Présentation du projet ε-TEX *

Philip TAYLOR
Directeur technique du projet NTS
P.Taylor@Vms.Rhbnc.Ac.Uk

Traduit de l'anglais par Michèle Jouhet et Sonia Jaroso

Résumé. Après quelques années d'études et de réalisation, ε-TEX est maintenant opérationel. De nombreux dispositifs ont été ajoutés à TEX pour en étendre ses possibilités. Le projet *New Typesetting System* se poursuit. Une version 2 d'ε-TEX est désormais à l'étude.

Abstract. *After a few years of discussions and implementation, the first version of ε-TEX was released in 1996. A lot of new features were added to TEX to increase its functionality. A second version is currently being designed.*

Le projet ε-TEX a vu le jour grâce à l'instigation de Joachim Lammarsch et sous les auspices de DANTE e.V. lors d'une réunion de DANTE à Hambourg en 1992. L'idée de ce projet était – et reste – de conserver tout ce qu'il y a de bon dans TEX tout en se libérant des contraintes que Knuth impose sur l'évolution de TEX lui-même. Pour cette raison, le projet s'est appelé NTS – abréviation de *New Typesetting System* – pour souligner que nous ne violons pas les souhaits de Knuth et que TEX reste ainsi entièrement sous sa responsabilité ; en effet, nous avons reçu la bénédiction de Knuth pour approfondir l'idée de NTS et ε-TEX. Il a même été jusqu'à faire certaines suggestions « qu'il aurait pu incorporer lui-même, s'il n'avait décidé de geler l'évolution de TEX » Nous avons, bien sûr, essayé d'incorporer ses suggestions, même si nous n'y sommes pas encore entièrement parvenus.

1. ε-TEX et NTS

Avant d'entrer plus dans les détails techniques, voyons la raison du nom « Groupe NTS » alors que le projet qui est présenté ici s'appelle « ε-TEX ».

∗ Ce texte, rédigé pour être prononcé lors des *Journées GUTenberg* (Strasbourg, 26–28 mai 1997), n'a peut-être pas toujous la forme voulue pour une communication écrite [Ndlr]. Sur ε-TEX, voir `http://www.rhbnc.ac.ul/e-TeX/`.

Pendant les premières délibérations du groupe, nous avons adopté l'idée de Joachim Schrod qui préconisait que, plutôt que de modifier le source Web de TEX nous devrions d'abord développer une nouvelles réalisation de TEX en utilisant un langage moderne de prototypage rapide comme LISP, CLOS ou PROLOG. La philosophie de Joachim était que le langage Web de TEX n'était pas assez structuré. Une réalisation plus structurée permettant des changements radicaux et présentant une interface bien définie entre les modules était donc nécessaire si nous ne voulions pas nous limiter à quelques changements purement élémentaires du système TEX. Le groupe a trouvé cette idée judicieuse mais a vite réalisé que l'effort nécessaire pour mettre en œuvre une nouvelle réalisation de TEX en partant de zéro était trop lourd pour des volontaires. Si nous voulions obtenir des résultats dans un délai raisonnable, il était essentiel que notre groupe puisse embaucher un ou plusieurs programmeurs qui se chargeraient de cette nouvelle mise en œuvre. Comme le groupe ne disposait d'aucun budget propre et comme il dépendait financièrement de la bonne volonté de DANTE, c'est à contrecœur que nous avons décidé que cette activité serait mise en sommeil jusqu'à ce que nous ayons les ressources financières nécessaires. En attendant, le groupe a décidé d'adopter une ligne plus conservatrice, évolutive plutôt que révolutionnaire, et de concentrer tous ses efforts sur ce projet avec une ambition moins radicale.

Récemment, le futur du projet NTS est devenu beaucoup plus clair. En effet pendant la dernière réunion de DANTE en février dernier à Munich, le Conseil d'administration et les membres de DANTE ont accepté de doter le projet NTS de 30 000 DM pour nous permettre de payer un programmeur plein-temps en République tchèque pendant un an. Il travaillera sur le projet NTS sous la supervision directe du Prof. Jiří Zlatuška, président de la faculté d'informatique à l'université Masaryk à Brno.

