Cahiers GUTenberg

QUELQUES RÉFLEXIONS SUR LE TRAITEMENT DE TEXTES INFORMATISÉS À PROPOS DU LOGICIEL TEX

François CHAHUNEAU

Cahiers GUTenberg, nº 1 (1989), p. 2-15.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1989___1_2_0>

Quelques réflexions sur le traitement de textes informatisé à propos du logiciel TEX

François CHAHUNEAU

A.I.S., 34, avenue du Roule, 92200 NEUILLY sur SEINE

1. Le contexte historique

1.1. L'âge de plomb ...

Dans les débuts de l'informatique, c'est à dire dans les années 50, on ne pouvait guère envisager d'utiliser les ordinateurs pour manipuler des textes : d'abord, les machines étaient rares; ensuite, on avait déjà suffisamment de mal à saisir des programmes sur des perforatrices de cartes pour ne pas avoir l'idée d'y saisir d'autres textes que ceux des commentaires des programmes; enfin, de toutes façons, le résultat obtenu en sortie sur des *imprimantes à impact* — en majuscules et sans accents! — aurait été extrêmement décevant...

1.2. Apparition des formateurs de texte

Dans les années 70, le développement parallèle des systèmes *temps partagé* et des terminaux imprimants (*"teletypes"*), l'adoption du code ASCII sur 7 bits qui rendait possible l'utilisation des caractères minuscules, et l'apparition des éditeurs de texte en mode ligne firent évoluer rapidement cette situation.

Les premiers éditeurs de texte en mode ligne (du type d'*edit*, l'éditeur du système TSO/MVS sur la série IBM 370) étaient fort peu puissants[1]. A la suite d'idées qui se trouvaient dans l'air à cette époque au MIT et dans les laboratoires de recherche de Bell Telephone sur ce que devrait être un système temps partagé moderne[2], on vit apparaître des éditeurs ligne beaucoup plus puissants (*TECO*, *QED*), dont les programmes *qedx*, *ted* sur *Multics* ou *ed* sur *UNIX* sont les descendants directs.

Les mêmes personnes sont à l'origine des premiers *formateurs de texte* tels que *Roff* dont les descendants directs — conservant la même philosophie — furent *Runoff* puis *Compose* sur *Multics* ou *Nroff* et *Troff* sur *UNIX*. Pour utiliser ces formateurs de texte, on mélange dans un *texte source* le texte *sensu stricto* et des commandes plus ou moins cryptiques au programme de formatage. La saisie du texte est censée être totalement indépendante du périphérique visé pour la sortie imprimée, la commande du périphérique étant à la charge d'un *pilote* spécialisé. Ceci permet de s'adapter à l'évolution rapide des technologies : les versions actuelles de ces programmes comportent des pilotes pour imprimantes matricielles (100 à 250 points/pouce), électrostatiques ou à laser (200 à 400 points/pouce), ainsi que pour photocomposeuses (jusqu'à 5000 points/pouce).

L'évolution des technologies s'orientant de plus en plus vers des périphériques à pilotage de type "graphique" (c.à.d. : où le caractère est dessiné en *bitmap*), il est devenu envisageable d'utiliser ces forma-

[1] le plus étonnant est qu'EDIT soit toujours le seul éditeur en mode ligne disponible en standard sur les systèmes temps partagé IBM! ...

[2] C'est ce mouvement d'idées qui fut à l'origine du système *Multics*, puis d'*UNIX*, après qu'une partie de l'équipe ait fait scission...

teurs de texte pour composer des textes techniques ou mathématiques. *Eqn* est une extension de *Troff* permettant la saisie des formules mathématiques sous forme symbolique linéarisée; *Tbl* est une autre extension facilitant la saisie des tableaux.

TEX peut être considéré comme l'ultime maillon de cette lignée évolutive de formateurs de texte; on verra brièvement comment il se situe par rapport aux autres logiciels comparables.

1.3. Apparition des éditeurs pleine page

Dans les années 80, la vulgarisation des terminaux à écran vidéo a impliqué le développement d'éditeurs pleine page. Les premiers ne concernaient que des terminaux utilisés en transmission synchrone (principalement chez IBM). L'apparition de terminaux disposant de fonctions locales évoluées (insertion et supression de lignes et de caractères) et l'augmentation des débits de communication asynchrone rendirent possible la généralisation des éditeurs pleine page en mode asynchrone (du type *Emacs* sur *MULTICS*).

