Minutes of the $\mathcal{NTS}$ meeting held at Lindau on October 11/12th 1994

Philip Taylor,
Technical Director, $\mathcal{NTS}$ project

Present: Philip Taylor, Jiří Zlatuška, Bernd Raichle, Friedhelm Sowa, Peter Breitenlohner, Joachim Lammarsch.

It was **agreed** that no progress had apparently been made on the 'canonical TEX kit' project, and that no progress was likely to be made unless and until an active proponent of the project emerged within, or was recruited to, the group; accordingly the project was officially placed on ice.

It was **agreed** that in the absence of adequate funding for the $\mathcal{NTS}$ project proper, no serious work could be carried out; several possible sources of funding remained to be explored, and the group were hopeful that this project would see the light of day before too long.

It was **agreed** that the ε-TEX project was both feasible and very worthwhile, and that all efforts should initially be concentrated on achieving progress in this area. With the benefit of hindsight it was **agreed** that the original proposal to issue new releases at six-monthly intervals had been over-optimistic, and that a more realistic timescale would involve new releases once per year. It was also **agreed** that the first release should be accomplished as soon as possible, consistent with the need to ensure that the code released was both bug-free and unlikely to require more than a minimum of re-thinking in the light of experience.

The group attempted to identify as many ideas as possible which either have already been proposed for incorporation in ε-TEX, or which were natural consequences of (or alternatives to) ideas already proposed. The remainder of this document lists the various ideas mooted, and discusses their intention and implementation.

Proposals

`\horigin`, `\vorigin` (dimen registers, default = 1 in)

These two registers, requested by Phil, will serve to make explicit for the first time the canonical (1″, 1″) origin decreed by DEK in the definition of the DVI format, and on which all formats are currently predicated. Phil explained that his college, amongst others, had eschewed this convention right from the outset, and has instead adopted the more logical (0″, 0″) origin, requiring all drivers to be configured in a non-standard manner. Providing the origin registers within ε-TEX would allow all drivers to be reconfigured to the standard, whilst existing practices could be maintained simply by local initialisation of the registers to (0″, 0″). As ε-TEX might eventually require the adoption of a new version of the DVI format (to encompass, for example, colour), that might also be the appropriate time at which to propose universal adoption of a (0″, 0″) origin.

\⟨enhancement⟩`state` (internal integer registers, one for each enhancement)

A unified mechanism is proposed for all enhancements [1] whereby an internal integer register is associated with each, the name of the register being derived from the concatentation of the name of the enhancement and the word 'state'; such registers are read/write, and if their value is zero or negative the associated enhancement is disabled. [2] If a positive non-zero value is assigned to any such register, then the associated enhancement shall be enabled, and if the register is interrogated then a positive non-zero value shall indicate that the associated enhancement is enabled. It is possible that in a future release differing values assigned to or returned by such registers may indicate the revision-level of enhancements, and therefore it is initially recommended that only the values zero or one be used.

`\TeXXeTstate`, `\MLTeXstate` (internal integer registers)

These are the only two enhancements currently under consideration, although Bernd Raichle also has a proposal for an alternative ligaturing mechanism which would probably of necessity form an enhancement if adopted. MLTEX is not proposed for incorporation in the first release, but may be incorporated in the second. The group acknowledges the generosity of Michael Ferguson in allowing the incorporation of his work on MLTEX.

`\interactionmode` (internal integer register)

Allows read/write access to the present `\scrollmode`, `\nonstopmode`, etc., family of primitives; the values will be a monotonic sequence of period one, and descriptive names will be associated through the ε-plain (and ε-LATEX?) formats. [3]

⟨additional `\tracing...` detail⟩

Peter has implemented augmented semantics for some of the `\tracing` commands whereby increasingly positive values given increasingly detailed output.

`\protected` (new prefix for macro definitions)

Analogous to `\long`, `\outer`, etc., causes the associated macro to be non-expanding in contexts

where such behaviour is likely to be undesirable (in particular in `\write`s and `\edef`s); an explicit `\expandafter \empty` may be used to force expansion in these circumstances.

`\bind` (new prefix for macro definitions)

Proposed by Phil, this was intended to allow macros to be bound to the current *meaning* of embedded control sequences rather than to their names, in a manner analogous to PostScript's 'bind def'. However the group were unconvinced of the merits of this proposal, and it was classified as 'more work needed' (MWN).

