[Bry88] Martin Bryan. *SGML: An Author's Guide to the Standard Generalized Markup Language.* Addison-Wesley, Woking, England; Reading, Massachusetts, second edition, 1988.

[CRD87] James H. Coombs, Allen H. Renear, and Steve J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, November 1987.

[Det91] Christine Detig. TeX & Hypertext — The future of electronic publishing. In Guenther [Gue91], pages 8–12. TUGboat 12 (March 1991) Number 1.

[Gol90] Charles Goldfarb. *The SGML Handbook.* Clarendon Press, Oxford, 1990.

[Gue91] Mary Guenther, editor. *TeX90 Conference Proceedings; University College; Cork, Ireland, September 10–13, 1990*, Providence, Rhode Island, U.S.A., March 1991. TeX Users Group. TUGboat 12 (March 1991) Number 1.

[Her90] Eric van Herwijnen. *Practical SGML.* Kluwer, Dordrecht, NL, 1990.

[Laa91] C.G. [Kees] van der Laan. SGML(, TeX and ...). In Guenther [Gue91], pages 90–104. TUGboat 12 (March 1991) Number 1.

[Pop91] N.A.F.M. Poppelier. SGML and TeX in scientific publishing. In Guenther [Gue91], pages 105–109. TUGboat 12 (March 1991) Number 1.

[SC88] A. Scheller and C. Smith. *DAPHNE; Document Application Processing in a Heterogeneous Network Environment; Dezentrale Verarbeitung von Dokumenten auf der Basis von SGML; Benutzeranleitung (Version 3.0).* GMD-FOKUS, Berlin, im Auftrag des Vereins zur Förderung eines deutschen Forschungsnetzes e.V., April 1988.

[Won90] Reinhard Wonneberger. Structured document processing: the LaTeX approach. In J. Nadrchal, editor, *Man-Machine Interface in the Scientific Environment. Proceedings of the 8th European Summer School on Computing Techniques in Physics. Skalský Dvůr, Czecholsovakia, 19–28 September 1989*, volume 61 of *Computer Physics Communications*, pages 177–189. North Holland Publishing Company; Elsevier Science Publishers B.V., 1990.

[Won92] Reinhard Wonneberger. Approaching SGML from TeX. *TUGboat* 13(3):223 (July 1992).

[Wonng] Reinhard Wonneberger. TeX in an industrial environment. In Anne Brüggemann-Klein, editor, *Proceedings of the 4th European TeX Conference, September 11th–13th, 1989*, Karlsruhe, forthcoming.

◇ Reinhard Wonneberger and
Frank Mittelbach
EDS Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56 (N15)
D-6090 Rüsselsheim
Federal Republic of Germany

Dreamboat

Editor's note: This column heading hasn't appeared for years, but it seemed an appropriate corner in which to collect ideas and suggestions related to the topic "Where do we go from here?" In addition to the following articles, which were written before the formal recognition of interest in future directions, Philip Taylor has reported in this issue (p. 138) on the first meeting of the working group coordinating the discussion.

TeX wish list

Michael Barr

It is the rare user of TeX who has not, at some time, felt that TeX lacks some feature or other. Since Knuth has announced that TeX is now frozen, save for an occasional bug fix, it is up to the TeX community to give thought to the kinds of features that we want in any successor to TeX.

I do not expect that my wish list will be exhaustive or that the future program will implement every one of my suggestions. I am merely trying to start a dialog on the kind of program we want in the future.

Let me say a few words about what I don't want. I don't expect to see a WYSIWYG program, although a multitasked previewer would be nice. I don't expect to see a page layout program. In fact, I don't want to think about page design at all. Ideally, futureTeX will take care of all design details itself. It is a *tour de force* to lay out *TV Guide* in TeX, but TeX is not the tool I would have chosen for the job.

