The Future of TEX*

Philip Taylor

Royal Holloway and Bedford New College, University of London, Egham Hill, Egham, Surrey, UK
p.taylor@vax.rhbnc.ac.uk

July 1992

Abstract

TEX and the other members of Knuth's *Computers & Typesetting* family are arguably amongst the most successful examples of computer software in the world, having been ported to almost every conceivable operating system and attracting an allegiance that verges on the fanatical. Development work on this family has now ceased, and many members of the computer typesetting community are concerned that some action should be taken to ensure that the ideas and philosophy enshrined in TEX are not allowed simply to fade away. In this paper, we discuss some of the options available for perpetuating the TEX philosophy, and examine the strengths and weaknesses of the present TEX system. We conclude by postulating a development strategy for the future which will honour both the letter and the spirit of Knuth's wish that TEX, METAFONT and the Computer Modern typefaces remain his sole responsibility, and at the same time ensure that the philosophy and paradigms which are the strengths of TEX are not lost for ever by having artificial constraints placed on their evolution.

Keywords: TEX, extended TEX, NTS, New Typesetting System

"My work on developing TEX, METAFONT and Computer Modern has come to an end." [1] With these words, Professor Donald E. Knuth, creator of TEX, informed the world that the evolution of probably the most successful computer typesetting system yet developed had ceased, and that with the sole exception of essential bug fixes, no further changes would be made. TEX's version number will asymptotically approach π as bug fixes are made, and at the time of his death, it will be renamed 'TEX, Version π'; thereafter it will remain exactly as he last left it: a fitting and appropriate memorial to one of the most productive and inspired computer scientists (and mathematicians, and Bible scholars) that the world has ever known.

The future of TEX is therefore totally determined: why, then, is this paper entitled "*The Future of TEX*"? Because, primarily, TEX is already fifteen years old — four years as a child (TEX 78); eight years as an adult (TEX 82); and three years in maturity (TEX 3). Fifteen years is a long time in the lifespan of computer languages: Algol 68, for example, was certainly at or beyond its peak by 1982, and is today almost as rare as the Tasmanian wolf,[1] if not yet as dead as the Dodo:[2] a language must evolve, or die. (There are numerous natural languages which are almost certainly in terminal decline, despite the most strenuous efforts of a nucleus of active speakers to artificially prolong their lives: Cornish and Manx are surely dead; Gaelic must feature in any linguistic 'Red Book' of endangered languages; only Welsh, which alone among the British native minority tongues continues to evolve, shews any real resistance to morbidity and eventual death). If natural languages must evolve or die, how much more so must computer languages, whose evolution must keep pace with a technology which evolves at a rate so rapid that it is unmatched in the natural world even by irradiated fruit-flies.[3]

So, my underlying hypothesis is: TEX must evolve, or die. If we are to believe the evidence of our ears and eyes, the underlying TEX philosophy is already as anachronistic as the horse and cart: TEX represents the pinnacle of Neanderthal evolution, building on the genetic heritage of Runoff, Nroff, Troff, Ditroff and Scribe, whilst Cro-Magnon man, in the guise of Ventura Publisher, Aldus Pagemaker and Quark Xpress, is already sweeping over the face of the planet. The halcyon days are long since gone (or so it would seem)

*This article is reproduced by kind permission of the organisers of the EuroTEX '92 conference in Prague, Czechoslovakia, September 14–18, in the proceedings of which [4] it first appeared.

1 *Thylacinus cynocephalus*
2 *Raphus cucullatus*
3 *Drosophila melanogaster*

when it was socially acceptable to: enter text; check it for spelling errors (by eye!); insert a series of formatting commands; pass the whole through an interpreter; identify the first error; correct the first error; pass the whole through the interpreter again; identify the second error; correct the second error; pass the whole through the interpreter for a third time; repeat for all subsequent errors. . . ; pass the whole through the interpreter for the n^{th} time; then pass it through the interpreter again (to resolve forward- and cross-references); preview a facsimile of the final copy on the computer screen; notice a formatting error; and go right back to editing the file: our colleagues sit there clicking away on their mice[4] like demented death-watch beetles[5] and think us totally mad; and mad we surely must be, for we not only enjoy this mode of working, we seek to convert the demented mouse clickers into TEX users as well!

Why? What is it about TEX that is so totally addictive? Is it perhaps TEX's descriptive and character-oriented nature —the fact that, in direct opposition to current trends, TEX requires the user to think about what he or she wants to achieve, and then to express that thought as a series of words and symbols in a file, rather than as a series of ephemeral mouse movements on a screen? Is it, perhaps, its portability —the fact that implementations (almost entirely public domain) exist for every major operating system in the world? Is it the deterministic nature of TEX —the fact that a given sequence of TEX commands and text-to-be-typeset will always produce *exactly* the same results, regardless of the machine on which it is processed? Is it the 'boxes and glue' paradigm, which provides a simple but somewhat naïve model of black and white space on the printed page? The ease with which form and content can be separated? The implementation as a macro, rather than a procedural, language? (would a procedural TEX still be recognisably TEX?) Is it, perhaps, the incredible contortions through which one occasionally has to go to achieve a desired result? (Or the incredible elation when such contortions finally achieve their intended effect?) How many of these elements could be eliminated and still leave something that is recognisably TEX? I propose to return to these questions, and to attempt to answer some of them, later in this paper.

