Documents, Compuscripts, Programs, and Macros

Jonathan Fine
203 Coldhams Lane, Cambridge, CB1 3HY England
J.Fine@pmms.cam.ac.uk

Abstract

Certain aspects of the history and nature of the TEX typesetting program are described. This leads to a discussion of strategies for possible future developments. For clarity, the key terms document, compuscript, program and macro are defined.

The main argument is that improved macro packages and .dvi file processors will solve many problems, and that a rigorous syntax for input compuscripts should be developed and used. Such a strategy will allow a different and superior typesetting engine, should such arise, to be used in the place of TEX. It will also allow the same compuscript to be used for other, non-typesetting, purposes.

The Beginning

Much has changed since the creation of TEX by Donald Knuth in the years around 1980. Many millions now use computers for document preparation and production, and these computers are many times more powerful than those so used in 1980. Laser printers are now cheap and commonplace. PostScript has become a widely available standard for driving phototypesetters. The occupation of specialists has become a widespread daily activity. Much indeed has changed.

TEX is one typesetting system among dozens if not hundreds, counting not only DTP packages but also the various word processors available. Here are some of TEX's particular characteristics

- extremely reliable and bug-free
- available on almost all machines
- available at no or low cost
- constant unchanging behaviour
- portable ASCII input
- high quality output
- mathematical setting capabilities
- programmability via macros

which leave it without rival for use by the scientific scholarly community, and elsewhere.

TEX has limitations. If it did not, it could not be. Hegel wrote, 'that one who will do something great must learn to limit oneself'. It was wise of Knuth, not to create a text editor for use with TEX. Nor did he create general indexing or cross-referencing tools. Nor a spell-checker. All but the most basic functions are omitted, to be supplied by macros and parameter values. This gives a great flexibility, and reduces the decisions and labor involved in writing the program. Knuth supplies a basic collection of 'plain' macros. But even that most basic part of computer typesetting, persuading an output device to emit a typeset page, this vital part of the system lies outside the limited system for which Knuth himself took responsibility.

Indeed, this abdication of responsibility is a master stroke. The output devices are numerous, diverse, and more are yet to come. Therefore, typesetting is brought to a stop with the production of the .dvi file, which is a rigorously specified description of the location of every character and rule on the page. Each implementation is then responsible for transforming this .dvi file to meet the requirements of the various output devices. Because there is a rigorous standard for .dvi files, this separation of duties is a pleasant cooperation. Moreover, the same .dvi standard and processors can now be used by other typesetting systems, new and yet to be.

Knuth did not write editor, indexer, or output device driver. Nor did he write more than a few thousand lines of macros. He did write TEX the program (and METAFONT, and the Computer Modern fonts). To support this activity he also wrote the WEB system for documentation or literate programming. The skillful use of this tool has contributed greatly, I believe, to the high quality and thus durability of TEX. This lesson needs must be well learnt and comprehended by those who seek to provide an improved replacement.

I think it very important to understand just what it is we have with TEX. Richard Palais (1992)

gives another, balanced, discussion of the nature of TEX, with which I am in broad agreement. Frank Mittelbach (1990) carefully investigates and describes some of the typesetting limitations of TEX. Philip Taylor (1992) exaggerates the deficiencies of TEX. In particular, of his list (pages 438 - 440) of 10 claimed limitations, at least 5 (namely 1, 3, 4, 5 and 6) are quite possible with TEX as it is today. The same applies (see Jonathan Fine (to appear)) to his goal (page 441) of a multiwindowed interactive display. There is a difference, it is important to note, between interacting with a visual or graphic representation of a document (so far as I know Scientific Word is the only TEX-compatible system that allows this) and having immediate preview of the result of changes to the underlying ASCII representation (as provided by Textures for smaller documents). Philip Taylor (1992a) seems to have no relevance to our discussion.

Stability

It is 5 years since Knuth (1989) released version 3 of TEX, and 4 years since his announcement (Knuth 1990)

> My work on developing TEX, METAFONT, and Computer Modern has come to an end. I will make no further changes except to correct extremely serious bugs.

which triggered a continuing debate on how, or whether, a successor to TEX should be provided. But much and more can be done with TEX as it is. Knuth wrote (*loc. cit.*)

> Of course I do not claim to have found the best solution to every problem. I simply claim that it is a great advantage to have a fixed point as a building block. Improved macro packages can be added on the input side; improved device drivers can be added on the output side.

and it is to these possibilities that we will now turn.

