Cahiers GUTenberg

TEX 3.0 OU LE TEX NOUVEAU VA ARRIVER

Donald E. KNUTH

Cahiers GUTenberg, nº 4 (1989), p. 39-45.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1989___4_39_0>

© Association GUTenberg, 1989, tous droits réservés.

L'accès aux articles des *Cahiers GUTenberg*
(http://cahiers.gutenberg.eu.org/),
implique l'accord avec les conditions générales
d'utilisation (http://cahiers.gutenberg.eu.org/legal.html).
Toute utilisation commerciale ou impression systématique
est constitutive d'une infraction pénale. Toute copie ou impression
de ce fichier doit contenir la présente mention de copyright.

TEX 3.0 ou le TEX nouveau va arriver*

Donald E. KNUTH

Professor of The Art of Computer Programming,
Computer Science Department,
Stanford University,
Stanford, CA 94305-2140 USA

Résumé Voici la traduction d'un article de D. E. Knuth, à paraître dans le prochain TUGboat, que nous avons reçu par courrier électronique ; s'agissant de la prochaine version de TEX qui doit pouvoir (entre autres) traiter les caractères accentués, il nous a semblé important d'en faire profiter, au plus tôt, la communauté TEXienne francophone même si la version définitive devait recevoir encore quelques modifications. — N.D.L.R.

Abstract *This is the french translation of D. E. Knuth's article:* The new versions of TEX and METAFONT: Draft description, *oct. 89, to appear in the next issue of* TUGboat.

Pendant plus de cinq ans, je suis resté fermement convaincu qu'un système stable était bien préférable à un système en constante évolution. Cependant, au cours du congrès du TUG en août 89 à Stanford, je me suis laissé convaincre qu'il fallait faire un dernier ensemble de modifications pour accorder parfaitement TEX et METAFONT avec leur philosophie générale et leurs objectifs.

La raison principale de ces modifications est le fait que j'avais fait un mauvais choix dans le codage des caractères, entre un codage sur 7 bits et un codage sur 8 bits. J'avais pensé que le standard d'entrée pour les textes continuerait indéfiniment à être fondé sur un ensemble d'au plus 128 caractères car je ne pensais pas qu'un clavier ayant 256 codes de sortie différents puisse être particulièrement efficace. J'ai eu tort, cela est évident, les développements réalisés en Europe et en Asie en particulier le montrent. Dès que j'eus compris qu'un programme de formattage de texte utilisant un codage d'entrée sur 7 bits semblerait bientôt aussi archaïque que les systèmes antérieurs à 6 bits, j'ai su qu'une révision fondamentale s'imposait.

Comme le choix du système à 7 bits s'était infiltré partout, j'ai dû reprendre chacun des programmes et les réécrire entièrement à la mode '8 bits'. C'était remettre TEX sur la table d'opération et sous le bistouri pour la première fois depuis 1984 et me donner une dernière occasion d'y inclure quelques nouveaux éléments dont j'avais eu l'idée ou qui m'avaient été suggérés par les utilisateurs depuis cette date.

Ces extensions nouvelles sont entièrement compatibles de façon ascendante avec les précédentes versions de TEX et METAFONT (mis à part les quelques exceptions qui sont mentionnées ci-dessous). Ceci signifie que les fichiers-sources qui fonctionnaient sans erreur avec les anciennes versions de TEX et METAFONT continueront à tourner sans erreur avec la nouvelle, et produiront les mêmes sorties.

Cependant quiconque s'aventurera à utiliser les extensions nouvelles ne pourra pas obtenir les résultats souhaités avec les anciennes versions de TEX et METAFONT. Je demande donc à la communauté des utilisateurs de TEX de mettre à jour

*Cet article, paru dans le *TUGboat*, (vol. 10, #3, p. 325 à 328) a été traduit de l'anglais par Alain COUSQUER (Lille) et est publié ici avec l'aimable autorisation de l'auteur et de Barbara BEATON, rédactrice en chef de *TUGboat*.

dès que possible toutes les copies des anciennes versions. Il nous faut détruire radicalement les versions sur 7 bits, même si nous arrivions à faire de très belles choses avec.

Dans cette note, je vais présenter ces modifications une par une et donner les exceptions concernant la compatibilité avec les versions précédentes.