A related question: what is the potential lifespan of a TEX-based typesetting system, or for that matter, of any computer language? Of all the general purpose computer languages which have sprung into existence since the advent of compilers (which point in time really marks the beginning of all the computer languages that are in general use today), Cobol and Fortran are probably among the longest lived; but Fortran has evolved enormously since the days of Fortran 2 (which is as far back as my memory goes), whilst Cobol has evolved relatively little; Basic, too, is still with us, although the originators of Dartmouth Basic would find little to recognise in the 'Visual Basic' of Microsoft today. Algol 60 evolved via various routes into Algol 68, which for me represents the pinnacle of language design, but evolved no further, and is today reaching the end of its twilight years. Pascal, which owes much to the Algol family, gave birth to Modula, which itself became transmuted into Oberon; in a sense, this last example represents a failure of the evolutionary system, for in its heyday Pascal was almost universally adopted, giving birth to the UCSD 'P' system as well as making possible the unbelievably successful (and revolutionary) 'Turbo Pascal', whilst Modula, although lauded by computer scientists, remained of relatively limited acceptance and acceptability, and Oberon remained almost unknown without the walls of academia. Most recently, among the procedural languages at least, we come to 'C', and its bastard offspring 'C++'; these languages have an honourable history, tracing their roots back through 'B' (or so I am told —I have never encountered 'B' myself) to BCPL, the 'Basic Combined (or Cambridge, depending on one's background) Programming Language', itself derived from CPL which simply wasn't so basic! *En route*, data typing was acquired, and lost, and acquired again, and polymorphism was acquired with the advent of 'C++'. Other evolutionary lines are represented by Prolog, which epitomises the declarative family, and Lisp, which is the archetype of list processing languages (and which remains almost unchanged since its inception). Poplog, encompassing as it does representatives of all three families (Pop 11, based on Pop 2, Prolog and Lisp) is perhaps a unique synthesis. Finally one should not omit mention of that most modestly titled of all programming languages, APL: '*A* Programming Language'.

But this is not a history of programming languages: I cite the above examples only to place TEX within context, for although when teaching TEX to secretaries one does not necessarily stress the fact of its being a computer programming language *per se*, a computer programming language it most certainly is. Indeed, TEX is 'Turing complete', which is a computer scientist's jargon for saying that TEX could be used as a general purpose programming language since it has the necessary flexibility, although apart from the intellectual satisfaction there would be little point in so doing: TEX's *forte* is clearly computer typesetting, and only programmers or perverts could derive pleasure from coercing it into calculating cube roots or cosines!

So what is the common theme among all the languages cited above? Simply this: that almost every one of them has either given birth to a successor (which is not necessarily more successful: *cf.* Pascal → Modula → Oberon), or has simply fallen into disuse; Cobol and

[4] *Mus ordinatus microsoftiensis* or *Mus ordinatus applemacintoshii*

[5] *Xestobium rufovillosum*

Lisp alone, which occupy highly specialised niches, remain relatively unchanged, and of these only Cobol continues to play a significant rôle in mainstream computing (although Lisp remains the language of choice for many linguistic and related tasks).

It seems, then, that we have a choice: we can either allow natural selection to take its course, in which case TEX, having fulfilled its appointed rôle on this planet (which I assume is to teach us the merits of literate programming, whilst encouraging us to devote ever more time to the typesetting of beautiful papers, presumably at the expense of ever less time spent actually researching or writing them), will surely join XCHLF, JEAN & JOSS in the great bit-bin in the sky; or we can adopt a corporate responsibility for the future of TEX and intercede in the process of natural selection, taking steps to ensure that TEX evolves into a typesetting system which is so demonstrably superior to the miasma of mouse-based, menu-driven, manipulators of text and images which are currently snapping at its heels that no-one will be able to deny it its rightful place at the forefront of typesetting technology for the twenty-first century.

Let us consider the options which are available to us:

1. We can leave TEX exactly as it is: this is clearly a defensible position as it is exactly what Knuth himself intends to do; it would be extremely arrogant of us to suggest that we know better than Knuth in this respect.
2. We can enhance TEX by just enough that those who really understand its power, its limitations, and its inner workings agree that it no longer has demonstrable defects (i.e. there are some 'simple' typesetting tasks with which TEX$_\pi$ could not deal correctly, but with which an enhanced TEX could).
3. We can enhance TEX by incorporating the combined wish-lists of its major practitioners, thereby seeking to make TEX all things to all men (and all women), whilst retaining its present 'look and feel'.
4. We can enhance TEX as in option 3 above, whilst taking the opportunity to re-consider, and perhaps substantially change, its present look and feel.
5. We can take the opportunity to do what I believe Knuth himself might do, were he to consider today the problems of typesetting for the first time: look at the very best of today's typesetting systems (clearly including TEX among these), and then design a *new* typesetting system, far more than just a synthesis of all that is best today, which addresses the needs and potential not only of today's technology, but that of the foreseeable future as well. We would need to find some way to incorporate that spark of genius which characterizes Knuth's work!