The purpose of a macro package is to transform an input document, written according to some rigorous or informal syntax, into a sequence of primitive typesetting commands, and thus, via the fundamental operations of line breaking, hyphenation, ligatures, boxes and glue, table formation and so forth have TEX the program produce typeset pages in the form of a `.dvi` file, and perhaps also some auxiliary text files. However, TEX does not contain a word-processor or text editor, and so offers little or no help in the composition of the input document.

Many benefits result from having a rigorously defined syntax for input documents, and so many problems disappear. Such rigor allows the same document to be processed in different ways for different purposes, such as editing, typesetting, spell-checking, on-line documentation, hypertext, or, if a program source file, compilation. Although this is not a new idea (see Charles Goldfarb (1990), pages 7-8)

> Markup should describe a document's structure and other attributes rather than specify processing to be performed on it, as descriptive markup need be done only once and will suffice for all future processing.
>
> Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and data bases can be used for processing documents as well.

none of the existing TEX macro packages is able to so typeset such a rigorously marked-up document. Moreover, the usual response to an error in markup is to have TEX the program generate an error message or worse, not generate an error. This behaviour is not a failing of TEX the program. Rather, it is a opportunity for improvement on the input side. The author has such work in progress.

It is worth noting that Knuth's `WEB` system made such a dual use (typesetting and compilation) of a single input file. This he did by writing two preprocessing programs (`WEAVE` and `TANGLE`) that convert a `WEB` input file into TEX and Pascal input files. For future reference note that although TEX source files are portable to any machine which has TEX installed, `WEB` files require the additional programs `WEAVE` and `TANGLE` to be also present.

On the output side, much can be done with `.dvi` files, provided suitable programs are available. By means of `\specials`, the device driver can be instructed to insert change bars, rotate tables, greyscale or color fonts, and so forth. All this is possible now, with TEX as it is, provided suitable programs are available.

It should be well understood that support for color, rotated tables, and other such goodies is not a matter of changing or 'improving' TEX the program. Rather, it requires matching facilities in the macro package used and in the `.dvi` file processor. TEX the program has no more involvement with the printing process that the moveable type typesetter of old, whose labor is blind to the color of the ink, or texture of the paper, used for the printing. Of course, the typographer or designer cares, or should, about these things.

There are other possibilities. Words to be indexed can be tagged using \specials (or even the whole word placed within the \special) and then extracted from the .dvi file. There are several advantages to this method. Firstly, it avoids the problems due to the asynchronous nature of the output routine, and also due to the expansion of macros during the \write command. Secondly, it allows the indexing software to extract additional information from the .dvi file, such as the location on the page (either by line or by physical location). Thirdly, this last data may be useful for hypertext applications. One can even cut-and-paste among .dvi files (see Asher 1992, von Bechtolsheim 1989, and Spivak et al. 1989). All this is possible so long as the TEX macros are properly set up, and so long as the .dvi file processing programs are available.

It is worth noting here that the work of the DVI driver standards committee (Reid and Hosek 1989, and Schrod 1991) seems to support my contention, that much remains to be done, to get the best out of what is already available to us. Lavagnino (1991), and Vesilo and Dunn (1993) discuss examples of how some applications require that much more than printed pages be produced. These problems can be solved by means of a suitable combination of macros and .dvi file processing programs.

Growth

This then is the background against which our use of TEX develops, and into which any successor will be introduced. TEX can still reach the highest typographical standards. But it seems that it is precisely in those areas, such as input file preparation and post-processing of the output file, which lay outside the limits that enabled Knuth's achievement, that the TEX system is deficient.

In particular, the lack of a front end for document preparation, that exploits the computing and graphical display capabilities that so many users now have available (and so few when TEX was first written) is a major obstacle to more widespread acceptance.

Elsewhere (Fine, to appear) I have indicated how TEX as it is today (and will be, major bugs aside, for the rest of time) can be used as the typesetting engine for such a visual document preparation system. However, any such will require programs that are specific to the architecture and capabilities of the host machine.

Much more can be done with TEX than is commonly realised. It is a powerful typesetting engine that can be turned to many purposes. Except for particular typographic functions (see Mittelbach 1990), such as detection and hence control of rivers of white space in paragraphs, most or all of its perceived limitations can be overcome by a judicious combination of improved macros and auxiliary programs. I have much work in progress (and less completed than I would like to admit) on improving macros.

The difficulty with auxiliary programs is that they are not automatically portable in the same manner as TEX the program is, and that they tend to become numerous and subject to change, much like macro packages.

A singular virtue of TEX, as vital to its success as the ground upon which we walk, and as commonly appreciated, is that it provides a programming environment, available and identical in operation on all machines. This is the TEX macro language. It is the basis for the portability of TEX documents. Moreover, transfer of such programs is no more than transfer of ASCII files.