En dépit de ce changement significatif dans l'état du projet NTS, le groupe n'a pas abandonné, ni même réduit ses efforts en ce qui concerne ε-TEX et nous sommes tombés d'accord sur les spécificités (provisoires) de la seconde version (V2) que nous envisageons de mettre en œuvre dans les mois prochains. Nous avons l'intention de livrer la mise à jour V2 d'ε-TEX une année après la mise à disposition d'ε-TEX V1, c.-à-d. début novembre 1997.

Pour clarifier la situation, NTS se réfère à un projet qui a pour but une mise en œuvre complètement nouvelle de TEX d'une façon modulaire, en utilisant un langage moderne de prototypage rapide, et en considérant en détail comment chaque module peut être amélioré. Parmi les modules possibles à améliorer il y a l'interface utilisateur, le langage de programmation utilisateur, le moteur de composition lui-même, et / ou les composants du moteur de composition comme les algorithmes pour couper les lignes ou les pages.

D'un autre côté ε-TEX se réfère au travail en cours qui cherche à développer une réalisation en source Web de TEX de manière à ce qu'elle soit, et reste, 100 % compatible avec TEX lui-même. Le ε d'ε-TEX peut se comprendre comme *évolution*, *extension*, *enrichissement* et *Européen* ; sa typographie : ε (epsilon) souligne qu'il s'agit d'un petit pas évolutif en partant de TEX lui-même et non d'un changement fondamental de conception.

Donc, le travail du projet NTS pourra commencer dans un futur proche, mais ε-TEX c'est le présent ! Après approximativement trois années de développement et de tests, nous avons livré ε-TEX à la réunion de DANTE à Hambourg à la fin de 1996. Quelques modifications légères ont ensuite été apportées. Malgré le grand nombre de connections réseau faites sur le site de référence WEB d'ε-TEX, nous n'avons reçu aucun rapport d'erreur ! Cela veut peut-être dire qu'à l'heure actuelle personne n'utilise ε-TEX. Toutefois, nous espérons que cela signifie qu'ε-TEX est sans erreur, autant que faire se peut.

2. Trois modes de fonctionnement

Quelles sont les caractéristiques d'ε-TEX ? Tout d'abord, ε-TEX est 100% compatible avec TEX : vous pouvez l'utiliser pour traiter tous vos anciens documents – le résultat sera identique à ce que vous auriez obtenu avec TEX, jusqu'à la compatibilité au niveau du test *TRIP*. Une fois que vous serez convaincus qu'ε-TEX peut faire tout ce que fait TEX, vous pouvez essayer les extensions : il y en a une trentaine, chacune ayant pour but de simplifier la programmation en TEX. Même lorsque ces extensions sont activées – c.-à-d. lorsqu'ε-TEX fonctionne en « mode étendu » – ε-TEX traitera tous les documents existants d'une façon identique à TEX, dans la mesure bien sûr où ces documents ne font pas référence, par inadvertance, à l'une des nouvelles primitives d'ε-TEX. Finalement, si vous désirez, vous pouvez utiliser ε-TEX en « mode enrichi » et ainsi accéder à de nouvelles possibilités qui sont trop radicales pour être 100% compatibles avec TEX. Dans la première version il n'y a qu'un seul enrichissement, `TeX--XeT`, basé sur un précédent système bi-directionnel de composition appelé `TeX-XeT` développé par Don Knuth et Pierre McKay. L'avantage de `TeX--XeT` est d'utiliser le format DVI standard en réalisant toutes les opérations de façon interne et ainsi de ne pas nécessiter de pilote spécifique, contrairement à `TeX-XeT`.