No doubt each of us will have his or her own ideas on the desirability or otherwise of each of these options; it is not my intention in this paper to attempt to persuade you that any one of them is clearly preferable; but I would be shirking my responsibilities were I not to caution that, in my opinion, option 3 appears to represent the worst of all possible worlds, representing as it does a clear case of 'creeping featurism' at its worst while not possessing any redeeming qualities of originality.

Option 1 is, as I have suggested above, clearly defensible, in that it is Knuth's own preferred position; despite my fears that TEX will succumb to the pressures of natural selection if it is adopted, it may be that TEX represents both the pinnacle and the end of an evolutionary line, and that future typesetting systems will be based on an entirely different philosophy (e.g. mouse-based).

Option 2 represents the most conservative evolutionary position and has, I believe, much to commend it, certainly in the short term: it would retain the present look and feel of TEX; and compatibility with current TEX programs, whilst not intrinsically guaranteed, could be ensured by careful design; at the very worst, one could envisage a command-line qualifier which would disable the extensions, leaving a true TEX 3 underneath. Although option 2 is in opposition to Knuth's expressed wishes, he has made it plain that he has no objection to such enhancements *provided that* the resulting system is not called TEX. I propose that we term the results of adopting option 2 'Extended TEX', both to indicate its nature, and, more importantly, to comply with the spirit as well as the letter of Knuth's wishes.

Option 3 is considerably less conservative, but does at least retain the present look and feel of TEX; it is completely open-ended in terms of the extensions made to TEX, and offers the opportunity to make sweeping enhancements (I hesitate to use the word 'improvements' for the reasons outlined above). Compatibility with current TEX programs need not prove problematic, provided that the design were adequately thought out, and again the possibility of a '`/noextensions`' qualifier provides a fallback position. The timescale for such an implementation would not be small if a new swarm of bugs is to be prevented, and it is not clear how future obsolescence is to be avoided: after all, if 'The Ultimate TEX' (as I will term it) includes all the proposed enhancements of TEX's major practitioners, what enhancements remain to be implemented in the future?

Option 4 represents the first attempt at a true re-design of TEX, allowing as it does the option to re-think TEX's look and feel, whilst continuing to incorporate many of its underlying algorithms. One could envisage, for example, an implementation of TEX in which text and markup were kept entirely separate, with a system of pointers from markup to text (and *vice versa*?). One advantage of such a scheme is that it would eliminate, at a stroke, the troublesome nature of the <space> character which currently complicates TEX; the escape character could become redundant, and the problems of category codes possibly eliminated. Of course, this is just one of many such possibilities: once one abandons the look and feel of TEX, the whole world becomes one's typesetting oyster. One might term such a version of TEX 'Future TEX'.

Option 5 is without doubt the most radical: not only does it reject (at least, initially), TEX's look and feel, it challenges the entire received wisdom of TEX and asks instead the fundamental question: "How should computer typesetting be carried out?" In so doing, I believe it best represents Knuth's own thoughts prior to his creation of TEX 78, and, by extrapolation, the thoughts which he might have today, were he faced for the first time with the problems of persuading a phototypesetter to produce results worthy of the texts which it is required to set. I think it important to note that there is nothing in option 5 which automatically implies the rejection of the TEX philosophy and paradigms: it may well be that, after adequate introspection, we will decide that TEX does, in fact, continue to represent the state of the typesetting art, and that we can do no better than either to leave it exactly as it is, or perhaps to extend it to a greater or lesser extent whilst retaining its basic model of the typesetting universe of discourse; on the other hand, neither does it imply that we *will* reach these conclusions. I will call such a system 'A New Typesetting System' (to differentiate it from 'The New Typesetting System' which is the remit of NTS, *q.v.*).

The options outlined above are not necessarily mutually exclusive: we might decide, for example, to adopt option 2 as an interim measure, whilst seeking the resources necessary to allow the adoption of option 5 as the preferred long-term position (indeed, I have considerable sympathy with this approach myself). But no matter which of the options we adopt, we also need to develop a plan of campaign, both to decide which of the options is the most preferable (or perhaps to adopt an option which I have not considered) and then to coordinate the implementation of the selected option or options.

As many of you will be aware, a start has already been made to this end: at a meeting of DANTE (the German-speaking TEX Users' Group) earlier this year, Joachim Lamarsch announced the formation of a steering group, organised under the ægis of DANTE, to co-ordinate developments of TEX; this group, diplomatically called 'NTS' so as to avoid any suggestion that it is TEX itself whose future is being considered, is chaired by Rainer Schöpf; the members are listed in Appendix A. An e-mail discussion list has also been created (called NTS-L),[6] with an open membership;[7] all messages are automatically forwarded to members of the NTS team. At the time of writing this article, the group has not yet formally met: instead, we have been content to listen to the many positive suggestions which have been put via the medium of NTS-L. It is clear that there is no general consensus at the moment as to which of the five options outlined above is preferable; some argue for strict compatibility with existing TEX implementations, whilst others argue that we must grasp the nettle and take this opportunity to create a truly revolutionary typesetting system. Some, at least, are quite content to adopt the Knuthian position, and simply use TEX as it is: "TEX is perfect" was the subject of more than one submission to NTS-L. One of the more interesting facts to emerge from the discussion is the different ways in which TEX is perceived: some see it simply as a tool for mathematical typesetting; others want to be able to create the most complex graphics without ever leaving TEX's protective shell; many want to be able to typeset arbitrarily complex documents (not necessarily containing one line of mathematics), but are content to leave graphics, at least, without TEX's remit.