Here are some of the things that I have felt lacking in TeX, in no particular order. I divide them into two groups, depending on whether or not they could be made compatible with current device drivers. The reason is that there is basically only one TeX program, but as many device drivers, and more, as there are devices. Thus the amount of work that is involved in upgrading the latter is orders of magnitude larger than that which is involved in upgrading TeX itself.

Features that could be implemented without changing device drivers

A smart `\put`. By a smart `\put`, I mean a procedure similar to the `\point` defined on page 389 of *The TEXbook*, but one that would set the width of the box properly. If you actually try that procedure, you will find that the box has zero width. The height and depth are set to the actual height and depth, but not the width. No variation I tried was able to do it either.

The reason I consider it important is that I use LATEX's picture mode extensively for commutative (and even non-commutative) diagrams, and you have to tell picture mode exactly what dimensions your picture is. What nonsense! TEX is smart enough to figure out how large your picture is, isn't it? Well, yes it is, but not at any great speed. I don't know what design consideration caused Leslie Lamport to implement `\picture` mode as he did, but it is entirely possible that it was the long time it took for a picture to work out its own size. If this were implemented in the program, it would take a fraction of the time. For my own macros, I have reimplemented both `\put` and `\picture`. However the compilation of a diagram of any complexity takes a long time. A page with even one complicated diagram takes an appreciable part of a minute (on my 16 Mh 386SX computer).

Implicit in this point is that there should be a built-in picture mode. It would be faster and more reliable than the LATEX `\picture` procedure. By the way, although it is not an important point, Lamport erred in having his coordinate system use the mathematician's orientation. TEX is a typesetting program and to a typesetter the positive y direction is down, not up. I find it a real nuisance to think upside down when drawing a complicated diagram.

More reliable program control. This rubric covers so many different things that I hardly know where to start. Take the entire appendix D of *The TEXbook* and ask yourself why most of them should require dirty tricks? Most of them are quite reasonable things and it is a mystery to me why you should have to resort to dirty tricks to do reasonable things. Take the discussion of trying to place `\n` stars on a page, where `\n` is an integer variable. *Why* is this so hard to do? It is, after all, a perfectly reasonable thing to want to do; why shouldn't the language provide a way to do it straightforwardly? I know that Knuth is exceedingly clever, much cleverer than I, but why didn't he design a language that I could program in? The June 1991 issue of *TUGboat* had no fewer than three new implementations of procedures for outputting `\n` asterisks, and each of them was based on some clever trick.

I suspect that one of the problems is that Knuth didn't at first think of TEX as a programming language. This seems even clearer if you look at TEX78. It was so deficient that you couldn't `\advance` a numeric variable, only increment or decrement it. Imagine how hard it would be to implement LATEX in that language!

I am getting indigestion hearing about TEX's digestive tract. The discussion of `\expandafter` is ludicrous. Both `\expandafter` and `\noexpand` ought to be able to take an entire brace-delimited phrase as argument, not just a single control sequence. Moreover a new control sequence `\expand` ought to be provided, preferably with a second, optional, parameter that tells how many levels of expansion are wanted, since in many cases you want only one level of expansion, not to the very bottom. More generally there ought to be a simple mechanism by which the user can specify when a control sequence should be expanded. For example, `\expandafter` is what in FORTH would be called an immediate control sequence; it controls compilation. The user should have the ability to define his own "immediate" control sequences as well as ways of overriding this specification (it is often necessary to override the immediate specification when defining a new immediate word).

Better arithmetic. This includes the ability to use numeric expressions as arguments and having real number registers. I have been told that the reason for the lack of the latter is that Knuth didn't want the user to have any access to the underlying floating point. Why should TEX use floating point arithmetic at all? Wouldn't everything be faster if everything were in fixed point? I thought all distances were in scaled points anyway and a scaled point is smaller than one wavelength of visible light.