1. Jeu de caractères

On peut maintenant utiliser jusqu'à 256 caractères différents dans un fichier d'entrée. Les codes qui étaient auparavant limités à l'intervalle *0...127* sont compris maintenant dans l'intervalle *0...255*. Tous les caractères sont traités de la même façon ; vous êtes libres d'utiliser n'importe quel caractère pour n'importe quel usage en TEX en fournissant les valeurs appropriées aux paramètres `\catcode`, `\mathcode`, `\lccode`, `\uccode`, `\sfcode` et `\delcode`. Le format *plain* initialise ces valeurs pour les caractères dont le code est supérieur à *127* de la même façon que pour les caractères de ponctuation ordinaires comme '!'.

Il existe une nouvelle convention pour l'entrée de n'importe quel caractère (sur 8 bits) dans TEX quand il n'est pas disponible au clavier : les quatre caractères `^^xy`, où `x` et `y` sont l'un quelconque des symboles hexadécimaux (en minuscules) `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `a`, `b`, `c`, `d`, `e` ou `f`, sont traités par TEX en entrée comme s'il s'agissait d'un seul caractère ayant le code correspondant. Par exemple, `^^80` fournit le caractère de code *128* ; le jeu complet de caractères, de `^^00` à `^^ff`, est ainsi disponible. L'ancienne convention, présentée dans l'appendice C du TEXbook, selon laquelle le caractère de rang *0* était `^^@`, celui de rang *1* *(control-A)* était `^^A`, ..., et le caractère *127*, `^^?`, fonctionne toujours pour les 128 premiers codes, le caractère suivant les '`^^`' ne devant cependant pas être un symbole hexadécimal en minuscules quand le caractère qui suit en est lui-même un.

L'existence de codes sur 8 bits a moins de conséquences pour METAFONT que pour TEX car les jeux de caractères de METAFONT sont dépendants de chaque installation et sont donc propres à celle-ci. Le jeu standard des 95 caractères imprimables tel qu'il est présenté en page 51 du METAFONTbook peut être élargi aux autres caractères comme il est indiqué en page 282, mais ceci est rarement réalisé car cela engendre des problèmes de portabilité. L'opérateur `char` de METAFONT est maintenant défini pour fonctionner *modulo 256* au lieu de *modulo 128*.

2. Tables de césures

Il peut y avoir maintenant jusqu'à 256 tables de césures différentes dans TEX. Il existe un nouveau paramètre entier appelé `\language`, dont la valeur courante indique quelle est la convention de césure en vigueur à ce moment là. Si la valeur de `\language` est négative ou supérieure à *255*, TEX agit comme si cette valeur était égale à *0*.

Quand vous faites la liste des exceptions pour les césures avec la commande `\hyphenation` de TEX, cette liste s'applique uniquement au langage en cours. De la même façon, la commande `\patterns` demande à TEX de mettre en mémoire les nouvelles règles de césures concernant le langage courant ; cette opération n'est possible que dans le programme d'initialisation spécial appelé `INITEX`. Les exceptions aux règles de césures peuvent être ajoutées à n'importe quel moment, mais

les nouvelles règles s'appliquant dans un paragraphe ne peuvent pas être données après la fin de celui-ci.

Quand TEX lit le texte d'un paragraphe, il insère automatiquement des marqueurs *ad hoc* dans la liste horizontale de ce paragraphe à chaque fois qu'il rencontre un caractère dont le `\language` diffère de celui de son prédécesseur. De cette façon, TEX peut savoir quelle est la table de césures à appliquer à chaque mot d'un paragraphe, même si vous jonglez fréquemment avec de nombreux langages.

Les marqueurs spéciaux *ad hoc* sont insérés automatiquement en mode horizontal ordinaire (c'est à dire lorsque vous créez un paragraphe, mais pas quand vous entrez le contenu d'une `\hbox`). Vous pouvez entrer vous-même un marqueur *ad hoc* en mode horizontal strict par la commande `\setlanguage<numero>`. Ceci n'est nécessaire que si vous êtes en train de faire quelque chose d'astucieux, comme sortir d'une boîte un élément destiné à un paragraphe.

3. Contrôle de la taille des césures

TEX possède de nouveaux paramètres, `\lefthyphenmin` et `\righthyphenmin`, qui fixent la longueur du plus petit fragment d'un mot qui peut apparaître au début ou à la fin d'un mot coupé. Auparavant, les valeurs `\lefthyphenmin = 2` et `\righthyphenmin = 3` étaient fixées par TEX et ne pouvaient pas être changées. Maintenant, le format *plain* fournit, par défaut, les anciennes valeurs qui sont toujours celles qui sont recommandées par la plupart des publications américaines ; mais vous pouvez obtenir plus ou moins de césures en diminuant ou en augmentant ces valeurs. Si la somme de `\lefthyphenmin` et `\righthyphenmin` est égale ou supérieure à 63, il n'y a plus aucune césure. Vous pouvez également obtenir ce résultat en utilisant une police pour laquelle `\hyphenchar = -1`, ou en passant à un `\language` pour lequel il n'y a ni règle de césures ni exception.