So far, this paper has been concerned primarily with generalities; but I propose now to look at some of the specific issues to which I have earlier merely alluded, and to offer some personal opinions on possible ways forward. I propose to start by attempting to answer the question which I believe lies at the very heart of our quest: "What is the essence of TEX?"

It seems to me that there are some aspects of TEX which are truly fundamental, and some which are merely peripheral: among the fundamental I include its descriptive and character-oriented nature, its portability, and its deterministic behaviour; I also include some elements which I have not so far discussed: its programmability (for example, the way in which loops can be implemented, even though they are not intrinsic to its design), its generality (the fact that it can be used to typeset text, mathematics, and even music), its device independence, and its sheer æsthetic excellence (the fact that, in reasonably skilled hands, it can produce results which are virtually indistinguishable from material set professionally using traditional techniques). Equally important, but from a different perspective, are the facts that it is totally documented in the ultimate exposition of literate programming (the *Computers & Typesetting* quintology), that it is virtually bug-free, that any bugs which do emerge from the woodwork are rapidly exterminated by its author, and finally that for higher-level problems (i.e. those which are at the programming/user-interface level rather than at the WEB level), there are literally thousands of skilled users to whom one can appeal for assistance. We should not forget, too, Knuth's altruism in making the entire source code[8] freely available with an absolute minimum of constraints. It is almost certainly true that this last fact, combined solely with

[6] `NTS-L@VM.URZ.Uni-Heidelberg.De`

[7] Send a message to `LISTSERV@VM.URZ.Uni-Heidelberg.De` with a single line body containing the text
`Subscribe Nts-L <given name> <SURNAME>`

[8] including source for the TEX and METAFONT books; this is frequently forgotten...

the sheer excellence of TEX, is responsible for TEX's widespread adoption over so much of the face of our planet today.

Among its more peripheral attributes I include its implementation as a macro, rather than as a procedural or declarative, language, and perhaps more contentiously, its fundamental paradigm of 'boxes and glue'. I hesitate to claim that boxes and glue are not fundamental to TEX, since in many senses they clearly are: yet it seems to me that if a descendant of TEX were to have detailed knowledge of the *shape* of every glyph (rather than its bounding box, as at present), and if it were perhaps to be capable of typesetting things on a grid, rather than floating in space and separated by differentially stretchable and shrinkable white space, but were to retain all of the other attributes asserted above to be truly fundamental, then most would recognise it as a true descendant of TEX, rather than some mutated chimera.

Without consciously thinking about it, I have, of course, characterized TEX by its strengths rather than its weaknesses.[9] But if we are to intervene in the processes of natural selection, then it is essential that we are as familiar with TEX's weaknesses as with its strengths: if it had no weaknesses, then our intervention would be unnecessary, and the whole question of the future of TEX would never have arisen. But whilst it is (relatively) easy to identify a subset of its characteristics which the majority of its practitioners (I hesitate to say 'all') would agree represent its fundamental strengths, identifying a similar subset of its characteristics which represent its fundamental weaknesses is far more contentious. None the less, identify such a subset we must.

Perhaps the safest starting point is to consider the tacit design criteria which Knuth must have had in mind when he first conceived of TEX, and which remain an integral part of its functionality today. TEX, remember, was born in 1978 —a time when computer memories were measured in kilobytes rather than megabytes, when laser printers were almost unknown, when the CPU power of even a University mainframe was probably less than that available on the desktops of each of its academics today, and when real-time preview was just a pipe dream.[10] Each and every one of these limitations must have played a part in TEX's design, even though Knuth may not have been consciously aware of the limitations at the time. (After all, we are only aware of the scarcity of laser printers in 1978 because of their ubiquity today; we aren't aware of the limiting effects of the scarcity of ion-beam hyperdrives because they haven't yet been invented…). But by careful reading of *The TEXbook* (and even more careful reading of `TEX.WEB`), we can start to become aware of some of the design constraints which were placed on Knuth (and hence on TEX) because of the limits of the then-current technology. For example, on page 110 one reads: "TEX uses a special method to find the optimum breakpoints for the lines in an entire paragraph, but it doesn't attempt to find the optimum breakpoints for the pages in an entire document. *The computer doesn't have enough high-speed memory capacity to remember the contents of several pages* [my stress], so TEX simply chooses each page break as best it can, by a process of 'local' rather than 'global' optimisation." I think we can reasonably deduce from this that if memory had been as cheap and as readily available in 1978 as it is today, TEX's page-breaking algorithm may have been very different. Other possible limitations may be inferred from the list of numeric constants which appear on page 336, where, for example, the limit of 16 families for maths fonts is stated (a source of considerable difficulties for the designers of the New Font Selection Scheme);[11] 16 category codes, too, although seemingly just enough, force the caret character (^) to serve triple duty, introducing not only 64-byte offset characters and hexadecimal character specifiers, but also serving as the superscript operator.