Du coup, la saisie et la correction des textes se trouva grandement facilitée et l'ordinateur devint un outil privilégié pour la manipulation de textes. Certaines fonctions dites de "traitement de texte" (centrage de lignes, reformatage de paragraphes...) furent intégrées aux éditeurs les plus perfectionnés, permettant d'obtenir à l'écran une mise en page voisine de celle de la version imprimée.

Toutefois, les caractères transmissibles ou affichables étant en nombre réduit (128 caractères ASCII) et la saisie se faisant sur un *clavier "pauvre"*, il n'était pas envisageable d'afficher à l'écran autre chose que du texte sans accents[3] et dans une seule police de caractères. Tous les problèmes de changement de police, de génération d'indices, d'exposants et de caractères spéciaux (symboles mathématiques...), de soulignements et de tracé de boîtes étaient à la charge du formateur de texte et de ses pilotes. Ces effets particuliers étaient obtenus en incluant dans le texte source des commandes symboliques, d'où un certain degré d'"abstraction".

1.4. Le "boom" de la micro-informatique et la tendance Wysiwyg

Sur un micro-ordinateur, il n'y a pas de problème de vitesse de transmission entre l'unité centrale et l'écran. L'édition pleine page fut donc, dès le début, la règle. L'utilisation des *attributs vidéo* reconnus par le contrôleur d'affichage (intensité lumineuse, couleur, soulignement, clignotement, vidéo inverse) permet d'enrichir nettement les modes d'affichage possibles par rapport à un terminal. De plus, puisqu'on peut accéder directement à la mémoire d'écran (bitmap), il est possible, moyennant un certain degré de complexité dans la conception de l'éditeur, d'afficher à la fois sur l'écran des caractères *préformés* (c.à.d. : se trouvant en ROM) et des caractères *dessinés*, c'est à dire en fait de mélanger du texte et du graphique.

En ce qui concerne les claviers, il n'existe aucune norme concernant le nombre et la disposition des touches (sauf la norme *de facto* imposée par l'IBM PC), ce qui permet d'envisager des *claviers "riches"*, comportant des touches surnuméraires par rapport aux claviers ASCII. Le programme d'édition ne reçoit pas du clavier des codes ASCII, comme dans le cas

[3] sauf si l'on décide d'afficher des lettres accentuées à la place de certains caractères

d'un terminal, mais sait seulement quelle touche a été enfoncée, ce qui permet de déclencher n'importe quelle opération lors de l'enfoncement d'une touche — selon le *mode* dans lequel se trouve temporairement le programme — y compris l'affichage d'un caractère dessiné.

Dans l'environnement bureautique, on assiste depuis plusieurs années à une compétition entre les *machines de traitement de texte* dédiées (qui succèdent, selon une évolution naturelle, à une gamme de machines à écrire électroniques de plus en plus perfectionnées) et les micros-ordinateurs "universels" munis de logiciels de traitement de texte puissants. Pour que les fabricants de micro-ordinateurs et les développeurs de logiciel gagnent cette guerre, ils doivent convaincre le personnel de bureau que le micro-ordinateur, muni du logiciel adéquat, peut maintenant fort bien *remplacer* la machine à écrire ou la machine de traitement de texte. Ils doivent donc proposer des produits qui déroutent le moins possible, et qui émulent de très près, dans leur manipulation, les matériels traditionnels.

Tous ces facteurs ont favorisé le développement de la tendance "WYSIWYG"[4] dans la conception des traitements de texte sur micros. Les logiciels doivent permettre d'obtenir à l'écran la meilleure approximation possible de la sortie imprimée, les seules différences avec une machine à écrire électronique étant la hauteur de l'écran (25 lignes), la possibilité de rechercher des séquences et d'effectuer facilement des corrections[5], et la capacité de stockage.

S'il est certain que, pour taper des lettres, ces caractéristiques sont hautement souhaitables, cela est beaucoup moins évident lorsqu'il s'agit de documents à structure complexe (rapports divisés en paragraphes et chapitres, avec notes infrapaginales, titres courants...), et encore moins lorsqu'il s'agit de préparer des documents directement imprimables en respectant certaines conventions typographiques.

