Dreamboat

ε-TeX V2: a peek into the future

Philip Taylor

Abstract

ε-TeX V1 was released towards the end of 1996, and it was intended at the time that ε-TeX V2 should be released approximately one year later. For various reasons the release date has slipped a little, but V2 is in alpha-test and we confidently expect to release it in the near future: indeed, it may well have been released by the time that this article appears in print. In this article the new features of V2 are reviewed, and we conclude by peeking a little further into the future to see what will happen with $\mathcal{NTS}$.

Introduction

When the $\mathcal{NTS}$ project was first established in 1992, we hoped that work would commence fairly quickly. Sadly it became all too obvious that a project of that magnitude would require one or more full-time workers, and none of the (volunteer) team had that sort of time to spare. The team decided that, until funding could be found, they would work on the less-demanding but still worthwhile task of extending TeX in its current (Pascal-web) form. The first fruits of that work were revealed in late 1996, when ε-TeX V1 was released. This version added approximately 30 new primitives to TeX and a small adjunct macro library was also produced which both extended the plain TeX format to accommodate the new primitives and also added new functionality such as natural language handling and TeX module libraries.

Once ε-TeX V1 had been released, the group were free to concentrate on the next version. Originally intended to ship a year later than the first, this version has slipped a little as pressure of other commitments has forced members of the team to invest less time in the project than they would have wished. However, V2 is now in alpha test, and we confidently expect to be able to make a general release in the near future: release may well have taken place by the time this article appears in print. The majority of the remainder of this article describes the features that we are fairly confident will be present in that release.

Ideas which are almost certain to appear in ε-TeX V2. Although we do not wish to give an absolute commitment at this stage, particularly as many of the proposals are the process of being tested at the time of writing, we do believe that the ideas contained in the following section are very likely to appear in ε-TeX V2. A subsequent section discusses ideas which may appear in future releases.

Increasing TeX's registers. TeX has 256 of the most commonly used registers: counts, dimens, skips, boxes, toks, etc., and whilst these are enough for normal applications, advanced formatting systems really require more. In ε-TeX V2, we intend to provide 32768 of each of these, which we hope will be sufficient for the most demanding packages. Insertion classes will still be restricted to 256 or fewer, and `\box` 255 will retain its special significance. The "etex" format will allow both local and global allocation of these registers ("plain" allows only global), and for efficiency reasons a user will be able to elect whether to allocate a register from the dense (0..255) pool or from the sparse (256..32767). To allow the allocation mechanism to overflow from dense to sparse without risking a conflict with the allocation of insertion classes, the format allows a user or package to pre-reserve a number of insertion classes. Facilities for block-allocating a contiguous set of registers will be provided.

Improved natural language handling. TeX overloads the `\lccode` concept, using it both for "real" lower-casing operations and also for purposes of hyphenation. In ε-TeX V2 these operations are unbundled, and the codes used for hyphenation can be staticised as the patterns are read in (the current set of lccodes is used). Thereafter, whenever a particular language is used, the corresponding set of hyphenation codes is loaded.

Arithmetic expressions. Although TeX can perform simple arithmetic (addition, multiplication and division), these operations are in general "assignments" and therefore cannot be used in expansion-only and certain other contexts. ε-TeX V2 provides a set of arithmetic primitives which evaluate an expression in such a way that the value of the expression can be accessed in expansion-only contexts, as well as being usable (for example) when TeX is looking for a ⟨*number*⟩, ⟨*dimen*⟩, etc. As TeX intentionally uses only integer arithmetic wherever the results of a computation are accessible to the user, floating-point arithmetic has *not* been provided. There are four new primitives, `\numexpr`,

`\dimexpr`, `\glueexpr` and `\muexpr`, each of which requires its operands to be of appropriate type (or coercible to that type). Parentheses may be used to indicate precedence wherever this will clarify or disambiguate an expression. The normal arithmetic operators "+", "-", "*" and "/" are allowed within an expression.

Discards are no longer discarded!. When TEX performs page-breaking, so-called "discardable items" which follow the chosen breakpoint are discarded; whilst this is perfectly reasonable if the page break is actually taken, recursive techniques aimed at optimising the appearance of multiple pages require the ability to "undo" a pagebreak in order to try the effect of breaking elsewhere. The discarded items are therefore required in order to re-create the vertical list which TEX is trying to break. In ε-TEX V2, we allow access to these "discarded items" via a new primitive `\pagediscards`. The discards which occur during `\vsplit`ting are also accessible via an analogous primitive `\splitdiscards`. Both of these primitives return a vertical list, similar to that obtained by `\unvbox`ing a box register.