So, we may reasonably infer that the combined restrictions of limited high-speed memory, inadequate CPU power, and very limited preview and proof facilities, combined to place limitations on the original design of TEX; limitations the effect of which may still be felt today. It is perhaps unfortunate that in at least one of these areas, that of high-speed memory, there are still systems being sold today which have fundamental deficiencies in that area: I refer, of course, to the countless MS/DOS-based systems (without doubt the most popular computer system ever invented) which continue to carry within them the design constraints of the original 8088/8086 processors. Because of the ubiquity of such systems, there have been a fair number of submissions to the NTS list urging that any development of TEX bear the constraints of these systems in mind; despite the fact that I too am primarily an MS/DOS user, I have to say that I do not feel that the 64K-segment, 640K-overall limitations of MS/DOS should in any way influence the design of a new typesetting system. Whilst I feel little affinity for the GUI-based nature of Microsoft Windows, its elimination of the 640K-limit for native-mode programs is such a step forward that I am prepared to argue that any future typesetting system for MS/DOS-based systems should assume the existence of Windows (or OS/2), or otherwise avoid the 640K barrier by using techniques such as that adopted by Eberhard Mattes' *emTEX386*.[12] If we continue to observe the constraints imposed by primitive systems such as MS/DOS, what

[9] OK, I admit it: TEX *might* have weaknesses…

[10] Although on page 387 (page numbers all refer to *The TEXbook* unless otherwise stated), we find "Some implementations of TEX display the output as you are running".

[11] Frank Mittelbach and Rainer Schöpf

[12] *emTEX386* uses a so-called 'DOS extender'.

hope have we of creating a typesetting system for the future rather than for yesterday?

These might be termed the historical (or 'necessary') deficiencies of TEX: deficiencies over which Knuth essentially had no control. But in examining the deficiencies of TEX, we must also look to the needs of its users, and determine where TEX falls short of these, regardless of the reasons. The term 'users', in this context, is all-encompassing, applying equally to the totally naïve user of LATEX and to the format designers themselves (people such as Leslie Lamport, Michael Spivak, and Frank Mittelbach); for although it is possible for format designers to conceal certain deficiencies in TEX itself (e.g. the lack of a `\loop` primitive), the more fundamental deficiencies will affect both. (Although it is fair to say that a sure sign of the skill of a format designer is the ease with which he or she can conceal as many of the apparent deficiencies as possible). An excellent introduction to this subject is the article by Frank Mittelbach in *TUGboat*, 'E-TEX: Guidelines for future TEX' [2], and the subsequent article by Michael Vulis, 'Should TEX be extended?' [3]. Perhaps less accessible, and certainly more voluminous, are the combined submissions to NTS-L, which are archived at `TeX.Ac.Uk` as `Disk$TeX:[TeX-Archive.Nts]Nts-L.All` and at `Ftp.Th-Darmstadt.De` as `/pub/tex/documentation/nts-l/*`.

So, what are these so-called 'fundamental deficiencies'? No doubt each of us will have his or her own ideas, and the three references cited above will serve as an excellent starting point for those who have never considered the subject before. What follows is essentially a very personal view —one person's ideas of what he regards as being truly fundamental. It is not intended to be exhaustive, nor necessarily original: some of the ideas discussed will be found in the references given; but I hope and believe that it is truly representative of current thinking on the subject. Without more ado, let us proceed to actual instances.

1. *The lack of condition/exception handling:* It is not possible within TEX to trap errors; if an error occurs, it invariably results in a standard error message being issued, and if the severity exceeds that of 'warning'[13] (e.g. overfull or underfull boxes), user interaction is required. This makes it impossible for a format designer to ensure that all errors are handled by the format, and actually prevents the adoption of adequate defensive programming techniques. For example, it is not possible for the designer of a font-handling system to trap an attempt to load a font which does not exist on the target system.
2. *The inability to determine that an error has occurred:* The `\last...` family (`\lastbox`, `\lastkern`, `\lastpenalty`, `\lastskip`) are unable to differentiate between the absence of a matching entity on the current list and the presence of a zero-valued entity; since there is all the difference in the world between a penalty of zero and no penalty at all, vital information is lost.
3. *The hierarchical nature of line-breaking and page-breaking:* Once a paragraph has been broken into lines, it is virtually impossible to cause TEX to reconsider its decisions. Thus, when a paragraph spans two pages, the material at the top of the second page will have line breaks within it which are conditioned by the line breaks at the bottom of the previous page; this is indefensible, as the two occur in different visual contexts. Furthermore, it prevents top-of-page from being afforded special typographic treatment: for example, a figure may occur at the top of the second page, around which it is desired to flow text; if the paragraph has already been broken, no such flowing is possible (the issue of flowing text in general is discussed below). The asynchronous nature of page breaking also makes it almost impossible to make paragraph shape dependent on position: for example, a particular house style may require paragraphs which start at top of page to be unindented; this is non-trivial to achieve.
4. *The local nature of page breaking:* For anything which approximates to the format of a Western book, the verso-recto spread represents one obvious visual context. Thus one might wish to ensure, for example, that verso-recto pairs always have the same depth, even if that depth varies from spread to spread by a line or so. With TEX's present page breaking mechanism, allied to its treatment of insertions and marks, that requirement is quite difficult to achieve. Furthermore, by localising page breaking to the context of a single page, the risk of generating truly 'bad' pages is significantly increased, since there is no look-ahead in the algorithm which could allow the badness of subsequent pages to affect the page-breaking point on the current page.
5. *The analogue nature of 'glue':* TEX's fundamental paradigm, that of boxes and glue, provides an elegant, albeit simplistic, model of the printed page. Unfortunately, the flexible nature of glue, combined with the lack of any underlying grid specification, makes grid-oriented page layup impossible to achieve, at least in the general case. The present boxes and glue model could still be applicable in a grid-oriented version of TEX, but in addition there would need to be what might be termed 'baseline attractors': during the glue-setting phase, baselines would be drawn towards one of the two nearest attractors, which would still honour the constraints of `\lineskiplimit` (i.e. if the effect of drawing a baseline upwards were to bring two lines too close together, then the baseline would be drawn downwards instead).