Pour récapituler, ε-TEX connaît trois modes d'opérations. Ce sont le mode de compatibilité où ε-TEX se comporte d'une façon identique à TEX, y compris une compatibilité absolue au niveau *TRIP* ; le mode étendu où ε-TEX offre une trentaine de nouvelles primitives tout en restant 100% compatible avec TEX, même si ε-TEX ne passe plus le test *TRIP* à cause de la présence de ces nouvelles primitives ; le mode enrichi où la compatibilité stricte a été sacrifiée

pour permettre des possibilités supplémentaires qui sont fondamentalement incompatibles avec TEX. Le choix entre compatibilité et mode étendu se fait au moment de la génération du format : une fois que le format est sauvé sous forme de cliché mémoire (*format dump*) il contient un indicateur (*flag*) qui spécifie en quel mode il est supposé fonctionner et ce format ne peut être utilisé dans l'autre mode. Le choix entre les modes étendu et enrichi se fait à l'exécution : même si un enrichissement a été activé pour la génération du format, celui-ci sera automatiquement désactivé juste avant que le *dump* soit généré. Ainsi quand ce format est utilisé l'enrichissement est désactivé au moment de l'initialisation. Il est à noter que l'on peut seulement entrer en mode enrichi en passant par le mode étendu, non par le mode de compatibilité.

3. Les nouvelles commandes

Les extensions de la première version d'ε-TEX peuvent être classées en six groupes distincts, plus un septième qui n'est accessible qu'en mode enrichi. Ces groupes sont :

- généralisation du concept de `\mark` ;
- contrôle supplémentaire du développement (*expansion*) ;
- possibilité de rebalayer (*re-scan*) un texte déjà lu ;
- requêtes d'environnement ;
- aides supplémentaires au débogage ;
- primitives diverses ;

et

- composition bi-directionnelle : primitives de `TeX--XeT`.

Commençons par parler des éléments qui composent ces sept groupes avant d'enchaîner sur nos projets pour ε-TEX V2.

Tout d'abord, nous avons commencé par éliminer d'ε-TEX les constantes fixes qui limitent le langage TEX actuel : nous avons généralisé le concept de `\mark` en introduisant un tableau `\marks` contenant 256 éléments dans la première version. Toutes les variables apparentées à `\mark` ont également été généralisées en tableau à 256 éléments.

Puis nous avons ajouté plus de vingt nouvelles primitives que nous allons décrire brièvement :

\protected est un préfixe qui peut être utilisé pendant la définition d'une macro. Une macro définie comme `\protected` ne se développera pas au cours des opérations `\edef` ou `\write` ou de toute autre opération similaire où, normalement, il y a expansion. Toutefois, la macro se développera si elle atteint *l'estomac* de TEX (par exemple pendant le processus de composition).

\detokenize est destiné à être utilisé juste avant une liste de *tokens* délimitée par des accolades (un texte général dans la terminologie du *TEXbook*). Il y aura développement pour produire une séquence de *tokens* de type caractère ayant un code de catégorie 10 ou 12 correspondant aux caractères qui composent les *tokens* de la liste non développée.

\unexpanded est également destiné à être utilisé juste avant une liste de *tokens* délimitée par des accolades mais le développement produit dans ce cas les véritables *tokens* de la liste. Si TEX exécute une commande `\edef` ou `\write` ou une opération similaire, il ne se produira pas d'expansion supplémentaire. Toutefois si ces *tokens* atteignent l'*estomac* de TEX (par exemple lors du processus de composition) alors ils se développeront normalement.

\readline est similaire à `\read`, mais traite chaque caractère comme s'il appartenait au code de catégorie 10 ou 12 ; le texte lu peut alors être balayé et rebalayé dans différents contextes de catcode en utilisant la primitive `\scantokens`.

\scantokens qui doit être suivi d'une liste de *tokens* délimitée par des accolades, décompose cette liste de *tokens* en une séquence de caractères correspondants, comme si cette liste était écrite dans un fichier sans être développée. Puis on applique le mécanisme `\input` de TEX à cette séquence de caractères, en recomposant les *tokens* et utilisant le contexte courant de `\catcode`.

\currentgrouplevel est un entier interne en lecture seulement qui correspond au niveau de groupe au moment de l'appel ; autrement dit on obtient le niveau courant d'imbrication des groupes.

\currentgrouptype est un entier interne en lecture seulement qui contient le type du groupe du niveau le plus profond, un entier entre 0 et 16. Ces nombres peuvent être transformés en chaîne de caractères en utilisant des commandes de la bibliothèque de macros d'ε-TEX.

