the INTERLISP program as it runs. It also notices and prints a message on the console about the presence of two spans starting on the same page.

Inclusive page numbers are elided to produce spans such as 107-9, 119-22, 245-49, 800-802. (The rule is: omit from the second number the digit(s) representing hundreds, except when the first number ends in two zeros, in which case the second number is given in full.) If you want page numbers to be given in full rather than elided, provide a second argument of T to the WRITEINDEX function.

Upper and lower case do not influence the alphabetical ordering of terms.

Cross-references are combined into a single list, in alphabetical order, regardless of where they were in the text. The LISP output indicates if there were any references to pages or any subterms so the TEX macros can produce *see* in some cases and *see also* in others. If the same cross-reference is given more than once in the source file, it appears only once in the output.

Spans (e.g. '3-5') are boxed, so they will not be split across lines.

Nesting of subterms can go any number of levels (assuming the width assigned for the index can stand it, and the TEX argument stack doesn't overflow). The deepest example here is 'Indexing, cross, strategies'.

When an entry is too long for its line, it is continued on the next line, indented two steps in the entry-per-line styles (see 'Artificial intelligence').

Spaces can appear in a term (e.g. 'Cognitive science'), as can parentheses (e.g. 'scrambled (yuk)'). In fact all characters except semicolon, '<' and '>' are treated exactly as TEX would normally treat them (e.g. single carriage returns, tabs and sequences of spaces become spaces, characters with special syntax like '{' and '\' have different effects, etc.). Because the TEX \send command is used, some slightly funny things happen when the text is used. For example, a built in TEX command (like \char) will get carried through the whole process and eventually produce its effect on the final output. A defined macro (including those defined in BASIC.TEX) will get expanded before the entry is sent to the file. If you really need a semicolon, '<' or '>' (or something else TEX won't let you use as a normal character), you have to put a \char in the index term. This may do funny things to the ordering, but then characters like semicolon and '>' are funny to alphabetize anyway.

Before the first entry for a letter, there is a call on the \indexChar macro with the character as argument. In our example, this macro simply puts out some blank space; in a larger index you might want to redefine \indexChar to put out the character too.

## PROGRAMS

The TEX macros for the first pass are:

```
\gdef\lessthan{<}
\gdef\angbr#1{<#1>}
\chcode'74+13 % This allows < to stand for \< and gives us
               % an alternative way to put a < into an output (including a send)
```

`\def<#1>{}%` Ignores index terms when index is not being generated.

```
\gdef\openIndex#1 {\open1=#1.INDEX
\gdef<##1##2>{\if .##1{\doIndex{N;P}{##2}}\else
{\if :##1{\doIndex{B;P}{##2}}\else
{\if -##1{\doIndex{N;S}{##2}}\else
{\if =##1{\doIndex{B;S}{##2}}\else
{\if +##1{\doIndex{N;E}{##2}}\else
{\if |##1{\doIndex{N;F}{##2}}\else
{\if !##1{\doIndex{B;F}{##2}}\else
{\if ,##1{\doIndex{N;N}{##2}}\else
{\if ;##1{\doIndex{B;N}{##2}}\else
{\if *##1{\doAuthor##2}}\else
{\if +##1{\doCross##2}}\else
{\error}}}}}}}}}}}
```

```
\gdef\doAuthor #1;#2>{\doIndex{N;P}{#1, #2}}
\gdef\doIndex#1#2{\send1 {\angbr{\count7;#1;#2}}}
\gdef\doCross#1=#2>{\send1 {\angbr{#2;N;C;#1}}}
```

Index terms will be ignored before a call `'\openIndex foo'`, which opens the file `FOO.INDEX`. If no call to `\openIndex` appears, all indexing stuff will be ignored. We expect this to be the normal state--the `\openIndex` line will be added only when an index is being generated. The macros assume that there will be a counter (this version uses `\count7`) which will contain the page number. **WARNING!!** at the time the index terms are sent, the `\output` routine has already been executed, so if your `\output` routine bumps the page counter at the end (as many do), everything will be one off. Rewrite it to bump at the beginning.

The output file produced by these macros is read by the INTERLISP program. That program is included in full at the end of this memo since it is only a few pages long and since (as everyone knows) INTERLISP code is self-documenting.