Read-write access to `\parshape`. Although TEX allows the user to create arbitrarily complicated paragraph shapes through the use of the `\parshape` primitive, it provides no way for the user to find out which `\parshape` is currently active (although it does allow the user to ascertain the number of lines of the current `\parshape` specification). In ε-TEX V2, we allow full read access to all the elements of the current `\parshape`.

Interrogating the current conditional context. In ε-TEX V1 we allowed users to make environmental enquiries concerning the current group, both as to its depth of nesting and to its type. In ε-TEX V2, we generalise this concept and allow analogous access to the current conditional context through the use of `\currentiflevel`, `\currentiftype`, `\currentifbranch` and `\showifs`.

Access to information concerning font-character combinations. Although it is possible to gain some information about a particular character in a given font by typesetting that character in a box and then measuring the dimensions of the box, not all the dimensions of the character can be reliably obtained in this way, and there is no way to ensure that the character actually exists in the font before attempting to typeset and measure it. In ε-TEX V2 we allow the user both to check whether a particular character exists in a given font, using `\iffontchar`, and (if it does exit) to measure the four fundamental dimensions of that font/character combination using `\fontcharwd`, `\fontcharht`, `\fontchardp`, and `\fontcharic` (representing width, height, depth and italic correction respectively). Furthermore, we ensure that users are alerted to the existence of missing characters in a font by causing lost characters to be logged to the console as well as to the log file if `\tracinglostchars` is set to a value greater than 1.

Better debugging aids. In order to assist in diagnosing mis-matched or runaway group problems, ε-TEX V2 allows the user to opt to be warned whenever a file is left in a group or conditional other than that at which it was entered. This may be accomplished by setting `\tracingnesting` to a value greater than zero.

Subtle change to the semantics of `\protected`. ε-TEX V1 introduced a new prefix, `\protected`, which inhibited the expansion of the "protected" macro in contexts in which expansion was unlikely to be required. Further research into this area suggested that at least one such case had been missed, and "protected" macros are now inhibited from expansion when TEX is scanning ahead while processing alignments.

Optimisations. To improve the overall efficiency of ε-TEX internal modifications have been made to reduce the resources required when there are a number of `\aftergroup`s active for a single group, and to eliminate the stack space wasted in setting a register to the same value as it currently holds.

Access to the components of a glue quantity. Whilst it is possible to gain access to the various components of a glue value by clever macro programming, the code required is sufficiently arcane to suggest that a better method is much to be preferred. Accordingly we are considering a set of primitives `\gluestretch`, `\gluestretchorder`, `\glueshrink` and `\glueshrinkorder` which will give much-simplified access to these quantities. As a part of the same process we are looking at two conversion primitives, `\mutoglue` and `\gluetomu`.

Improved typographic quality. Whilst the majority of the work in ε-TEX is aimed at providing the ε-TEX programmer with more powerful tools, we are aware that the real purpose of TEX is to generate typeset output of the highest quality. During a meeting in Brno with Prof. Knuth on the occasion of his honorary doctorate, he suggested that we might like to consider improving the typographic quality of the last line of a paragraph. According to Don, traditional (hot-lead) typesetters would set the last line to the same tightness or looseness as the immediately preceding line, and he thought that ε-TeX should be capable of doing likewise. We are looking into providing this but in a parameterised manner, so that *all* possibilities between TeX's current behaviour and that suggested by Don can be achieved. We think that this might be controlled by a parameter called `\lastlinefit`.

Improved typographic quality, cont.. In the same vein, we are looking into ways of allowing better parameterisation of the page-breaking process by having not just one penalty for (say) a club- or widow line but a whole array of such penalties which can reflect the undesirability of leaving one, two, three to *n* lines at the top or bottom of a page. Other related penalties are also candidates for this process.

Ideas still under discussion

The following ideas are all under discussion but are very unlikely to find their way into ε-TeX V2: some may be deferred to ε-TeX V3, and some may never appear at all. Although the group have some idea into which category each of these ideas may fall, it is probably not helpful to go into too much detail here, and so they are all lumped together as "under discussion".