\ifcsname a la même syntaxe que la commande `\csname` de TEX, mais il s'agit d'un `\if` booléen, dont la valeur est vraie (*true*) si et seulement si cette éventuelle séquence de contrôle est déjà connue d'ε-TEX.

\ifdefined est similaire à `\ifcsname` mais prend comme paramètre une séquence de contrôle où un caractère actif ; sa valeur est *true* si et seulement si la séquence de contrôle ou le caractère actif sont déjà connus d'ε-TEX.

\lastnodetype est un entier interne en lecture seulement qui correspond au type du dernier nœud de la liste courante ; c'est un entier entre –1 et 15 ou plus. Ces nombres peuvent être transformés en chaîne de caractères en utilisant des commandes de la bibliothèque de macros d'ε-TEX.

\eTeXversion est un entier interne en lecture seulement contenant la partie entière de la combinaison version / révision d'ε-TEX.

\eTeXrevision est une primitive qui se développe pour produire une séquence de *tokens* de type caractère avec code de catégorie 12 représentant la partie fractionnelle de la combinaison version / révision d'ε-TEX.

\showtokens doit être suivie par une liste de *tokens* délimitée par des accolades. Cette commande permet ainsi d'afficher, d'un manière simple, le contenu d'un élément donné de la famille des tableaux `\marks`. De nombreuses autres applications potentielles peuvent être envisagées.

\interaction est une commande qui permet l'accès en lecture-écriture au mode d'interaction courant. L'attribution d'une valeur numérique mettra le mode correspondant, alors que le mode courant peut être obtenu en interrogeant sa valeur. La bibliothèque macros d'ε-TEX dispose de définitions symboliques pour ces valeurs.

\showgroups demande à ε-TEX d'afficher le niveau et le type de tous les groupes actifs à partir du point où la commande a été donnée.

\tracingassigns est une variable qui, si elle a une valeur positive différente de zéro, demande à ε-TEX d'afficher la valeur des registres avant et après chaque affectation. TEX standard affiche uniquement la nouvelle valeur et non l'ancienne.

\tracinggroups est une commande qui peut aider au débogage de problèmes dans le cas de groupes qui ne se terminent pas (*runaway*). Si cette variable a une valeur positive différente de zéro, ε-TEX signale chaque entrée et sortie de groupe.

\tracingifs est un outil de débogage pour les développements d'environnements conditionnels. Si cette variable a une valeur positive différente de zéro, ε-TEX montre en détail le flux de contrôle à travers les instructions conditionnelles.

\tracingscantokens est une variable qui, si elle a une valeur positive différente de zéro, demande à ε-TEX d'afficher une parenthèse gauche suivie d'un espace à chaque invocation de la commande `\scantokens` ; la parenthèse droite correspondante sera affichée lorsque l'opération de balayage sera terminée.

\tracingcommands est défini dans TEX pour contrôler le degré de « verbosité ». On dispose des valeurs entre zéro et deux. ε-TEX offre en plus un niveau supérieur et plus de détails en spécifiant une valeur supérieure à deux.

\everyeof est, comme le dit Knuth à la fin du fichier `tex82.bug`, une de ses « idées qui peuvent être bonnes » . Cette commande est similaire aux autres primitives

de type `\every...` et elle prend comme paramètre une liste de *tokens* délimitée par des accolades dont les *tokens* sont insérés à chaque fois qu'une fin de fichier est rencontrée.

`\middle` est analogue aux primitives `\left` et `\right` de TEX. Cette commande indique que le délimiteur qui suit doit servir à la fois de délimiteur droit et gauche. Un espace propre à un délimiteur droit sera inséré par rapport à l'atome précédant et similairement un espace propre à un délimiteur gauche sera inséré par rapport à l'atome suivant.

`\unless` inverse le sens des conditionnels booléens. Ainsi, `\unless\ifeof` affiche *true* tant que la fin du fichier n'est pas encore atteinte.