As for using expressions as arguments, almost anyone who has ever used a macro has had to write complicated procedures because you couldn't give, say, `\hsize-10pt` or similar expressions involving counters as arguments. At one time, this lacuna was justified on the grounds that TEX was to run in as small a memory as possible, but this is no longer a valid reason.

Successive super and subscripts. This seems like a picky point, but in my work it comes up surprisingly often. I refer, in the first instance, to the fact that you cannot say `x_1_2` for `x_{12}`. Why not? They are logically equivalent. To see what pain even Knuth had to go through on this point, see the definition of the `\prime` operator. I have an operator `\op` defined as `{{}^{op}}` and the initial brace pair is there to avoid running into the "double" superscript error. But it also means that it doesn't work properly if there is a subscript on the same symbol. Surely a simple parser could interpret double superscripts properly.

More reliable global page procedures. I was recently unable to get marks to work right in the twocolumn environment of either the macros supplied by LaTeX or those of Frank Mittelbach. Footnotes are not reliably placed by Mittelbach's style either. It is not clear if, in the present version of TeX, it is possible to combine a multicolumn style that allows changing the number of columns in the middle of a page with proper placement of footnotes and marks. I don't use inserts, but virtually everyone who does complains that they don't work as expected. Changebars have proved extremely difficult to implement reliably. Someone wrote to TeXhax several months ago asking if it was possible to leave a 2 by 2 inch box blank in a lower corner of each page. So far as I know, it can't be done, except perhaps by some sort of cut and try procedure similar to that of the column balancing on page 387 of *The TeXbook.*

More control over tfm's. The internal variables pertaining to a whole font can be changed, but not, as far as I am aware, those for single characters. I have occasion to use fairly frequently the notations d^0 and d^1. I do not know how these would look in the family used to print *TUGboat*, but in the cmmi font the first of these comes out with the top of the d running into the 0. Since the 1 is thinner, this doesn't happen. And of course, as it happens, d is one of only three characters in the lowercase Roman alphabet that have an ascender sticking out that far to the right (l and f being the others). If I understand rules 17 and 18 of page 445 of *The TeXbook* correctly, an italic correction is added between a character and a superscript. But the italic correction is set globally in a font and it seems clear that a bit more is needed for those three letters when they have a superscript.

A completely different example is provided by my experience in making a minus sign with a dot on it. Try as I might, I could not get the dot low enough. Eventually, I asked TeXhax and got an answer from Barbara Beeton. For some reason Knuth gave all the standard arithmetic operators the same height as the largest, which is probably the plus. The result is that a dot on the minus comes out at the same height as it would on a plus and, of course, looks awful. The definition I now uses `\smash` and then gives the minus sign the (completely arbitrary, as far as I am concerned) height of 0.55`ex`. My feeling is that what Knuth did was an error in judgment, but that is not my point here. If the user had control over these things, then the height of the minus could have been left at its natural height and defined as being the height of the plus any time that was needed. The reason I think Knuth was in error is that you can make a box containing the minus whose height is that of the plus, but given that the tfm entry for the minus gives it the height of the plus, there is no way of getting its natural height back. You simply have to guess a number like 0.55`ex`, which is bad for a number of reasons. It might be wrong, it might not be correct in a different sized font, depending on how that size was selected and it might not be right in a different family.

Features that require new device drivers

Diagonal rules. Traditional typesetting didn't have anything like diagonal rules, but it would be extremely helpful if TeX went beyond traditional typesetting here. To some extent, the LaTeX line fonts compensate for this, but only partly and unsatisfactorily. First off, the number of different slopes is severely limited. Only 26 slopes are allowed (including horizontal and vertical) and arrowheads are available at only 14 of them. This isn't so limiting; what is more serious is the fact that the shortest segment available at any oblique slope is much too long. I have been trying to implement diagonal dashed lines (and arrows), but the shortest segments available are much too long and it will have to be done with dots. This is inefficient both in time and in memory.