`\evaluate {`⟨arithmetic expression⟩`}`

Intended for use on the r-h-s of `\count`, `\dimen` and `\skip` assignments, it would allow the use of infix arithmetic operators such as `+`, `-`, `*` and `/`; the type of the result would, in general, be the type of the simplest operand forming a part of the expression, and the normal semantics of TEX would allow this to be further coerced where necessary. Parenthesised sub-expressions would be allowed. [4]

`\contents` ⟨box #⟩

Proposed by Jiří, this is intended to allow the TEX programmer access to the sort of information normally available only via the log file as a result of a `\show`; in principle it would generate the simplest list of TEX tokens which would generate the box specified, assuming that each token generated still had its canonical meaning. MWN.

⟨anchors⟩

Proposed by Jiří, an "anchor point" would be in some senses analogous to a mark, but rather than recording textual information it would instead record the co-ordinates of itself, relative to the reference point of the smallest surrounding box. Additional new primitives would be required to return the co-ordinates of a specified anchor point. MWN.

`\scantokens {`⟨balanced text⟩`}`

Allows an existing token-list to be re-input under a different catcode regime from that under which it was created; as it uses all of TEX's present `\input` mechanism, even `%%ff` notation will be interpreted as if `\input`. Causes an 'empty filename' to be input, resulting in '`( )`' appearing in the log file if `\tracingscantokens` (q.v.) is strictly greater than zero. If the token list represents more than one line of input, and if an error occurs, then `\inputlinenumber` will reflect the logical input line from the token list rather than the current input line number from the current file.

`\unexpanded {`⟨balanced text⟩`}`

An alternative to `\protected`, for use when a whole brace-delimited token list ('balanced text') is to be protected from expansion. Intended to be used in `\write`s and `\edef`s.

`\every`⟨whatever⟩

The group discussed many possibilities of implementing additional `\every` primitives in ε-TEX; most were classified as MWN, but one (`\everyeof`) is being considered for ε-TEX version 1.

`\futuredef` ⟨cs⟩ ⟨tok⟩ ⟨tok⟩

Analogous to `\futurelet`, but the ⟨cs⟩ will be expandable, and expand to the next token encountered (or to the next balanced text if the next token is of catcode 1). MWN.

`\futurechardef` ⟨cs⟩ ⟨tok⟩ ⟨char-or-tok⟩

A combination of `\futurelet` and `\chardef`, will allow the next character to be inspected and its character code returned iff it has not yet been tokenised. If tokenisation has already taken place, will return −1. Intended to allow the catcode of the next character to be changed based on its value.

`\ifdefined` ⟨cs⟩

Allows direct testing of whether or not a given ⟨cs⟩ is defined.

`\ifcsname ... \endcsname`

Ditto, but for a sequence of ⟨tokens-expanding-to-characters⟩; this also avoids wasting of hash table space.

`\unless` ⟨boolean-if⟩

Inverts the sense of the following boolean-if; particularly useful in conjunction with `\ifeof` in `\loop ... \ifeof ... \repeat` constructs, but also of use with (say) `\ifdefined` and `\ifcsname`.

`\TeX`⟨whatever⟩`state`

More work needed! A mechanism whereby a TEX document can ask TEX some questions about the current state of its digestive tract. For example it would be nice to know if TEX was currently involved in an assignment, and if so which part of the assignment was currently being elaborated.

`\marks` ⟨integer⟩

Allows, for the first time, a whole family of marks rather than just the one provided by TEX; will also require analogous `\topmarks` ⟨integer⟩, etc. The group propose to provide 16 such marks, but are interested to know if the (LA)TEX community consider this sufficient. A related `\markdef` primitive may be provided to simplify mark allocation, in a manner analogous to the existing `\...def` primitives.

`\deferred \special` (or perhaps `\deferredspecial`)

At the moment, only `\writes` are deferred; there are cases when it would be desirable for other things, too, to be expanded only during `\shipout`, and `\special`s are one of these.

`\textcode` ⟨integer⟩

Could provide a text-mode analogy to TEX's `\mathcode`. MWN.

`\middle` ⟨delimiter⟩

Analogous to `\left` and `\right`, allows delimiters to be classed as `\middle`, and their spacing thereby adjusted.

`\filename`

Would allow access to the name of the file currently being `\input`. Lots of discussion on just how much or how little should be returned. MWN.

`\OSname`

Very contentious. Would provide the name of the operating system, and thereby allow documents to behave differently on different systems. Deprecated on that basis, and will not be provided unless/until a `\system` primitive is also provided.

`\system {`⟨balanced text⟩`}`

Definitely not proposed for ε-TEX version 1. Would allow operating system calls to be made, and their status and result(s) returned in some way. A lot MWN.

`\tracingscantokens` (internal integer register)

See `\scantokens`.

⟨smarter discretionaries⟩, e.g. `\discretionarylefthyphenmin`

Hyphenation after an implicit hyphen is sometimes highly desirable, and the group are investigating mechanisms whereby this could be both provided and parameterised. MWN.