Here are the TEX macros to format the index in the various styles.

```
% after \indexEntry comes the following:
% level{term and pages}{[crossrefs]<flag>[subterms]}
% <flag>="+" if have subterms,="-" if don't
```

```
% someone will \let\indexEntry = one of the following
% \indexLine for entry-per-line style
% \indexPar for paragraph style
% \indexComb for combined style
```

```
\gdef\indexLine#1#2#3{\setcount9#1\advcount9 by 2\noindent\hangindent\count9wd9 after
\hskip#1wd9\gdef\crossIndexDot{.}#2\indexTail#3}
```

```
\gdef\indexTail#1#2{#1\par}
% this hack puts out the crossrefs, discards the <flag>, does \par
% and leaves the subterms to be read next
```

```
\gdef\indexPar#1#2#3{\if 0#1{\indexPMain{#2}#3}\else
{\indexPSub{#2}#3}}
```

```
\gdef\indexComb#1#2#3{\if 0#1{\indexCTop{#2}{#3}}\else
{\if 1#1{\indexPMain{---#2}#3}\else
{\indexPSub{#2}#3}}
```

```
\gdef\indexCTop#1#2{\noindent\hangindent 1wd9 after
```

```

\gdef\crossIndexDot{.}#1\indexTail#2}

%\indexPMain and \indexPSub
%      #1=term&pages, #2=crossrefs, #3=subterms flag, #4=subterms

\gdef\indexPMain#1#2#3#4{\noindent\hangindent 1wd9 after 1
\gdef\crossIndexDot{.}#1\if-#3{\else{\gdef\firstSubEntry{T}:#4}#2\par}

\gdef\indexPSub#1#2#3#4{\if T\firstSubEntry{\gdef\firstSubEntry{F}}\else{;
}\gdef\crossIndexDot{.}#1\if-#3{\else{\gdef\crossIndexDot{.}\gdef\firstSubEntry{T}\
(\!#4)\gdef\crossIndexDot{.}}\let\indexSee=\indSee\let\indexAlso=\indAlso#2}

\gdef\indexSpan#1-#2.{\hbox{#1--#2}}
% this is for reference to a span of pages

\gdef\indexFF#1.{#1ff}
% this is for reference to a long span of pages
% "ff" stands for "following folios"

\gdef\indexN#1.{#1n.}
% this is for reference to footnote
% "n" stands for "note"

\gdef\pageNumDot{\gdef\crossIndexDot{}}
% this appears at end of pages for entry if they end with "."
% change \crossIndexDot to null so don't get double "."

\gdef\mainEntry#1{\bf #1}

\gdef\indexSee#1{\crossIndexDot\ {\it See }#1.}
\gdef\indexAlso#1{\crossIndexDot\ {\it See also }#1.}
\gdef\indSee#1{({\it See }#1)\gdef\crossIndexDot{.}}
\gdef\indAlso#1{({\it See also }#1)\gdef\crossIndexDot{.}}

\gdef\indexChar#1{\vskip 12pt} % can be changed if you want to put in letter headings
% parameter is capital letter for next section of index

\gdef\indexStart{} % change this to put in your own heading

\gdef\indexEnd{\par\vfill\ject} % needs to be more complex for multiple columns

```

**Important Note:** In addition to loading the previous macros, you must `\let\indexEntry=` either `\indexLine`, `\indexPar`, or `\indexComb` depending on which style of index you want. We have used a box and the "wd" parameter to make it easier to produce indentations that are multiples of a constant width. You must do `\save9` before you generate the index (e.g., `\save9\hbox{---}` is the right thing to do for the combined style). We have also made use of `\count9` for doing the arithmetic. If you use this counter (or `\box9`) for other purposes, you need to change these. The output can be put into multiple columns by modifying the `\output` routine as described in the TEX manual.

A file containing these macros can be found on SCORE `<tex.distrib>indexer.tex`.