4. Ligatures plus conviviales

Voici maintenant la modification la plus radicale. Les anciennes versions de TEX ne connaissaient qu'un seul type de ligature dans laquelle deux caractères comme **f** et **i** se transformaient en un simple caractère comme **fi** quand ils se suivaient. Le nouveau TEX reconnaît des constructions bien plus complexes dans lesquelles, par exemple, nous pouvons transformer le **i** suivant un **f** en un ı (i sans point) alors que le **f** reste sans changement : **fı**.

Tout comme avant, vous ne pouvez obtenir des ligatures que si elles existent dans la police que vous utilisez. Jetons donc un coup d'œil aux nouvelles caractéristiques de METAFONT qui permettent de créer ces nouveaux types de ligatures. En METAFONT, le programmeur peut spécifier un *programme de ligature/crénage* pour chaque caractère de la police qu'il est en train de créer. Si, par exemple, la ligature **fi** apparait en douzième position dans la police, le remplacement du **f** et du **i** par **fi** est indiquée en insérant l'instruction `"i"=:12` dans le programme de ligature/crénage pour `"f"` ; c'est la convention de METAFONT qui est actuellement en vigueur.

Les nouvelles ligatures vous permettent de conserver l'un, ou l'autre, ou les deux caractères tout en en insérant un nouveau. Au lieu de `=:` vous pouvez aussi taper `|=:` si vous voulez conserver le caractère de

gauche, `=:|` pour celui de droite ou `|=:|` pour conserver les deux. Ainsi, si le ı (i sans point) est le caractère de rang 16 dans la police, vous pouvez obtenir le résultat dont nous venons de parler en mettant `"i"|=:16` dans le programme concernant le **f**.

Il y a également quatre opérateurs supplémentaires `|=:>`, `=:|>`, `|=:|>` et `|=:|>>`, dans lesquels chaque `>` indique à TEX qu'il doit se décaler d'une position vers la droite. Ainsi, si **f** et **i** ont été remplacés par **fi** comme indiqué précédemment, TEX cherchera à exécuter à nouveau son programme de ligature/crénage pour **f** en insérant éventuellement un crénage devant le ı ou en changeant éventuellement le **f** en un caractère entièrement différent, etc. Si par contre l'instruction avait été `"i"|=:>16` au lieu de celle que nous avons donnée, TEX aurait traité immédiatement la ligature/crénage des caractères suivant le caractère *16* (ı : le i sans point) ; aucune modification supplémentaire n'aurait été faite entre **f** et ı, même si la police avait indiqué quelque chose à cet endroit.

5. Ligatures et extrémités de mots

Toute chaîne de *caractères* lue par TEX en mode horizontal (après exécution des macros) peut être appelée *mot*. (Pratiquement, nous appelons *caractère* dans cette définition soit un caractère dont le `\catcode` est une lettre ou un caractère équivalent, soit une suite de codes de contrôle qui a été définie comme étant égale à un tel caractère, soit une suite de codes de contrôle définie par `\chardef`, soit encore une construction du type `\char<number>`. La nouvelle version de TEX imagine qu'il y a maintenant un caractère *borne gauche* invisible juste avant un tel mot, et un caractère *borne droite* invisible juste après lui. Ces caractères de bordure ne jouent un rôle que si le créateur de la police a spécifié les ligatures et/ou les crénages entre eux et les lettres adjacentes. Le premier ou le dernier caractère d'un mot peut ainsi changer automatiquement de forme.

Un programme de ligature/crénage pour le caractère *borne gauche* est spécifié dans METAFONT en utilisant l'étiquette spéciale `||:` dans une commande de `ligtable`. Une ligature ou un crénage avec le caractère *borne droite* est indiquée en affectant une valeur à la nouvelle variable interne `boundarychar` de METAFONT et en spécifiant une ligature ou un crénage relatif à ce caractère. Les caractères *bornes* peuvent exister ou ne pas exister dans la police comme caractères réels.