[13] I use the VAX/VMS conventions of 'success', 'informational', 'warning', 'error' and 'severe error' as being reasonably intuitively meaningful here.

6. *The lack of any generalised ability to flow text:* TeX provides only very simple paragraph shaping tools at the moment, of which the most powerful is `\parshape`; but one could envisage a `\pageshape` primitive and even a `\spreadshape` primitive, which would allow the page or spread to be defined as a series of discrete areas into which text would be allowed to flow. There would need to be defined a mechanism (not necessarily within the primitives of the language, but certainly within a kernel format) which would allow floating objects to interact with these primitives, thereby providing much needed functionality which is already present in other (mouse-oriented) systems.
7. *An over-simplistic model of lines of text:* Once TeX has broken paragraphs into lines, it encapsulates each line in an `\hbox` the dimensions of which represent the overall bounding box for the line; when (as is usually the case) two such lines occur one above the other, the minimum separation between them is specified by `\lineskiplimit`. If any two such lines contain an anomalously deep character on the first line, and/or an anomalously tall character on the second, then the probability is quite great that those two lines will be forced apart, to honour the constraints of `\lineskiplimit`; however, the probability of the anomalously deep character coinciding with an ascender in the line below, or of the anomalously tall character coinciding with a descender in the line above, is typically rather small: if TeX were to adopt a 'skyline'[14] model of each line, rather than the simplistic bounding-box model as at present, then such line pairs would not be forced apart unless it was absolutely necessary for legibility that they so be. Note that this does not require TeX to have any knowledge of the characters' *shape*; the present bounding-box model for characters is still satisfactory, at least for the purposes of the present discussion.
8. *Only partial orthogonality in the treatment of distinct entities:* TeX provides a reasonably orthogonal treatment for many of its entities (for example, the `\new`... family of generators), but fails to extend this to cover all entities. Thus there is no mechanism for generating new instances of `\marks`, for example. Similarly, whilst `\the` can be used to determine the current value of many entities, `\the \parshape` returns only the number of ordered pairs, and not their values (there is no way, so far as can be ascertained, of determining the current value of `\parshape`). It is possible to `\vsplit` a `\vbox` (or `\vtop`), but not to `*\hsplit` an `\hbox`. The decomposition of arbitrary lists is impossible, as only a subset of the necessary `\last`... or `\un`... operators is provided. The operatorless implicit multiplication of <number><dimen-or-skip register> (yielding <dimen>) is also a source of much confusion; it might be beneficial if the concept were generalised to <number><register> (yielding <register-type>). However, this raises many related questions concerning the arithmetic capabilities of TeX which are probably superficial to our present discussion. I would summarise the main point by suggesting that orthogonality could be much improved.
9. *Inadequate parameterisation:* TeX provides a very comprehensive set of parameters with which the typesetting process may be controlled, yet it still does not go far enough. For example, one has `\doublehyphendemerits` which provide a numeric measure of the undesirability of consecutive hyphens; it might reasonably be posited that if two consecutive hyphens are bad, three are worse, yet TeX provides no way of indicating the increased undesirability of three or more consecutive hyphens. Also concerned with hyphenation is `\brokenpenalty`, which places a numeric value on the undesirability of breaking a page at a hyphen; again it might be posited that the undesirability of such a break is increased on a recto page (or reduced on a verso page), yet only one penalty is provided. A simple, but potentially infinite, solution would be to increase the number of parameters; a more flexible solution might be to incorporate the concept of formula-valued parameters, where, for example, one might write something analogous to `\brokenpenalty = {\ifrecto` |500| `\else` |200| `\fi}`, with the implication of delayed evaluation.
10. *Inadequate awareness of æsthetics:* TeX is capable of producing results which æsthetically are the equal or better of any computer typesetting system available today, yet the results may still be poorer than that achieved by more traditional means. The reason for this lies in the increased detachment of the human 'operator', who now merely conveys information to the computer and sits back to await the results. When typesetting was accomplished by a human compositor, he or she was aware not only of the overall shape of the text which was being created, but of every subtle nuance which was perceivable by looking at the shapes and patterns created on the page. Thus, for example, rivers (more or less obvious patterns of white space within areas of text, where no such patterns are intended), repetition (the same word or phrase appearing in visually adjacent locations, typically on the immediately preceding or following line), and other æsthetic considerations leapt out at the traditional typesetter, whereas TeX is blissfully unaware of their very existence. Fairly complex pattern matching and even image processing enhancements might need to be added

[14] This most apposite and descriptive term was coined by Michael Barr.

to TEX before it was truly capable of setting work to the standards established by hot-metal compositors.