Le dernier groupe de primitives comprend celles qui ne sont accessibles que lorsqu'ε-TEX fonctionne en mode enrichi. Un programme ε-TEX entre dans le mode enrichi lorsque l'on met une des variables du mode enrichi à une valeur positive différente de zéro. Dans la première version d'ε-TEX il y a seulement `\TeXXeTstate`. Dès que cette variable est mise à une valeur positive différente de zéro, cinq autres primitives deviennent disponibles : `\beginL`, `\beginR`, `\endL`, `\endR` et `\predisplaydirection`. Notez que si l'une de ces primitives est utilisée alors qu'ε-TEX n'est pas en mode enrichi, un message d'erreur est affiché.

`\TeXXeTstate` est un entier interne en lecture seulement qui, lorsqu'il a une valeur positive différente de zéro, permet l'utilisation des primitives de `TeX--XeT`

`\beginL` indique le début d'une zone qui doit être composée de gauche à droite.

`\endL` indique la fin d'une zone qui doit être composée de gauche à droite.

`\beginR` indique le début d'une zone qui doit être composée de droite à gauche.

`\endR` indique la fin d'une zone qui doit être composée de droite à gauche.

`\predisplaydirection` est un entier interne en lecture et écriture qui contient la direction de composition du dernier paragraphe avant un affichage mathématique. Sa valeur est utilisée pour gérer le positionnement d'éléments tels que les numéros d'équations ; elle peut être modifiée pour changer ce positionnement si nécessaire.

4. Une bibliothèque de macros

Bien que la majeure partie de nos efforts pour développer ε-TEX ait été consacrée à l'introduction de nouvelles fonctionnalités au langage TEX, nous avons

néanmoins pris un peu de temps pour créer une petite bibliothèque de macros pour l'accompagner. Essentiellement, il s'agit d'une extension du fichier source du format `plain`, où certaines définitions existantes ont été modifiées pour tenir compte des nouvelles primitives. Nous avons également saisi l'occasion d'ajouter deux possibilités qui, d'après nous, pourraient être appréciées par la communauté TEX toute entière. Ainsi nous avons différé la lecture des motifs de césure et des exceptions jusqu'au moment où un mécanisme de traitement de langage naturel soit défini (donc, les motifs et les exceptions sont maintenant associés à un langage donné, au lieu d'être défini dans le vide). Nous avons également ajouté un support pour les fichiers bibliothèque ε-TEX afin qu'il soit possible de charger des modules au lieu de charger des fichiers complets. Le mécanisme de traitement de langage n'est pas lié à un système particulier pour traiter les langages naturels : il peut être utilisé avec Babel, par exemple, ou avec tout autre système de traitement de langage naturel. En outre, des points d'attache (*hooks*) permettent à l'utilisateur d'introduire son propre code avant et après le choix du langage. De cette façon nous espérons offrir un environnement de traitement de langage suffisamment flexible pour satisfaire les besoins de la majorité des groupes nationaux d'utilisateurs TEX. Ceci ne signifie pas que nous ne percevions pas de rôle pour Ω (*Omega*) ; au contraire, il est clair d'après les discussions sur la liste *Omega*, qu'il existe un réel besoin pour un système d'une telle complexité. Toutefois, nous considérons que pour les environnements de composition dans lesquels seul un accès aux principales langues occidentales est nécessaire ε-TEX se révélera suffisant.

5. Versions disponibles

Lorsqu'ε-TEX fut annoncé fin 1996, deux versions de référence étaient disponibles : celle de Peter Breitenlohner *Public ε-TEX*, une réalisation en Turbo Pascal pour la famille IBM PC sous MS / DOS ; et celle de Christian Spieler *VMS ε-TEX*, une réalisation en Pascal pour la famille des machines VAX / VMS et AXP / VMS. Depuis la sortie d'ε-TEX, une version a d'abord été préparée pour l'environnement Commodore Amiga, puis pour l'environnement Windows 95 / NT. Fin février 1997, Bernd Raichle a annoncé que sa version d'ε-TEX pour Web2C Version 7 était également prête à être distribuée ; Bernd, qui travaille à l'université de Stuttgart, nous a informé qu'Eberhard Mattes a l'intention de sortir, sous peu, une version combinée ε-TEX / ML-TEX pour la famille des IBM PC.