Supposons par exemple que nous voulions transformer le *F* initial d'un mot en *ff* si nous sommes en train de faire du vieil anglais ; le créateur de la police peut maintenant écrire en METAFONT :

```
ligtable ||: "F"|:=11
```

si le caractère de code *11* est *ff*. La même instruction `ligtable` peut apparaître dans les programmes de caractères comme (, ‘, ” ou - qui peuvent précéder des chaînes de lettres ; ainsi `Bassington-French` pourrait donner *Bassington-ffrench*.

Supposons que le **s** de notre police soit un **s** d'avant le XIX^{e} siècle qui ressemble à un *f* estropié et que nous ayons un **s** moderne en position *128*, nous pouvons alors convertir les **s** finaux comme le faisait Benjamin Franklin en introduisant des instructions de ligatures comme :

```
boundarychar := 255;
```

```
ligtable "s": 255 =:| 128,
"." =:| 128,
"," =:| 128,
")" =:| 128,
"'" =:| 128,
```

et ainsi de suite (un véritable style ancien devrait également posséder des ligatures pour **ss**, **si**, **sl**, **ssi**, **ssl** et **st** ; comme il serait amusant de créer un style **Computer Modern Old** !).

Le caractère implicite *borne gauche* est ignoré par TEX si vous donnez l'instruction **\noboundary** juste avant le mot. Le caractère *borne droite* est ignoré si vous écrivez **\noboundary** juste après.

6. Plus grande concision des ligatures

Deux **ligtables**, ou plus, peuvent maintenant partager un même code. Pour réaliser ceci en METAFONT, vous indiquez **skipto** <n> à la fin d'une commande **ligtable**, et vous dites alors <n>:: dans une autre. De telles étiquettes locales peuvent être réutilisées ; ainsi vous pouvez dire **skipto** 1 à nouveau après que 1:: ait été écrit, ce qui aura pour effet de provoquer un saut jusquà la prochaine occurrence de 1::. Il y a 256 étiquettes locales, numérotées de *0* à *255*, avec une restriction : 128 ligatures ou crénages au plus peuvent intervenir entre un **skipto** et l'étiquette correspondante.

Le format des fichiers *TFM* a été étendu en conséquence pour permettre plus de 32 500 ligatures/crénages par police (auparavant, il y avait une limite effective de 256).

7. Meilleure allure du « relâchement »

Il existe maintenant un meilleur moyen d'éviter les messages

'{**overfull box** ...'

pour ceux qui ne veulent pas réentrer dans leurs documents pour corriger à la main les coupures de lignes impossibles. Auparavant, ils essayaient de le faire en mettant **\tolerance=10000**, mais le résultat était affreux, car TEX avait tendance à rassembler tous les coefficients de laideur (**badness**...) dans une ligne horrible unique (TEX considère les coefficients de laideur supérieurs ou égaux à 10 000 comme étant infiniment mauvais et toutes ces infinités sont égales).

Le nouveau mécanisme est un paramètre de dimension nommé **\emergencystretch**. Si **\emergencystretch** est positif, et si TEX a été incapable de mettre en page un paragraphe sans dépasser les tolérances données, une deuxième passe sur le paragraphe est faite, au cours de laquelle TEX considère qu'une dilatation additionnelle égale à **\emergencystretch** est possible dans chaque ligne. Il en résulte que les coefficients de laideur sont ramenés dans un intervalle où ceux qui étaient infinis deviennent finis ; TEX s'arrangera pour trouver une solution optimale à ce nouveau problème, solution qui sera dans la pratique aussi satisfaisante que possible. La dilatation supplémentaire n'est pas réellement présente ; par conséquent, les boîtes trop vides seront signalées dans les messages d'avertissement sauf si la valeur de **\badness** a été augmentée.

8. Possibilité d'examiner le « coefficient de laideur »

TEX possède un nouveau paramètre entier nommé `\badness` qui enregistre le coefficient de laideur de la dernière boîte construite. Si cette boîte était en débordement, `\badness` deviendra égal à *1 000 000*, sinon sa valeur restera entre *0* et *10 000*.

9. Possibilité d'examiner le numéro de ligne

TEX possède également un nouveau paramètre entier nommé `\inputlineno` qui contient le numéro de la ligne que TEX pourrait afficher dans un message d'erreur si une erreur survenait à ce moment là. La valeur de ce paramètre, ainsi que celle de `\badness`, n'est accessible qu'en lecture, de la même façon que celle de `\lastpenalty`, c'est-à-dire en écrivant `\ifnum\inputlineno>\badness...\fi` ou bien `\the\inputlineno`, mais sans pouvoir leur affecter de nouvelles valeurs.