## INTERLISP PROGRAM

The following pages contain the complete program. Any functions or constructs used in them that is not defined here can be found in the INTERLISP manual (October 1978 version). A file containing this program can be found on SCORE <tex.distrib>indexer.lsp.

```
(RECORD INDEXENTRY (UTERM TERM PAGES CROSSES SUBS))
(RECORD INPUTENTRY (PAGE BOLD TYPE . TERMS))
(RECORD PAGEREF (PAGEREFTYPE BOLD PAGENUM))
(RECORD SPANREF (PAGEREFTYPE BOLD PAGENUM ENDNUM))

(DEFINEQ

(INITIALIZE [LAMBDA NIL (* sets up readtable and creates empty index)
  (SETQ INDEXREADTABLE (COPYREADTABLE T))
  (for X from 1 to 127 do (SETSNTAX X (QUOTE OTHER)
    INDEXREADTABLE))
  (SETSNTAX (QUOTE <)(QUOTE LEFTPAREN) INDEXREADTABLE)
  (SETSNTAX (QUOTE >) (QUOTE RIGHTPAREN) INDEXREADTABLE)
  (SETSNTAX (QUOTE ;)
    [QUOTE (MACRO (LAMBDA (FL RDTBL) (RSTRING FL RDTBL)
    INDEXREADTABLE)
  (SETSNTAX (QUOTE %
)
  (QUOTE SEPR) INDEXREADTABLE)
  (SETQ WHOLEINDEX NIL))

(SPANREF? [LAMBDA (X) (* distinguishes SPANREFs from PAGEREFs)
  (CDDDR X)]

(SAMEREF? [LAMBDA (X Y) (* ignores BOLD and PAGEREFTYPE properties)
  (AND X Y (EQUAL (CDDDR X) (CDDDR Y))

(MAINENTRY? [LAMBDA (X)
  (COND
    ((EQUAL "B" (FETCH BOLD OF X)) T)
    (T NIL))

(CONVERTSPANTOPAGE [LAMBDA (X) (* deletes field for ENDNUM)
  (RPLACD (CDDDR X) NIL))

(READINDEX [LAMBDA (FILENAME) (* reads one index file, merging it into index)
  (PROG (INDEXFILE INPUT ENTRY TERM UTERM)
    (SETQ INDEXFILE (OPENFILE (PACKFILENAME (QUOTE BODY) FILENAME
      (QUOTE EXTENSION) (QUOTE INDEX))
      (QUOTE INPUT) (QUOTE OLD)))
    (WHENCLOSE INDEXFILE (QUOTE EOF) (FUNCTION FILEEND))
    (while (SETQ INPUT (READ INDEXFILE INDEXREADTABLE))
      do (SETQ TERM (CAR (fetch TERMS of INPUT)))
        (SETQ UTERM TERM)
        (SETQ TERM (MKATOM TERM))
        [COND
          ((NOT (SETQ ENTRY (GETPROP TERM (QUOTE ENTRY)
            (SETQ ENTRY (create INDEXENTRY TERM + TERM UTERM +(U-CASE UTERM))))
            (PUTPROP TERM (QUOTE ENTRY) ENTRY)
            (SETQ WHOLEINDEX (CONS ENTRY WHOLEINDEX)
            (PUTINDEX ENTRY INPUT (CDR (fetch TERMS of INPUT))

(FILEND [LAMBDA (FILE) (* handles end of file on input)
  (CLOSEF FILE)
  (RETFROM (QUOTE READ))

(PUTINDEX [LAMBDA (ENTRY INPUT SUBTERMS) (* puts one entry into index, checking for unmatched ends)
  (COND
    ((NOT SUBTERMS)
      (SELECTQ (MKATOM (fetch TYPE of INPUT))
        [P (replace PAGES of ENTRY
          with (CONS (create PAGEREF BOLD +(MAINENTRY? INPUT)
            PAGEREFTYPE + NIL PAGENUM +(fetch PAGE of INPUT))
```

```

      (fetch PAGES of ENTRY)
[F (replace PAGES of ENTRY
  with (CONS (create PAGeref BOLD +(MAINENTRY? INPUT)
    PAGEREFTYPE +(QUOTE F) PAGENUM +(fetch PAGE of INPUT))
    (fetch PAGES of ENTRY)
[N (replace PAGES of ENTRY
  with (CONS (create PAGeref BOLD +(MAINENTRY? INPUT)
    PAGEREFTYPE +(QUOTE N) PAGENUM +(fetch PAGE of INPUT))
    (fetch PAGES of ENTRY)
[C (replace CROSSES of ENTRY with (CONS (fetch PAGE of INPUT)
    (fetch CROSSES of ENTRY)
[S (replace PAGES of ENTRY
  with (CONS (create SPANREF BOLD +(MAINENTRY? INPUT)
    PAGEREFTYPE + NIL PAGENUM +(fetch PAGE of INPUT))
    (fetch PAGES of ENTRY)
(E (for P in (fetch PAGES of ENTRY) when (SPANREF? P)
  do [COND
    ((fetch ENDNUM of P) (UNMATCHED INPUT))
    ((EQ (fetch PAGENUM of P) (fetch PAGE of INPUT))
      (CONVERTSPANTOPAGE P))
    (T (replace ENDNUM of P with (fetch PAGE of INPUT)
      (RETURN)
    finally (UNMATCHED INPUT)))
  (SHOULDNT))
(T (PROG (SUBENTRY TERM UTERM)
  (SETQ TERM (CAR SUBTERMS))
  (SETQ UTERM (U-CASE TERM))
  [COND
    ((NOT (SETQ SUBENTRY (SASSOC UTERM (fetch SUBS of ENTRY)
      (SETQ SUBENTRY (create INDEXENTRY TERM + TERM UTERM + UTERM))
      (replace SUBS of ENTRY with (CONS SUBENTRY (fetch SUBS of ENTRY)
        (PUTINDEX SUBENTRY INPUT (CDR SUBTERMS))

(WRITEINDEX [LAMBDA (FILENAME NOELISION) (* writes entire index onto an output file)
  (* if NOELISION then dont elide second number in span)
  (bind (LASTCHAR NEXTCHAR (ACODE +(CHCON1 (QUOTE A)))
    (ZCODE +(CHCON1 (QUOTE Z)))
    (TEXTFILE +(OPENFILE (PACKFILENAME (QUOTE BODY) FILENAME
      (QUOTE EXTENSION) (QUOTE TEX))
      (QUOTE OUTPUT) (QUOTE NEW)))
    (OLDLEN +(LINELENGTH 1500)))
  first (printout TEXTFILE "\indexStart") for ENTRY in (SORT WHOLEINDEX T)
  do (SETQ NEXTCHAR (CHCON1 (fetch UTERM of ENTRY)))
  (COND
    ((OR (EQ NEXTCHAR LASTCHAR) (LESSP NEXTCHAR ACODE)
      (LESSP ZCODE NEXTCHAR))
    NIL)
  (T (SETQ LASTCHAR NEXTCHAR)
    (TERPRI TEXTFILE)
    (printout TEXTFILE "\indexChar{" (CHARACTER NEXTCHAR) "}"))
  (WRITEENTRY 0 (fetch TERM of ENTRY) ENTRY)
  finally (TERPRI TEXTFILE) (printout TEXTFILE "\indexEnd" T)
  (CLOSEF TEXTFILE) (LINELENGTH OLDLEN) (CLEANINDEX))

(CLEANINDEX [LAMBDA NIL (* after finish writing index)
  (for ENTRY in WHOLEINDEX do (REMPROP (fetch TERM of ENTRY) (QUOTE ENTRY))
  finally (SETQ WHOLEINDEX NIL))

(WRITEENTRY [LAMBDA (DEPTH TERM ENTRY)
  (* writes a top-level entry along with its subentries.
  suppresses duplicates and checks for unmatched begins)
  (PROG (REFJUSTPRINTED CROSSREFS NORFF)
    (TERPRI TEXTFILE)
    (printout TEXTFILE "\indexEntry" DEPTH "{" TERM)
    [COND
      ((fetch PAGES of ENTRY)
      (for REF in (SORT (fetch PAGES of ENTRY) (FUNCTION REFBEFORE))
        when (NOT (SAMEREF? REF REFJUSTPRINTED))
        do (printout TEXTFILE "," (COND
          ((fetch BOLD of REF) " \mainEntry{"
            (T " ")))
          (SETQ NORFF NIL)