Avec un système WYSIWYG, si l'on décide que, finalement, tous les mots que l'on avait soulignés devront apparaître en italique, il faudra probablement reprendre les mots un par un. De même si l'on décide que les débuts de paragraphes doivent être *indentés* de 3 caractères et non de 5, ou que les titres de paragraphes d'un certain niveau doivent être en caractère gras et non soulignés. Chacune de ces conventions de présentation doit être "concrétisée" manuellement lors de la frappe du texte, et l'on doit faire attention à être cohérent d'un bout à l'autre du texte. Enfin, si les paragraphes sont numérotés et que l'on décide d'en déplacer ou d'en insérer un... toute la numérotation est à refaire. La situation est claire : en cherchant à émuler fidèlement une machine à écrire perfectionnée, les logiciels de type WYSIWYG s'interdisent par définition de faire beaucoup mieux. En encourageant l'utilisateur à conserver des habitudes de dactylographie, ils limitent la possibilité d'accès à la plupart des finesses typographiques. En bannissant toute abstraction de l'interaction avec le programme, ils privent l'utilisateur des automatismes souhaitables dans un véritable *système de production de documents*, comme la prise en charge de la cohérence de présentation typographique du texte, selon certains paramètres aisément modifiables, ou comme l'automatisa-

[4] "What You See Is What You Get"

[5] Il est d'ailleurs remarquable de constater que, chez les utilsateurs non avertis de la micro-informatique, il existe une confusion totale entre la notion d'*éditeur de texte* et celle de *"traitement de texte"*.

tion de la numérotation des paragraphes, des références bibliographiques...

Pour reprendre une phrase de B. Kernighan, la philosophie "What you see is what you get" a comme corollaire : "What you see is *all* you get".

2. Tour d'horizon rapide des formateurs de texte

2.1. La famille "roff"

Dérivés du programme *Runoff* du MIT, *Nroff* et *Troff* ont été développés dans les laboratoires de la Bell Telephone vers 1975. Il s'agit des formateurs de texte disponibles en standard sous *UNIX*. *Nroff* est la version pour imprimantes traditionnelles (à marguerite ou à aiguilles), c'est à dire ne sachant pas *dessiner* de caractères, et où les déplacements micrométriques sont impossibles. *Troff* est la version pour imprimantes à laser ou photocomposeuses. Du point de vue de l'utilisation, ces deux logiciels sont très voisins et l'on peut aisément créer un fichier source compréhensible par les deux programmes.

Les préprocesseurs *Eqn* (qui existe dans les deux versions *Troff* et *Nroff*) pour la saisie de formules mathématiques, *Tbl* pour la saisie de tableaux, et divers jeux de macros viennent compléter cet ensemble, qui est très bien intégré à la collection d'*outils de préparation de documents* sur *UNIX*. Ce groupe d'outils comprend les éditeurs de textes et les divers "filtres" qui manipulent des textes, mais aussi les processeurs *pic*, *ideal* pour l'insertion de figures géométriques, *grap* pour l'insertion de graphiques, et *refer* pour la gestion automatisée des citations et références bibliographiques. Tous ces outils s'interfacent de manière très naturelle grâce aux possibilités d'*UNIX* [Kern. 84]. La version actuelle de *Troff* améliore l'indépendance vis-à-vis du périphérique d'impression et est connue sous le nom de *DITroff* (Device Independant Troff).

Les habitués de *Troff* prétendent que la saisie des formules mathématiques est légèrement plus naturelle dans *Eqn* que dans le mode mathématique de TEX, qui impose une certaine rigueur. On verra cependant que *Troff* est nettement moins sophistiqué que TEX dans ses fondements algorithmiques, notamment en ce qui concerne la construction des paragraphes.

Groff , conçu à l'Ecole des Mines de Saint-Etienne[6] appartient à la même famille, et accepte les fichiers *Troff* et *Nroff*, tout en étant mieux adapté à la langue française.

2.2. SCRIBE

Le logiciel *scribe*, conçu en 1980 par B.K. Reid de Carnegie-Mellon University est un autre formateur de texte, susceptible de traiter des textes structurés, mais n'acceptant pas les formules mathématiques. Je ne dispose d'aucune documentation sur ce produit, mais il ne figure pas au catalogue **SM90**.

