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 TEXarcana.

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

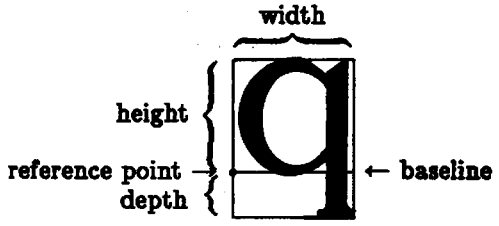
I am sure many early TEX users have had the experience of watching with surprised delight as their first pages came off the printer and thinking, "Did I really set that myself!" And on reflecting, we knew that while we did, and while we had reason to be proud of our new skill, in a very deep sense Don 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 TEX and implementations at other sites. All these users have to do if they want to interchange documents written in 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 TEX called *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 *AMS*-TEX, and the author of its manual, *The Joy of* TEX, reports on it later in the newsletter.

(3) All of this doesn't answer one of the main questions a potential TEX user will be interested in. How difficult is it to learn to use 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 TEX a paper he needed in a hurry. Pizer was the driving force behind getting TEX running on the TOPS-10 system at the University of Rochester and is very 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 TEX and the experimental version of *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.

TEX as more than a Production Typesetter

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 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 *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 $\mathcal{AMS}$-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 utilised 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