`\everyhyphen` (token list register)

Would allow TEX's present hard-wired behaviour of placing an empty discretionary after every explicit hyphen to be modified. However, there are potentially problems of recursion, and perhaps even a need to remove the hyphen. MWN.

`\clubpenalties`, `\widowpenalties`

A start at improving TEX's penalty system by making it more flexible. These two penalty 'arrays' would allow a different penalty to be associated with one-line widows, two-line-widows, etc. [5]

`\ifenhanced`

A boolean-if which would return `true` iff any enhancement is enabled. Would allow a ε-TEX document to check if it is being processed in 'extended' more or 'enhanced' mode. Phil argues for this one but the group are unconvinced: the advice of the TEX community is to be sought.

`\lastnodetype`

Would allow, for the first time, the unambiguous identification of the type of the last node of one of TEX's internal lists, removing (for example) the ambiguity when `\lastpenalty` returns 0 (which can indicate no penalty node, or a penalty node with value 0). Would return one of a monotonic series of integers of period one. Meaningful names would be assigned to these through the ε-series formats. [3]

`\unnode`

Would allow the removal of *any* node from the end of one of TEX's internal lists.

`\lastnode`

Perhaps analogous to `\contents` (q.v.), or perhaps quite different, would allow access to the value of the last node of one of TEX's internal lists. Generalises TEX's present mechanism whereby only a subset of nodes can be accessed. MWN.

`\readline` ⟨integer⟩ `to` ⟨cs⟩

Allows a single line to be read from an input file as if each character therein had catcode 12. [6] Intended to be used for verbatim copying operations, in conjunction with `\scantokens`, or to allow error-free parsing of 'foreign' (non-TEX) files.

`\everyeof {`⟨balanced text⟩`}`

Provides a hook whereby the contents of a token list register may be inserted into TEX's reading orifice when end-of-file is encountered during file reading. Would not be invoked if the file indicated logical e-o-f through the medium of `\endinput`. Proposed by Phil to allow clean processing of file-handling code which requires a (sequence of characters yielding) `\else` or `\fi` to be found in a file, where no such sequence can be guaranteed.

`\listing` (internal integer register)

Would allow the generation of a listing containing (for example) TEX's analysis of current brace depth, macro nesting, etc. Different positive values would allow different amounts of information to be generated. Would the TEX community like such a feature?

`\defaultextension`

Would allow TEX's present hard-wired behaviour of appending `.tex` to a filename not possessing an explicit extension to be modified, allowing an alternative extension to be specified. Would this be of use to the L2ε/L3 team, and/or to the TEX world in general?

⟨fixed point arithmetic⟩

Several of the above ideas cannot be implemented at the moment, as they would allow access to the 'forbidden area' of machine-dependent arithmetic. If TEX's present floating point calculations were replaced by Knuth's fixed-point arithmetic proposals, then there would no longer be a forbidden area and all such ideas could, in principle, be implemented.

Notes:

[1] 'Extensions' are basically new primitives which have no effect on the semantics of existing TEX documents, except insofar as any document which tests whether such a primitive is, in fact, undefined, will clearly obtain opposite results under TEX and ε-TEX; 'enhancements' are more fundamental changes to the TEX kernel which may affect the semantics of existing TEX documents even if no new primitive is used or even tested. Such changes may be, for example, differences in the construction of TEX's internal lists, or perhaps different hyphenation or ligaturing behaviour.

[2] It is currently proposed that all enhancements be disabled by ε-IniTEX immediately prior to the execution of `\dump`. This decision was taken based on the advice of Frank Mittelbach.

[3] Question: should there, in fact, be an ε-plain (or ε-LATEX) format, or should there simply be an `e-plain.tex` file which can be loaded by a user document? Peter votes for an `e-plain.tex` file that will `\input plain.tex` but no hyphenation patterns.

[4] Should ε-TEX allow access to more powerful operators than just `+`, `-`, `*` and `/`?

[5] 'Arrays' are not very obvious in TEX at the moment, although there are, for example, `\fontdimen`s and such-like. But should these have fixed bounds (as in 256 count registers, for example), or arbitrary upper bounds (as in font dimens, if the 'extra' elements are assigned as soon as the font is loaded). Or should they be finite-but-unbounded, as in `\parshape`, wherein the first element indicates the number of elements which follow? These questions are applicable to marks as well as to penalties . . .

[6] Should spaces have catcode 10 for this operation? Peter thinks so, but based on existing simulations of this operation, Phil is more inclined to think they should have catcode 13.

◇ Philip Taylor
The Computer Centre, RHBNC
University of London, U.K.
<P.Taylor@Vax.Rhbnc.Ac.Uk>