My TEX quiz question is:

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

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

Paragraph with dropped initial

Consider:

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

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

Question:

How can TEX be made to do this generally?

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

* * * * * * * * * * *

Letters et alia

* * * * * * * * * * *

OBSERVATIONS ON TEX FROM A DIVERGENT VIEWPOINT: A CRITICAL COMMENTARY

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

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

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

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

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

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

Sam Whidden

* * * * * * * * * * *

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

TO: STI Staff

FROM: J. R. Roesser

SUBJ.: TEX Users Group Meeting, July 1983

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

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

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

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

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

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

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

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

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

End of digression.

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

Now for some high points.

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

   Need I continue? This is rubbish.

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

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

Some random comments.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6. Table capabilities are primitive.

7. User manuals are only usable by programmers. In general my impression is that TEX was written by programmers for other programmers.
8. TEX does not have the auxiliary programs which help make the process of composition easier. Consider how much more difficult our task was before we had the check program, fotrun, and the ability to set patches.

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

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

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

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

WHAT YOU SEE IS WHAT YOU GET

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

Comments on Jim Roesser's Memo

Don Knuth

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

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

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

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

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

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

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

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

* * * * * * * * * * *

J. R. Roesser's Memo on TEX Meeting

David Fuchs

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Well, what if Prof. Knuth is hit by a train? Bugs in TeX will still not be a problem, because of the unprecedented internal system documentation. All of our source code is available to interested parties, and in fact some of the bug reports we have received have been of the form "in module vvv line www, you should change xxx to yyy, to avoid a bug when zzz". Of course, the average user should never have to look at the program itself, but it is quite a relief to know that it's there if needed. In fact, a number of large companies are using TeX in-house and TeX82 can't be available soon enough for them. They are not worried about support because they have looked at the code of TeX82 and are confident that it is maintainable, flexible, and efficient. Initially they became interested in TeX because of Prof. Knuth's reputation in the computer community, and they are happy with our track record to date for fixing bugs, but they also know that their own programmers are quite capable of delving into the system in a pinch.

At least four companies are coming out with TeX-based products, including such large firms as Hewlett-Packard and such up-and-coming concerns as Intergraph. They know they can support a program like TeX on their own, or they would not be spending the large amount of money necessary to bring these products to market. Right now, anyone who wants traditional support for TeX can buy an HP9836 computer with the TeX package, and they'll get it from HP. Note that TeX82 isn't even officially released yet; I expect more companies to pick it up as a product in the future.

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

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

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

* * * * * * * * * * *

Remarks on the STI report

Michael Spivak

Here are some remarks on the STI report.

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

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

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

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

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

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

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

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

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

```
$$@bx↑a+@by↑b$$
```

would yield

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

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

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

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

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

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

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

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

$$\big(x - f(x)\big)\big(x + f(y)\big)$$

which almost always appear in journals in the less pleasing form

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

and `\bigg` is for things like

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

which looks better than

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

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

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

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

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

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

* * * * * * * * * * *

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

Richard S. Palais

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

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

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

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

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

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

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

Comments on Jim Roesser's Memo

Barbara Beeton

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

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

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

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

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

* * * * * * * * * * *

A Reply to the Replies

by J. R. Roesser

To Donald Knuth

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

To Michael Spivak

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

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

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

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

To David Fuchs

1) The purpose of my memo was to inform the STI staff about TEX. The fact that so much time was spent by Mr. Spivak on that particular discussion of macros is an indication of the level reached in the TEX development.
2) The extra space at the end of sentences is not in general use by publishers. It complicates things because when in use one must remember to key special spaces after initials and abbreviations.
3) If you are interested I will send you a copy of the test which we use for intercharacter spacing. If TEX gives equally good results I will withdraw the objection.

   Your statements about a lack of adverse comment from AMS are easily explained when one realizes that their math production is on the STI program.
4) "... can you imagine telling a *mathematician* or ... to use `*gq` instead of `\theta`; `*n7` instead of `\notin`?" Yes. As a physicist I find it simple and logically complete. The asterisk in all cases indicates a change of font. The `g` tells that the new font is Greek (or the `n` (for not) indicates a "canceled" symbol font). Finally the `q` or `7` refers to a specific character. If I forget the code I can look at a single sheet where all symbols are grouped in a logical way.

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

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

   In my opinion the impediment to good communication is the idea that each author should be free to design his own input language. A good language (macro package) requires time and control to provide efficiency and completeness. (Obviously it will never be totally complete.) The beauty of the STI approach is that our language is now complete enough that we are able to satisfy virtually all of the authors with which we deal.
5) You refer to "features deep down in the STI program". I am sorry if I gave you that impression. (Again, the staff at STI for whom the memo was written know that we can alter the program at will to satisfy authors—though only when we have the publisher's approval.) The difference in practice between changing TEX macros and changing "features deep in the STI program" is mostly one of semantics.

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

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

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

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

To Barbara Beeton

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

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

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

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

To Richard S. Palais

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

3

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

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

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

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

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

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

Jim Roesser's Memo and Responses – Wrapup

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

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

Some features of the STI system are very attactive in a production environment: (1) A very wide variety of fonts is available, potentially any font in the typesetter manufacturer's library; those manufacturers will have to become persuaded that TEX users constitute a lucrative market before such a situation will exist for TEX. (2) STI pagination, although designed for manual intervention after all line and paragraph decisions are complete, does allow a page to be made shorter or longer than the norm to avoid very short final pages; it is not particularly easy in TEX to force one excess line onto the previous page. (3) The STI program usually keeps going, regardless of the severity of input errors, allowing output to be generated for proofing; with TEX, a keyboarder may tempted to stop for every error, and reprocess the file from the beginning. We still have insufficient experience with TEX82's error interrupts to know whether the number of runs per file will be decreased substantially from our TEX80 experience.

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

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

* * * * * * * * * * *

Late-Breaking News

* * * * * * * * * * *

The following article by Mike Spivak gives details of how to use the TEX82 version of the AMS-TEX macro package, as he presented them at the Beginners' Course held in conjunction with the July TUG meeting. This article wasn't really late, but is too large to fit in any other column.

Mike's article is published here by permission of the American Mathematical Society; it will also be published as a supplement to the reprint of *The Joy of TEX*, version 0, which still describes AMS-TEX for TEX80. *Joy* for AMS-TEX82 will not be ready for at least several months.

In the meantime, Mike has requested that any comments on AMS-TEX82 be sent to him in writing, c/o the American Mathematical Society, P.O. Box 6248, Providence, RI 02940. Please make a distinction between TEX and AMS-TEX questions, asking your local TEXnician for assistance first, if possible, and send only AMS-TEX questions to Mike.

The real news is, this article is the first output from TEX82 at the AMS to be published in TUGboat. It was manufactured using macros and an AMS-TEX manual 'style' file created by Mike, with a bit of tinkering to apply the TUGboat running heads. With luck, time and persistence, it may actually be possible for the next issue of TUGboat (that's in 1984) to be prepared with TEX82.

Barbara Beeton