Voilà, pour résumer, l'état actuel d'ε-TEX. Passons maintenant aux idées qui sont à l'étude pour ε-TEX V2.

6. Nouveaux dispositifs envisagés

– Premièrement, nous étudions comment introduire la possibilité de vérifier si un certain caractère existe dans une police donnée : `\iffontchar` prend deux paramètres, la police et le numéro du caractère, et aura la valeur « *true* » si et seulement si ce caractère existe dans la police. Ainsi il sera facile de définir une macro `\ifchar` pour vérifier l'existence d'un caractère dans la police courante.

– Il existe quatre primitives apparentées pour connaître les dimensions d'un caractère particulier dans une police donnée : `\fontcharwd`, `\fontchardp`, `\fontcharht` et `\fontcharic`. Chacune d'entre elles prend deux paramètres : une police et un numéro de caractère. Elles retournent, respectivement, la largeur, la profondeur, la hauteur et la correction italique. Ce résultat ne peut être obtenu en composant le caractère dans une boîte et puis en mesurant celle-ci car une ou plusieurs dimensions peuvent être négatives. Comme avec `\iffontchar`, il est facile de définir des macros qui exécutent les mêmes tâches pour la police courante.

– Nous sommes encore en discussion sur l'opportunité d'incorporer une commande `\iffont` qui aurait comme paramètre le nom externe d'une police et retournerait la valeur *true* uniquement si un fichier de métrique de police (`.tfm`, pour *TEX font metric*) du même nom peut être ouvert ; tous les membres du groupe ne sont pas convaincus de la nécessité d'une telle commande, ainsi nous vous invitons à donner vos commentaires sur cette idée.

– Dans ε-TEX V1, nous avons fourni des diagnostics supplémentaires qui indiquent à quelle ligne un groupe a été ouvert, s'il n'a pas été fermé à la fin du programme. Dans ε-TEX V2, nous proposons d'afficher en plus le nom du fichier, bien que des différences de réalisation puissent nous empêcher de restituer le chemin complet du fichier. Nous envisageons également d'indiquer si une fin de groupe manque à la fin d'un fichier. Cette fonction ne peut être qu'un contrôle relativement limité du niveau du groupe plutôt qu'une vérification complète permettant de s'assurer qu'on quitte le fichier à l'intérieur du même groupe que celui d'entrée.

– Nous espérons éviter quelques-uns des nombreux problèmes qui assaillent actuellement les développeurs de routines d'*output* en leur donnant accès aux éléments qui sont écartés après un changement de page. Ces éléments seront disponibles, jusqu'au changement de page suivant dans une nouvelle boîte réservée appelée `\pagediscards`. Par contre, nous considérons qu'actuellement il est impossible de donner un accès similaire aux éléments écartés lors d'une coupure de ligne.

– Dans un domaine similaire, nous espérons rendre `\vsplit` plus utile en permettant au programmeur de supprimer la colle `\topsplitglue` qui est actuellement introduite par TEX. Nous travaillons à une réalisation très générale qui proposera quatre destructeurs de liste, dont un figure déjà dans ε-TEX V1. Les

quatre destructeurs sont `\firstnodetype`, `\lastnodetype`, `\removefirstnode` et `\removelastnode`. Dans une prochaine réalisation, il est probable que nous donnerons l'accès à ces nœuds, avec possibilité de les supprimer tout simplement.

– Dans TEX, `\parshape` est en réalité une quantité en écriture seulement ; seul le nombre d'entrées dans le `\parshape` courant peut être interrogé par la suite. Dans ε-TEX V2, nous envisageons de donner un accès en lecture à toutes les dimensions de `\parshape`, même si nous ne savons pas encore avec certitude si cela se fera au moyen d'une seule commande `\parshapedimen` ou par une paire de commandes `\parshapewidth` et `\parshapeindent`. Quelle que soit la solution choisie, la commande sera indexée par un entier afin de pouvoir retourner une dimension particulière du `\parshape` actif.