Imagine now that we have a similar foundation for the writing of .dvi file processors. All manner of problems would go away, or at least be mitigated. There are about 10 standards for using \specials to access PostScript. The lack of a macro language gives an unwanted rigidity to the .dvi file processors, and so each standard is (or is not) hard-coded into each particular .dvi program.

Many indexing and hypertext problems can be resolved by post-processing the .dvi file, but not in a portable manner unless the .dvi processing program is similarly portable. Elsewhere (Fine, to appear) I have indicated how a visual front end to TEX can be assembled out of a suitable combination of a previewer (which is itself a .dvi file processor), a .dvi file editor, and TEX as it is but running a suitable and rather special macro package.

For such to be flexible, its outer form must be controlled by macros or the like. For such to be portable, the supporting programs must be both portable and ported.

Definitions

In order that my conclusions be stated as precisely as is possible, I will make some definitions.

By a *document* I will mean a physical graphical and perhaps substantial object containing text in various fonts, and perhaps other items such as symbols and photographs. Examples of a document are a book, a magazine or journal, a preprint, and a restaurant menu. These are substantial items, in the sense of their being made out of stuff. The

quality of the ink and paper, and the impression of the one on the other, are subtle aesthetic qualities of the document, in no sense determined by the typesetting process.

However, I will also regard an image on the screen of a computer to be a document, although of the insubstantial or un-stuffy kind. Such documents allow a different range of interactions with the reader, usually called the user, than the printed page. Indeed, in external form many computer programs are documents in this broad sense.

By a *compuscript*, or script for short, I mean a finite sequence of symbolic or numerically coded characters, such as ASCII, satisfying a formal or informal syntax. It may also contain references to external entities, which may be other documents, or to non-document elements such as photographs or illustrations. It is sometimes convenient to break a script down into complements, which are either mark-up or text. The syntax is then a system of rules which relate the mark-up to the text. Examples of compuscripts are TEX and LATEX document source files (these have an informal syntax), and SGML and program source files (which have a rigorous syntax).

By a *program* I mean an executable binary file. Program files cannot be read as a comprehensible sequence of characters. They contain machine instructions that are specific to the host machine on which the program is to be run. Properly written, programs will run as quickly as any software can to perform their given function, but to change a program is usually a slow and sometimes laborious process. Knuth wrote TEX the program and METAFONT the program. More exactly, he wrote documents which were then transformed via a compiler and other tools (literate programming) into versions of TEX the program, one for each machine architecture. He also wrote the 'plain' macros for TEX, and the Computer Modern source files for METAFONT.

We can now say what *macros* are. A collection of macros is a compuscript which controls or influences the operation of a program. This definition includes both the configuration or option files that many programs use to store system data and user preferences, but also the macro files used by TEX and METAFONT, or any other code written to be executed by an interpreting program. The distinction between a program and macros is not always clear-cut. For example, many microprocessors contain microcode which is called upon to perform various functions. Emulation is often achieved by expanding machine code for one processor into sequences of machine instructions for another. If not present, it is common to emulate machine instructions to a mathematics coprocessor.

The US photographer Ansell Adams compared the negative to the score for a piece of music, and the print to the performance. Adams is famed for his marvellous atmospheric photographs of Yosemite National Park. Developing his photographic analogy (is it a rule that every article should have one bad pun?), the compuscript is the negative for the production of a document, the program the fixed darkroom equipment, while the macros are the consumeable papers and chemicals and also the skill, habits, standards and creativity of the darkroom operator. Incidentally, many negatives require special human activity related to their content such as 'dodging' and 'burning' (this means giving more or less exposure to different parts of the negative) in order that they come out at their best.

Note added in Proof

There are several articles also in these proceedings that bear upon the topics discussed here. Rokicki expresses the idea of a programmable `.dvi` file processor, although as an implementor his focus is more on what is immediately possible or practical. I should have realised for myself the important 'color' motive, whose difficulties in the production setting are well expressed by Sofka. Laugier and Haralambous describe Philippe Spozio's interactive and visual `.dvi` file editor, and also Franck Spozio's TEX to SGML translation tools. These programs go some way to resolving, for documents marked up in the traditional plain TEX or LATEX manner, various real world problems, which are among the motives for the point of view I adopt in my article.

The deficiencies of TEX are once again exaggerated by Taylor. It is possible, for example, to typeset material on a grid, to flow text around insertions, to treat the two-page spread or even the chapter as the region over which page make-up and optimisation are performed, all this is possible with today's TEX, by writing admittedly tricky macros. The goal of Schrod is to provide a formal model of TEX the program (particularly its macro facilities) with which a user can interact, whereas my goal is to have formal syntax for compuscripts that can be understood by TEX (given suitable macros) and by the user alike.