Clearly one could continue adding to this list almost indefinitely; every system, no matter how complex, is always capable of enhancement, and TEX is no exception to this rule. I have quite deliberately omitted any reference to areas such as rotated text and boxes, support for colour, or support for graphics, as I believe them to be inappropriate to the current discussion: they are truly *extensions* to TEX, rather than deficiencies which might beneficially be eliminated. But I believe I have established that there *are* areas in which TEX is capable of being improved, and would prefer to leave it at that.

This brings us therefore to the final theme: how should we proceed? The NTS-L approach is obviously helpful, in that it allows the entire (e-mail connected) TEX community to contribute to the discussion, but I see at least two problems:

1. Those who are not on e-mail[15] are essentially excluded from the discussion; I do not see any easy solution to this problem.
2. The views expressed are, in some cases, radically different, and I wonder whether we will ever converge on a universally acceptable decision.

The second is in many ways the more important issue (Knuth apart), for unless the decisions made are acceptable to a very large majority of the contributors, the group may split, with part electing to go one route and another part electing to adopt a different strategy. This could result in a proliferation of *Über-TEX*s, with a concomitant fragmentation of the user community. Natural selection would surely winnow out the real non-starters before too long, but I seriously worry about the effect of such a proliferation on the TEX community, and even on TEX itself: after all, if we can't agree amongst ourselves whether there should be a successor to TEX, and if so what functionality it should possess, the whole credibility of the TEX ethos will be called into question. I would not like this to happen.

Somehow, therefore, we have to find a generally acceptable solution. My intuitive feeling is that such a solution will either be conservative or radical, but nothing in between. (This may seem like a distinct hedging of bets, but I hope that my meaning is clear: I believe that a compromise solution, which tries to be all things to all people, is doomed to failure). I do truly believe that adopting both solutions (one conservative, one radical) may be the best way forward: as an initial step, we identify (as I have tried to do above) any true *deficiencies* of TEX —those that actually prevent it from accomplishing its stated aims —and rectify those, producing a system that is backwards compatible with present TEX implementations whilst being capable of achieving superior results. In parallel with this (which is intended to be a reasonably short term and straightforward project, requiring not too much in the way of resources), we start planning a truly radical New Typesetting System, with the same fundamental design desiderata as TEX (portability, freely available, fully documented, bug-free...), but designed for the technology of tomorrow[16] rather than that of today.

Considering first the conservative approach, we will need to identify what is feasible, as well as what is desirable. Clearly this will require advice from those who are truly familiar with `TEX.WEB`, as I see this approach purely as modifications to the WEB rather than as a re-write in any sense. Chris Thompson and Frank Mittelbach are obvious candidates here, and Frank is already a member of the NTS team; I would suggest that if we adopt this strategy, Chris be invited to participate as well. Once we have identified what is possible, we will need a reasonably accurate estimate of time-to-implement, and if this exceeds that which can be achieved with volunteer labour, we will need to seek funds to implement this solution. I would suggest that TUG be approached at this stage (obviously they will have been kept informed of the discussions), and asked if they are willing to fund the project. There seems no point in projecting beyond this stage in the present paper.

For the radical approach, familiarity with WEB is probably unnecessary, and indeed may be a disadvantage: if we are seeking a truly NEW Typesetting System, then detailed familiarity with current systems may tend to obfuscate the issue, and certainly may tend to constrain what should otherwise be free-ranging thoughts and ideas. We will need to consult with those outside the TEX world, and the advice of practising typographers[17] and (probably retired) compositors will almost certainly prove invaluable. But above all we will need people with vision, people who are unconstrained by the present limits of technology, and who are capable of letting their imagination and creativity run riot.

And what conclusions might such a group reach? Almost by definition, the prescience required to answer such rhetorical questions is denied to mere mortals; but I have my own vision of a typesetting system of the future, which I offer purely as an example of what a New Typesetting System might be. Firstly (and despite my quite ridiculous prejudices against windowing systems), I believe it will inherently require a multi-windowing environment, or will provide such an environment itself (that is, I require that it will make no assumptions about the underlying operating environment, but will instead make well-defined calls through a generic interface; if the host system supports a multi-windowing environment such as Microsoft Windows

[15] Knuth is not on e-mail...

[16] and beyond...