Opaque boxes. It doesn't come up often, but every once in a while I feel the need to be able to place one box opaquely over another. I don't even know if this is possible in either HP printer control language or PostScript, but it would be awfully handy if it were. One example of where this could be used would be if one box had an arrow and a second had a label for that arrow in a suitably sized box that you wanted to cover part of the arrow.

Documentation

I find *The TeXbook* pretty good for the most part, but people unused to programming mostly find it impenetrable. But the story for LaTeX is much worse. It has seriously retarded the adoption of LaTeX as a standard. Several of my colleagues tell me they won't use LaTeX because 'using LaTeX you can't do X'. In every case, you can do X often more easily than you can in plain. But it is not documented anywhere. Our office staff mostly use plain TEX because they find the LATEX book so uninformative. As difficult as they find *The TEXbook*, they feel they can eventually get the information out of it, but it just isn't there in the LATEX manual. Of all its deficiencies, the worst is the paucity of examples. The situation is somewhat better in French and German, and one of our secretaries makes good use of Raymond Seroul's book, *Le petit Livre de TEX* [InterEditions, 1989, ISBN 2-7296-0233-X]. A somewhat expanded version, by Raymond Seroul and Silvio Levy, has now appeared in English: *A Beginner's Book of TEX* [Springer Verlag, 1991, ISBN 0-387-97562-4]. Leaving all other considerations aside, I consider LATEX far superior to plain because it encourages you to think of a document in logical, not page layout terms. The criticisms of the diagram mode and `\put` above are precisely because they are such a departure from that ideal. Lamport actually suggests laying your diagrams out on graph paper before entering them. This is absurd. I have coauthored two books using TEX and they each include several hundred diagrams.

Conclusions

I have sucessfully used TEX for books, papers and even routine letters. I find it much easier to use than the standard text processors. Nonetheless, I find it has some deficiencies. Since Knuth has decided that TEX will remain static, the time has come to think of a possible successor. I have set out above some of the possible directions in which change might come. Some of them might be done by a few modifications to the language that would leave the dvi output format unchanged. These could be accomplished by modifications to the underlying language, but would leave all device drivers and previewers current. However, some of the changes would require new device drivers which would render many of our auxiliary tools obsolete.

When TEX was written the computing power available to the average user was much less. Freed from such limitations, we can now hope for a language that is a lot more powerful and easier to use. I hope to see a successor to TEX that is worthy of its predecessor.

Since the first draft of this paper was written, there has been a new development. A formal network, called NTS-L ("New Typesetting System List") has been set up to discuss the question of a successor to TEX. All issues are up for discussion. Should this new language be an incremental improvement to TEX or a new beginning? Should it be upward compatible? Should it be aimed at microcomputers or only for workstations and larger? Even, should it make a pass at being WYSIWYG? The debate is wide-ranging and sometimes heated. Anyone interested should subscribe. Send email to
listserv@vm.urz.uni-heidelberg.de
with a one line message
subscribe nts-l ⟨Your Name Here⟩.

◇ Michael Barr
Department of Mathematics and Statistics
McGill University
Montreal, Quebec, Canada
`barr@math.mcgill.ca`

Approaching SGML from TEX

Reinhard Wonneberger

Abstract

The present memorandum intends to encourage discussion on a pragmatic TEX approach to SGML.

It assumes a basic knowledge about SGML and builds on [WM92], which also contains bibliographic information.

Comments and contributions are welcome.

Situation

§ **1** *Concern*

Although TEX has become a *de facto* standard by now, the corresponding General Markup language LATEX cannot claim to be a standard.

This implies severe limitations in using TEX outside the academic world.

Such limitations might be overcome by combining TEX with an accepted General Markup standard, which seems to be SGML.

§ **2** *καιρός (time of opportunity)*

The present development project of a new LATEX gives the unique chance to introduce a new Markup Language instead of staying frozen in upward compatibility.

§ **3** *Conclusion*

The community of TEX users, esp. the implementors and other wizards, are encouraged to think about