Finally, the papers of Baxter, Ogawa, and Downes discuss progress and problems in the typesetting of structured documents—again, using traditional TEX macro tools. It is my contention that the macro development and performance difficulties that they face can be greatly eased by

the introduction of powerful development tools, amongst which will be sophisticated macros that will combine compuscript parsing macros with style sheet values to give rise to the document production macros.

Conclusions

It should now be clear that Knuth is responsible for only one part of the TEX typesetting system, although that part is its mighty heart or engine. It is my opinion that, good though they are, there is considerable room for improvement in those parts of the TEX system that Knuth did not provide, viz. macros and `.dvi` file processors.

Perhaps in the next 20 years, someone will write a worthy successor to TEX. This would be, like TEX itself, a great achievement. To supplant TEX, it will need to be substantially better. I would expect such a system to continue to use more-or-less if not exactly the same `.dvi` file format as TEX. It would be nice if both TEX and its successor shared at least one syntax for the compuscripts that are to be processed into documents. This will surely require that both operate to a syntax that is as rigorous as that for the `.dvi` files. Work on defining such a syntax and creating suitable TEX macros to process such documents can begin today, without knowing what the future may bring, but all the same helping to bring it about.

To hope for compatibility at the level of macros or format files is probably too much, and likely to be self-defeating. Fortunately, many though formats are, they are, or at least should be, few in relation to documents.

TEX as it is today can be used as the engine of an interactive and visual typesetting system. I encourage all those who want to write programs to join with me in turning this possibility into a reality. A valuable first step, with independent benefits and merits of its own, would be to write a 'universal' `.dvi` file processor that is controlled by macros, just as TEX is a universal typesetting engine.

If all is done properly, and to rigorous standards for both input and output, then it will be a simple matter to replace TEX the program by the new and much improved engine, when and if it arrives. Indeed, part of the whole strategy is to provide a clear rôle and interface for the typesetting engine.

Donald Knuth has not written much on successors to TEX. It is thus our responsibility to read carefully what he has written. I close by repeating his advice quoted earlier

> Of course I do not claim to have found the best solution to every problem. I simply claim that it is a great advantage to have a fixed point as a building block. Improved macro packages can be added on the input side; improved device drivers can be added on the output side.

Bibliography

Asher, Graham. "Inside Type & Set", *TUGboat*, **13** (1), pages 13 - 22, 1992.

Bechtolsheim, Stephan von. "A `.dvi` file processing program", *TUGboat*, **10** (3), pages 329 - 322, 1989.

Clark, Malcolm. "NEXTEX: A personal view", *TUGboat*, **14** (4), pages 374 - 380, 1993.

Fine, Jonathan. "Editing `.dvi` files, or visual TEX", *TUGboat*, (to appear)

Goldfarb, Charles. *The SGML Handbook*, Oxford University Press, 1990

Knuth, Donald E. "The new versions of TEX and METAFONT", *TUGboat*, **10** (3), pages 325 - 328, 1989.

Knuth, Donald E. "The Errors of TEX", *TUGboat*, **10** (4), pages 529 - 531, 1989.

Knuth, Donald E. "The future of TEX and METAFONT", *TUGboat*, **11** (4), page 489, 1990.

Lavagnino, John. "Simultaneous electronic and paper publication", *TUGboat*, **12** (3), pages 401 - 405, 1991.

Mittelbach, Frank. "E-TEX: Guidelines for future TEX", *TUGboat*, **11** (3), pages 337 - 345, 1990.

Palais, Richard. "Moving a fixed point", *TUGboat*, **13** (4), pages 425 - 432, 1992.

Reid, Tom and Don Hosek. "Report from the DVI driver standards committee", *TUGboat*, **10** (2), pages 188 - 191, 1989.

Schrod, Joachim. "Report on the DVI Driver Standard", *TUGboat*, **12** (2), pages 232 - 233, 1991.

Spivak, Michael, Micheal Ballantyne, and Yoke Lee. "HI-TEX cutting & pasting", *TUGboat*, **10** (2), pages 164 - 165, 1989.

Taylor, Philip. "The future of TEX", *TUGboat*, **13** (4), pages 426 - 442, 1992.

Taylor, Philip. "NTS: the future of TEX?", *TUGboat*, **14** (3), pages 177 - 182, 1992.

Vesilo, R.A. and Dunn, A. "A multimedia document system based on TEX and DVI documents", *TUGboat*, **14** (1), pages 12 - 19, 1993.