10. Suppression de l'affichage du contexte des erreurs

Il existe un nouveau paramètre entier nommé `\errorcontextlines` qui spécifie le nombre maximum de paires de doubles lignes affichées dans les messages d'erreur de TEX (en plus de la première et de la dernière qui le sont toujours). Le format *plain* fixe maintenant la valeur de `\errorcontextlines` à 5, mais des ensembles de macros de niveau supérieur peuvent préférer faire l'affectation `\errorcontextlines=1` ou même `\errorcontextlines=0`. Dans ce dernier cas, une erreur qui impliquait trois paires de contextes ou plus apparaîtra dorénavant ainsi :

```
! Error
<quelque part>  \The\top
                \The \top\ line
...
        1.123   \The
                \The \bottom\ line
```

(Si `\errorcontextlines` est négatif, vous ne verrez même pas les '...' précédents.)

11. Recyclage de la sortie

Pour finir, il existe encore un paramètre entier : `\holdinginserts`. Si sa valeur est positive quand TEX insère, dans l'*output routine*, la page courante dans `\box255`, TEX ne mettera rien aux points d'insertion des boîtes correspondantes ; tous les points d'insertion resteront en place. Les personnes qui développeront des *output routines* pourront utiliser ceci quand elles voudront remettre le contenu de la boîte *255* dans la page courante en vue de la recomposer pour y changer `\vsize` ou autre chose.

12. Exceptions à la compatibilité ascendante

Les nouveaux mécanismes de TEX et METAFONT n'ont d'influence que sur un petit nombre de choses qui fonctionnent maintenant de façon différente. Je vais essayer d'en faire la liste ici (à l'exception des cas où le comportement précédent était erroné par suite d'une erreur dans TEX ou METAFONT). Je ne sais pas dans quels cas les utilisateurs seront réellement gênés, car la plupart de ces exceptions sont assez ésotériques.

— TEX substituait aux chaînes de caractères `^^0`, `^^1`, ..., `^^9`, `^^a`, `^^b`, `^^c`, `^^d`, `^^e`, `^^f` respectivement les caractères `p`, `q`, ..., `y`, `!`, `"`, `#`, `$`, `%`, `&`. Il ne le fera plus désormais si le caractère suivant est l'un des caractères `0123456789abcdef`.

— TEX n'insérait aucun caractère à la fin d'une ligne en entrée quand la valeur de `\endlinechar` était supérieure à 127. Il en insèrera dorénavant un, sauf si `\endlinechar` est supérieur à 255. Comme avant, `\endlinechar<0` supprime le caractère *fin-de-ligne* qui est normalement le caractère *ASCII 13*, *Control-M* ou *retour-charriot*.

— Quelques messages de diagnostic de TEX utilisaient la notation `["80]` ... `["FF]` pour faire référence aux caractères *128* ... *255* (par exemple pour afficher le contenu d'une boîte en débordement impliquant des polices contenant ces caractères). On utilise maintenant la notation `^^80` ... `^^ff`.

— Les expressions `char128` et `char0` étaient considérées comme équivalentes dans METAFONT ; maintenant, `char` est défini *modulo 256*. En conséquence, `char-1` est égal à `char255`, *etc.*

— **INITEX** oubliait tous les exemples de césure précédents à chaque nouvelle spécification de `\patterns`. Maintenant, toutes les spécifications de `\patterns` sont cumulatives et vous ne pouvez pas utiliser `\patterns` dès l'instant où **INITEX** a mis en place les césures dans un paragraphe.

— TEX avait l'habitude de fonctionner un peu différemment quand vous essayiez de mettre en page des caractères absents d'une police. Un caractère absent est maintenant considéré comme étant une limite de mot, ce qui aura pour effet de vous donner un peu plus d'explications quand `\tracingcommands` sera positif.

— TEX et METAFONT fourniront des statistiques différentes en fin de compilation car ils ont maintenant un nombre différent de primitives.

— Les programmes qui utilisaient le dispositif de traitement de chaînes de caractères de **TANGLE** ne pourront plus fonctionner sans modifications car la nouvelle version de **TANGLE** commence la numérotation des chaînes à plusieurs caractères à partir de *256* au lieu de *128*.

— Pour pouvoir fonctionner dans les mêmes conditions que précédemment, **INITEX** doit maintenant imposer les valeurs `\lefthyphenmin=2` et `\righthyphenmin=3`.