Can TeX find this font?. In "the good old days", a TeX program could count on finding all 76 of the standard TeX fonts no matter where it was run in the world. These days, with many documents being set in exotic fonts from a myriad of sources, it is no longer certain that, just because site A has font F, site B will have the same font. We are therefore considering providing an `\iffont` primitive which will allow ε-TeX to ascertain at run-time whether a particular font exists on the system on which the document is being processed. It is not certain at this stage whether this would be a simple "does this font exist?" test, or a more complex "does it exist and is the TFM file for it valid?". `\tryfont` has also been suggested as an alternative approach.

Maths alignments. Peter Breitenlohner, the implementer of ε-TeX, probably typesets more mathematics than the rest of the group put together, and he believes that there is a case for a maths alignment primitive. He has not yet finished his research on this topic, and all that can be said at this stage is that we are considering implementing some form of `\malign`.

Typesetting on a grid. Whilst TeX is *excellent* at typesetting in designs where variable quantities of white space can be allowed to occur, trying to coerce it to set on a regular grid (something at which packages such as Quark Xpress excel) is *far* more difficult. The various macro-based solutions which have been tried do not seem to address the underlying problems, and we are looking at providing an entirely new paradigm within ε-TeX whereby material being typeset can be caused to "lock on" to a grid at some point in the page-building process. Although at first sight it might be thought that it is the reference points of the lines making up the page which need to lock to this grid, we are fairly certain that this is not always the case, and we are therefore looking at ways of associating one or more "handles" with a particular box. In the degenerate case there will be one handle which is coincident with the reference point of the line, but in more complex cases there may be two, three or ever more handles, each of which will lock on to one line of the raster. Even so, there are also situations in grid-based designs where the grid-lock contraints just have to be violated, and one topic still unresolved is whether it is then better simply to allow the box to float free, or whether it is better to constrain it in some way, perhaps by associating with the handle(s) a degree of flexibility which is in some way analogous to TeX's current use of the "glue" concept.

More on improved typographic quality. Another point made by Don during his stay in Brno is that there are situations in which TeX's (vertical) positioning of elements of mathematical formulæ is less than ideal. He points out that even in typesetting the TeXbook he had to make use of kludges such as `\sub \strut` in order to achieve the best visual effect. We are investigating ways in which the effect (and related effects) could be achieved by better parameterisation of the mathematical typesetting process.

NTS

In the introduction to this paper it was mentioned that the $\mathcal{NTS}$ project proper had been put "on ice" until the group had sufficient funds to allow a programmer to be employed full-time to work on the project. It is with great pleasure that I can now report that, as a result of the generosity of DANTEeV the group has DM 30 000 which can be used for this purpose. On the recommendation of Jiří Zlatuška, we have made an offer to Karel Skoupy of the Czech Republic, which he has accepted, and Karel will be starting work on 𝒩𝒯𝒮 during late February 1998. It has been agreed that the language of implementation will be Sun's JAVA, and Karel's first task (apart from becoming a JAVA expert...) will be to draw up a specification for 𝒩𝒯𝒮. Provided that the group agree with his design, he will then start work on implementing 𝒩𝒯𝒮, and we hope to be able to review his work after a further six months. Within one year of commencement we hope to have a working implementation of 𝒩𝒯𝒮 not simply a port of TEX to JAVA but a total re-design intended to emphasise the deep structure of TEX whilst avoiding the design features which make the present system rather difficult to extend or change.

The group are still determined that 𝒩𝒯𝒮 will be 100% TEX-compatible, and are confident that it will remain so for at least the first five years of its life. We are less certain whether divergence should then be permitted, in order to add new functionality which is in some way incompatible with TEX. If we do decide that compatibility must be sacrificed, we will give considerable notice of that decision, and users who must retain the ability to process legacy documents in a manner identical to TEX will be advised to take an archive copy of 𝒩𝒯𝒮 before compatibility is lost.

One exciting idea which the use of JAVA permits is the possibility to integrate access to CTAN (CNAN?!) with 𝒩𝒯𝒮: it is by no means impossible that 𝒩𝒯𝒮 might be able to fetch for itself any module which cannot be found on the local system and which is needed in order to process a document. If that becomes a reality, TEX will have become truly integrated into today's (and tomorrow's) globally-networked world.

◇ Philip Taylor
Royal Holloway and Bedford New College, University of London
Post: Room 106, Computer Centre,
Egham Hill, Egham
Surrey TW20 0EX, United Kingdom
P.Taylor@Vms.Rhbnc.Ac.Uk