```

```

(COND
  ((SPANREF? REF)
    (COND
      ((NOT (fetch ENDNUM of REF)) (UNMATCHED (LIST TERM REF)))
      (T (ELIDESPAN (fetch PAGENUM of REF) (fetch ENDNUM of REF)
        ((EQ (fetch PAGEREFTYPE of REF) (QUOTE F))
          (printout TEXTFILE "\indexFF " (fetch PAGENUM of REF) ".")
          (SETQ NORFF T))
        ((EQ (fetch PAGEREFTYPE of REF) (QUOTE N))
          (printout TEXTFILE "\indexN " (fetch PAGENUM of REF) ".")
          (SETQ NORFF T))
        ((NULL (fetch PAGEREFTYPE of REF))
          (printout TEXTFILE (fetch PAGENUM of REF)))
        (T (SHOULDNT)))
      (COND
        ((fetch BOLD of REF) (printout TEXTFILE "]))))
      (SETQ REFJUSTPRINTED REF))
    (COND
      (NORFF (printout TEXTFILE "\pageNumDot")
        (printout TEXTFILE "){{"")
      (COND
        ((fetch CROSSES of ENTRY)
          (SETQ CROSSREFS (SORT (fetch CROSSES of ENTRY)))
          (printout TEXTFILE "\index" (COND
            ((OR (fetch PAGES of ENTRY) (fetch SUBS of ENTRY)) "Also{"")
            (T "See{"")
          (CAR CROSSREFS))
          (SETQ REFJUSTPRINTED (CAR CROSSREFS))
          (for REF in (CDR CROSSREFS) when (NEQ REF REFJUSTPRINTED)
            do (printout TEXTFILE ", " REF) (SETQ REFJUSTPRINTED REF))
          (printout TEXTFILE "}")
        (printout TEXTFILE "}")
      (COND
        ((NOT (fetch SUBS of ENTRY)) (printout TEXTFILE "-{"))
        (T (printout TEXTFILE " + {"")
          (for ENT in (SORT (fetch SUBS of ENTRY) T)
            do (WRITEENTRY (ADD1 DEPTH) (fetch TERM of ENT) ENT))
          (printout TEXTFILE "}")
        (printout TEXTFILE "}")
    (ELIDESPAN [LAMBDA (X Y) (* print elided form of span)
      (COND
        ((OR NOELISION (ZEROP (REMAINDER X 100))
          (NEQ (QUOTIENT X 100) (QUOTIENT Y 100)))
          NIL)
        (T (SETQ Y (REMAINDER Y 100)
          (printout TEXTFILE "\indexSpan " X "-" Y "."))
    (REFBEFORE [LAMBDA (X Y)
      (* ordering function used to sort page references.
        single pages are before spans and bold before non-bold if page is the same)
      (COND
        ((EQ (fetch PAGENUM of X) (fetch PAGENUM of Y))
          (COND
            ((SPANREF? X)
              (COND
                ((SPANREF? Y)
                  (printout T "TWO SPANS START TOGETHER" T X T Y))
                  (T NIL)))
            ((SPANREF? Y) T)
            (T (fetch BOLD of X)
              (T (LESSP (fetch PAGENUM of X) (fetch PAGENUM of Y))
    (UNMATCHED [LAMBDA (INPUT) (* prints error message on terminal)
      (printout T "UNMATCHED ENTRY" T INPUT)
    )
  )
  STOP