[17] Michael Twyman and Paul Stiff have indicated a keen desire to be involved in the project.

or the X Window System, the NTS will exploit this; if the host system does not provide such intrinsic support, then it will be the responsibility of the implementor to provide the multi-windowing facilities). I envisage that perhaps as many as eight concurrent displays might be required: linked graphic and textual I/O displays, through which the designer will be able to communicate the underlying graphic design in the medium of his or her choice (and observe in the other window the alternative representation of the design); an algorithmic (textual) display, through which the programmer will communicate how decisions are to be made; two source displays, one text, one graphic, through which the author will communicate the material to be typeset; and a preview display, through which an exact facsimile of the finished product may be observed at any desired level of detail. A further display will provide interaction (for example, the system might inform the user that some guidance is needed to place a particularly tricky figure), and the last will enable the user to watch the system making decisions, without cluttering up the main interactive window. Needless to say, I assume that the system will essentially operate in real time, such that changes to any of the input windows will result in an immediate change in the corresponding output windows. I assume, too, that the input windows will be able to slave other unrelated programs, so that the user will be able to use the text and graphics editors of his or her choice. Of course, not all windows will necessarily be required by all users: those using predefined designs will not need either the design-I/O or the algorithm-input windows, and will be unlikely to need the trace-output window; but the interaction window may still be needed, and of course the source-input windows unless the source, too, has been acquired from elsewhere. For just such reasons, the system will be capable of exporting any designs or documents created on it in plain text format for import by other systems.

And underneath all this? Perhaps no more than a highly refined version of the TEX processor; totally re-written, probably as a procedural language rather than a macro language (why procedural rather than, say, list processing or declarative? to ensure the maximum acceptability of the system: there are *still* more people in the world who feel comfortable with procedural languages than with any of the other major genres), and obviously embodying at least the same set of enhancements as the interim conservative design, together with support for colour, rotation, etc. The whole system will, of course, be a further brilliant exposition of literate programming; will be placed in the public domain; will be capable of generating DVI files as well as enhanced-DVI and POSTSCRIPT; and will be so free of bugs that its creators will be able to offer a reward, increasing in geometric progression, for each new bug found. . .

But we will need one final element, and I have deliberately left this point to the very end: we will need the advice of Don Knuth himself. Don has now distanced himself from the TEX project, and is concentrating on *The Art of Computer Programming* once again. This detachment is very understandable —TEX has, after all, taken an enormous chunk out of his working (and, I suspect, private) life —and I hope that we all respect his wish to be allowed to return once again to 'mainstream' computer science, mathematics, and Bible study. But I think it inconceivable that we can afford to ignore his advice; and if I were to have one wish, it would be this: that I would be permitted to meet him, for whatever time he felt he could spare, and discuss with him the entire NTS project. I would like to know, above all, what changes *he* would make to TEX, were he to be designing it today, rather than fifteen years ago; I would like to know if he agrees that the deficiencies listed above (and those that appear elsewhere) are genuine deficiencies in TEX, or are (as I sometimes fear) simply the result of an inadequate understanding of the true power and capabilities of TEX; and I would like to know how he feels about the idea of an 'Extended TEX' and of a New Typesetting System (I suspect he would be far more enthusiastic about the latter than the former). And I suppose, if I am honest, I would just like to say 'Thank you, Don', for the countless hours, days, weeks, months and probably years of pleasure which TEX has given me.

References

[1] Donald E. KNUTH: "The Future of TEX and METAFONT", in *TUGboat*, Vol. 11, No. 4, p. 489, November 1990.

[2] Frank MITTELBACH: "E-TEX: Guidelines for future TEX", in *TUGboat*, Vol. 11, No. 3, pp. 337–345, September 1990.

[3] Michael VULIS: "Should TEX be extended?", in *TUGboat*, Vol. 12, No. 3, pp. 442–447, September 1991.

[4] Zlatuška, Jiří(ed): *EuroTEX '92 Proceedings*, pp. 235–254, September 1992. Published by CSTUG, Czechoslovak TEX Users Group, ISBN 80-210-0480-0.

Appendix

Inaugural members of the NTS-L team

- Rainer Schöpf (Chairman)
- Peter Abbott
- Peter Breitenlohner
- Frank Mittelbach
- Joachim Schrod
- Norbert Schwarz
- Philip Taylor

E-TEX: Guidelines for Future TEX Extensions*

Frank Mittelbach

Electronic Data Systems (Deutschland) GmbH
Eisenstraße 56, D-6090 Rüsselsheim,
Federal Republic of Germany
mittelbach@mzdmza.zdv.uni-mainz.de

Abstract

With the announcement of TEX 3.0, Don Knuth acknowledged the need of the (ever growing) TEX community for an even better system. But at the same time, he made it clear, that he will not get involved in any further enhancements that would change the TEXbook.

TEX started out originally as a system designed to typeset its author's own publications. In the meantime it serves hundreds of thousands of users. Now it is time, after ten years' experience, to step back and consider whether or not TEX 3.0 is an adequate answer to the typesetting requirements of the nineties.

Output produced by TEX has higher standards than output generated automatically by most other typesetting systems. Therefore, in this paper we will focus on the quality standards set by typographers for hand-typeset documents and ask to what extent they are achieved by TEX. Limitations of TEX's algorithms are analyzed; and missing features as well as new concepts are outlined.

*Published in TUGboat, Volume 11 (1990), No 3 — 1990 Conference Proceedings.