2.3. TEX et MINT

TEX a été écrit par D.E. Knuth de l'université de Stanford. Knuth est, entre autres, l'auteur d'un ouvrage encyclopédique : *The Art of Computer Programming*, et sa compétence en matière d'algorithmique n'est pas à mettre en doute. TEX a été conçu dès le départ en collaboration avec des professionnels de l'édition, de manière à incoroporer le maximum du savoir-faire traditionnel des typographes. L'*American Mathematical Society* — dont Knuth est

[6] commercialisé aujourd'hui par la société SYNC

membre — fut l'un des principaux commanditaires de TEX , et a fortement encouragé son développement puis, par la suite, favorisé son adoption par la communauté scientifique universitaire.

La première version de TEX date de 1978; écrite en langage *sail* — peu répandu — elle est restée à l'état de prototype et s'est trouvée confinée à Stanford. La première version portable, écrite en *pascal*, date de 1980. La version actuellement disponible est TEX82; elle n'est pas encore considérée comme la version définitive. Cette version a été portée sur de très nombreux systèmes, y compris des "postes de travail" utilisant diverses version d' *UNIX* (Sun, SM90), et les compatibles IBM-PC sous *MS-DOS*.

Mint a été développé par Peter Hibbard à Carnegie-Mellon University (première version disponible en 1984). Hibbard est un ancien élève de Knuth qui connait bien TEX, et de nombreux concepts et algorithmes sont communs à TEX et à *mint*. *Mint* se veut plus simple d'utilisation que TEX , tout en étant moins puissant. *mint* prend d'avantage d'initiatives que TEX en ce qui concerne la typographie, ce qui le rapproche un peu d'un *système expert en typographie*[7]. De ce fait, les possibilités sont moindres, pour l'utilisateur, de contrôler les détails de la typographie. Le style du document (lettre, rapport, note de service...) est défini par une référence, en début de fichier, à un jeu de macros faisant partie d'une bibliothèque. Ce mode de fonctionnement, possible avec TEX, devient la règle dans *mint*. Enfin, *mint* inclut en standard des possibilités d'insertion de figures et d'illustrations. *mint* est actuellement commercialisé sur SM90 (et VAX) par la société TANGRAM.

[7] TEX ne prend de telles initiatives qu'en mode mathématique

3. Les points forts de TEX

3.1. Coût, portabilité, diffusion...

TEX est considéré comme un produit "universitaire" du domaine public, c'est à dire que le *source* du programme est distribué, et ce, moyennant une somme symbolique pour les laboratoires universitaires ou assimilés.

TEX est écrit en *Pascal*, ce qui garantit sa portabilité au prix d'une certaine perte d'efficacité. En particulier, l'absence d'entrées-sorties en accès direct dans de nombreuses implantations de it Pascal (ou plutôt l'absence de norme à ce sujet) a amené Knuth à se passer de cette possibilité, ce qui a pour conséquence une structure artificiellement compliquée des fichiers DVI (Device Independent) générés par TEX. Pour certains, comme B. Kernighan, l'un des défauts majeurs de TEX est précisément d'être écrit en *Pascal* : "*Pascal* is a toy language, suitable for teaching but not for real programming" [Kern. 81]. Signalons qu'il existe maintenant des "traductions" de TEX en langage C, mais il est difficile de savoir s'il s'agit de traductions "mot-à-mot" ou de réelles adaptations...

Néanmoins, ces deux caractéristiques de TEX ont favorisé fortement sa diffusion, tant en milieu universitaire — où de nombreux développements autour de TEX on été réalisés (notamment : écriture de jeux de macros ou de nouveaux pilotes) — qu'en milieu industriel, où l'on commence à utiliser TEX comme noyau dans des systèmes intégrés de traitement de texte (cf. plus loin : UNITEXT).

TEX a été adopté dès le départ comme "norme" par l'AMS qui avait en grande partie initié son développement. Divers journaux scientifiques aux USA, notamment en mathématiques, suivent le mou-

vement et commencent à accepter des fichiers DVI à la place de manuscrits. Il est possible pour ces journaux de fournir leurs "instructions aux auteurs" sous forme de "formats TEX", c'est à dire de jeux d'instructions et/ou de macros spécifiant les caractéristiques typographiques souhaitées. On peut penser que cette tendance risque de se généraliser, mais reste à savoir si les Européens ont les mêmes raisons d'accepter TEX comme norme. En 1985, TEX était installé dans 137 universités dont 74 aux USA et dans 141 sociétés dont 112 aux USA. Il a été diffusé dans plus de 12 pays [Auzou. 84].

Enfin, signalons l'existence du TEX Users Group (TUG) qui est très actif et publie trois frois par an la revue TUGboat.