– De façon identique aux primitives `\currentgrouplevel` et `\currentgrouptype` qui figurent dans ε-TEX V1, nous avons l'intention d'ajouter à ε-TEX V2 de nouvelles commandes permettant d'interroger le flux de contrôle à travers les opérations conditionnelles : `\currentiflevel` et `\currentiftype`. Nous envisageons aussi un troisième commande, `\currentifbranch`, qui permettra au programmeur de déterminer si pendant le développement on se trouve dans le code de la branche `\then` ou `\else` et pour déterminer si le `\if` courant a déjà reçu assez de *tokens* pour décider quel branchement doit être pris. Nous pourrons peut-être utiliser cette commande également pour déterminer quel branchement `\or` a été choisi dans un `\ifcase`, mais nous ne sommes pas encore sûrs que ce soit possible.

– Aussi, par analogie avec la primitive `\showgroups` d'ε-TEX V1, nous avons l'intention de fournir `\showifs` en V2.

– Nous envisageons, sans être pour le moment parvenu à nous mettre d'accord, d'introduire une nouvelle classe d'alignement, `\malign`, qui donnerait (si elle était acceptée) une primitive d'alignement mathématique.

– Toujours sur le thème de la suppression des limites fixes d'ε-TEX, nous considérons la possibilité de supprimer ces limites fixes pour le nombre des registres de comptage, de *dimen*, de *skip*, de *muskip*, des listes de *tokens* et pour le nombre de boîtes. Si nous réussissons à introduire ces changements, nous supprimerions probablement en même temps les limites fixes sur le nombre de marques (*mark*).

– Pendant les discussions de Brno, Don nous a demandé de considérer comment nous pourrions permettre un meilleur contrôle du « relâchement » (*looseness*) de la dernière ligne d'un paragraphe. Bien que TEX la compose à sa longueur naturelle (si `\parfillskip` est infini, ce qui est généralement le cas), Don confirme que cette ligne a, d'habitude, le même relâchement que la ligne précédente. Nous considérons actuellement la possibilité d'avoir non seulement ces deux conditions limites, mais également tout le continuum de valeurs entre ces deux extrêmes, vraisemblablement à travers une commande `\finaladjdemerits`.

– Nous essayons d'économiser de l'espace dans les piles dans ε-TEX V2 en éliminant les affectations inutiles (c'est-à-dire, des valeurs qui seront remises à la même valeur à la sortie d'une boucle). Cela n'exige pas de nouvelle primitive, il s'agit simplement d'une optimisation interne.

– Nous pensons que le message *lost chars* qui est entré actuellement uniquement dans le fichier compte-rendu (*log*) est suffisamment important pour être affiché également à l'écran ; en conséquence, nous étendons la sémantique de la commande `\tracinglostchars` afin que, pour une valeur supérieure à 1, le message apparaisse à l'écran ainsi que dans le fichier compte-rendu.

– Nous réfléchissons à la manière de permettre au programmeur d'ajouter des éléments – par exemple des hirondelles (*crop marks*) – à une boîte avant qu'elle ne soit envoyée dans le fichier de sortie (au `\shipout`), même au cas où l'*output* routine a été rendue inaccessible (par un format complexe tel que LATEX, par exemple). Nous réaliserons cette fonction probablement à l'aide d'une commande `\outpage`, similaire à `\output`, et la routine qui lui est associée sera insérée au moment où la boîte est envoyée dans le fichier de sortie. Cela permettra également au programmeur de superposer à la boîte des éléments devant être placés à des endroits déterminés de la page. Nous continuons les discussions pour déterminer la meilleure façon de réaliser `\outpage`. Ainsi nous vous invitons à nous dire si vous considérez que cette primitive doit être récursive, autrement dit une primitive `\shipout` appelée à l'intérieur d'une commande `\outpage` devrait-elle automatiquement invoquer une nouvelle fois `\outpage` à moins que la commande `\outpage` précédente n'ait déjà effacé le registre de listes des *tokens* `\outpage` ?