```



# NOTICE



**Panel Discussion on *AMS-TEX***

**Friday, January 9, 1981, 4:30 p.m.**

**Continental Ballroom, San Francisco Hilton**

**At the 87th Annual Meeting of the American Mathematical Society,  
a session will be devoted to a discussion of *AMS-TEX* (see page 10).**

**Panelists:**

**Donald Knuth  
Robert Morris  
Richard Palais  
Arnold Pizer  
Michael Spivak**

### Contents

October 1980

Robert Welland. <i>Editor's Comments</i> . . . . .	2
<b>General Delivery</b>	
Richard Palais. <i>Message from the Chairman</i> . . . . .	3
Ellen Swanson. <i>Publishing &amp; T<sub>E</sub>X</i> . . . . .	7
Michael Spivak. <i>AMS-T<sub>E</sub>X—"A Very Friendly Product"</i> . . . . .	10
Robert Morris. <i>Minutes of the TUG Meeting</i> . . . . .	12
<b>Interface Software</b>	
Ignacio Zabala and Luis Trabb-Pardo. <i>The Status of the Pascal Implementation of T<sub>E</sub>X</i> . . . . .	16
David Fuchs. <i>The Format of T<sub>E</sub>X's DVI Files</i> . . . . .	17
Thea Hodge. <i>University of Minnesota CDC Site Report</i> . . . . .	19
<b>Warnings &amp; Limitations</b>	
Barbara Beeton. <i>Troubles with Trace and Other Oddities</i> . . . . .	20
<b>Macros</b> . . . . .	20
Terry Winograd and Bill Paxton. <i>An Indexing Facility for T<sub>E</sub>X</i> . . . . .	Appendix A
Michael Spivak. <i>The Joy of T<sub>E</sub>X</i> . . . . .	order form
<b>Questions &amp; Answers</b> . . . . .	21
<b>Letters</b> . . . . .	21
Gérard Emch and Arnold Pizer . . . . .	22
<b>Miscellaneous</b> . . . . .	21
Order Form for <i>The Joy of T<sub>E</sub>X</i>	
T <sub>E</sub> X Errata	
TUG Mailing List	