– Pour permettre l'utilisation de formats très différents comme ceux de la famille Plain TEX, LATEX, $\mathcal{AMS}$-TEX ou LA$\mathcal{MS}$-TEX (ATML, par exemple), nous pensons offrir une alternative pour la valeur par défaut actuelle qui rajoute `.tex` au nom de tout fichier spécifié sans extension explicite lors des commandes `\input`, `\openin` ou `\openout`. Nous envisageons de permettre la possibilité de spécifier explicitement l'extension par défaut. Dans ce cas encore nous comptons sur vos commentaires pour nous indiquer si vous pensez que cela vous parait utile.

– Une proposition qui revient souvent, mais sur laquelle nous ne sommes pas encore entièrement tombés d'accord, est l'idée d'une primitive `\evaluate`, qui permettrait des opérations arithmétiques dans la « bouche » d'ε-TEX. Bien que nous considérions tous qu'une commande comme `\evaluate` s'avérerait extrêmement utile, nous sommes conscients que pour garantir des résultats reproductibles sur toutes les plates-formes il faudra la coder sans avoir recours à l'arithmétique en virgule flottante de la plate-forme concernée. Cela impliquerait donc une réalisation très limitée ou alors il faudrait un temps considérable pour développer un progiciel complet d'arithmétique en virgule flottante. Nous continuons les discussions pour décider si nous adoptons l'une ou l'autre (ou aucune) de ces solutions pour ε-TEX V2.

– Pendant nos discussions à Brno, Don nous a demandé d'examiner comment donner un plus grand contrôle sur l'espacement dans les fractions afin de permettre, par exemple, une dépendance moindre de l'utilisation de constructions de bricolage comme `\sub \strut`. Nous n'avons pas encore développé de modèle approprié pour ce cas, mais nous continuons à étudier différentes possibilités.

Parlons enfin de deux propositions majeures : MlTEX et TeX2pdf. Nous savons bien que tous les membres de GUTenberg connaissent MlTEX et qu'ils sont nombreux à en dépendre pour leur travail quotidien. Ce qu'il faut savoir c'est que Mike Ferguson, l'auteur et le créateur de MlTEX, a demandé à Bernd Raichle d'assumer la responsabilité de son développement futur. Comme celui-ci est un des pivots du projet ε-TEX, vous pouvez être sûrs que nous ne négligerons pas MlTEX lorsque nous considérerons le futur d'ε-TEX, même si nous sommes conscients que la réalisation actuelle a quelques faiblesses, notamment en ce qui concerne la synchronisation de certaines opérations. Nous espérons être en mesure d'améliorer sa mise en œuvre, mais nous tenons à vous assurer que les utilisateurs de MlTEX auront la possibilité de participer aux discussions sur ces améliorations avant que toute décision soit prise.

TeX2pdf est un projet relativement récent, entrepris par Han The Thanh à l'Université Masaryk à Brno. Nous sommes vraiment impressionnés par le travail de Thanh, mais nous ne sommes pas convaincus que le modèle qu'il a adopté soit la meilleure façon de procéder. En particulier, nous pensons que des changements de l'ampleur requise pour mettre en œuvre sa version actuelle pourraient introduire dans TEX des erreurs subtiles, difficilement détectables et qui pourraient avoir un effet défavorable sur sa stabilité. Par conséquent, nous sommes intéressés par l'étude d'un modèle plus simple qui confierait la majeure partie du traitement à un post-processeur similaire au programme `dvi2pdf` de Sergey Lesenko. Toutefois, comme Thanh l'a déjà fait remarquer, cela serait incompatible avec l'adoption d'une variante de l'algorithme HZ qui, selon lui, pourrait être incorporé dans sa version actuelle. Nous examinons encore différentes variantes dans ce domaine, mais nous aimerions vraiment soutenir le concept de TeX2pdf sous quelque forme que ce soit.

Conclusion et remerciements

Fruit d'un effort d'un groupe international sur plusieurs années ε-TEX est désormais utilisable sur plusieurs plate-formes. La parole est maintenant donnée aux utilisateurs qui l'essayeront et / ou qui souhaitent y voir certaines améliorations.

L'équipe du projet NTS remercie tout particulièrement toutes celles et ceux qui contribuent à le faire avancer.