3.2. TEX, l'"anti-Wysiwyg"

Avant tout, il est nécessaire de comprendre que dans TEX , la notion de "nombre de caractères par ligne" n'existe pas. Les caractères, selon leur police et leur taille, ont certes un "encombrement" minimal fixe, mais l'espace blanc qui les sépare se mesure en unités *continues* (cm, mm, *inches* ou points typographiques). Ce "blanc" est contractible et extensible dans certaines limites : D.E. Knuth le désigne par le terme de "glu". C'est dans ces unités qu'on spécifie l'*indentation* des paragraphes, l'interligne, les sauts verticaux, la largeur et la hauteur des pages... Par exemple, on utilisera en tête du source TEX la commande : "`\parskip 0.5cm plus 1mm minus 1mm`" indiquant que l'espace blanc vertical entre deux paragraphes doit être de "0.5 cm en principe, d'au plus 0.6 cm et d'au moins 0.4 cm".

Cette seule caractéristique montre bien déjà que TEX ne cherche absolument pas à imiter une machine à écrire, mais plutôt une photocomposeuse... Ceci explique aussi pourquoi les pilotes TEX utilisent nécessairement les périphériques de sortie en mode *graphique*, de manière à pouvoir positionner un caractère n'importe où dans la page.

Dans TEX, l'unité minimale de mise en page est le *paragraphe*. Une fin de paragraphe est repérée soit par une (ou plusieurs) ligne blanche, soit par la commande `\par`. A part cela, tous le blancs entrés dans le texte source lors de la saisie sont ignorés par TEX, qu'ils se situent dans la ligne, en début de ligne, ou entre des paragraphes (une ligne vide est équivalente à plusieurs lignes vides). Ce comportement, diamétralement opposé à la philosophie WYSIWYG, surprend au début. On s'y habitue cependant assez vite.

3.3. Traitement des paragraphes et des pages

A la différence de *tous* les formateurs de texte actuellement connus (sauf ceux qui dérivent de TEX), c'est au niveau du *paragraphe* que TEX optimise la répartition des mots entre les lignes. Pour chaque répartition possible, TEX calcule un *critère* de la qualité de présentation, basée sur la mesure des contraintes (compression ou extension) subies par la "glu", et sur le nombre et le type des coupures de mots. Littéralement, on peut dire que TEX refusera d'adopter une disposition des mots dans un paragraphe si elle est trop "tirée par les cheveux". Ce mode de fonctionnement *global* est *a priori* fort coûteux en temps de calcul puisque, si un paragraphe contient n points de coupure possible, il existe 2^n façons de diviser le paragraphe en lignes. C'est là où toute la science algorithmique de Knuth intervient : en utilisant un algorithme de programmation dynamique, il ramène ce problème à l'ordre n^2, puis à n

en adaptant l'algorithme au cas particulier de la coupure d'un paragraphe en lignes. Ce mode de travail, évidemment, est également gourmand en place mémoire, surtout si les paragraphes sont longs...

Dans *Troff* comme dans tous les autres formateurs de texte — ainsi, par définition, que dans tous les traitements de texte WYSIWYG —, ce problème est résolu *localement*, ligne par ligne. Ceci signifie que l'on pourra avoir une ligne contenant beaucoup de "blanc" entre les mots, suivie d'une ligne en contenant très peu, ou bien plusieurs lignes de suite comportant des coupures de mots, ce qu'évitent les typographes professionnels.

Pour la coupure en pages, TEX se limite à une approche *locale* car, pour des raisons d'encombrement mémoire, il n'est évidemment pas envisageable de mémoriser tous les pargraphes constituant un chapitre (comme on mémorise tous les mots constituant un paragraphe). TEX évite cependant de "mal" couper un paragraphe entre deux pages, et ceci de manière automatique.

C'est principalement à cause de ce mode de fonctionnement[8] — qui fait la puissance et l'orginalité de TEX — qu'il est assez difficile d'envisager une implémentation de TEX dans laquelle le résultat de la composition serait affiché *en temps réel* sur un écran *bitmap*, comme dans un traitement de texte WYSIWYG. TEX ne sait comment il va composer un paragraphe que lorsque celui-ci est clos, et ne sait comment il va composer une page que lorsque tous les paragraphes susceptibles d'y tenir sont clos. Ainsi, même si l'on envisageait d'intégrer TEX dans un ensemble interactif, comprenant un éditeur qui permette de modifier le source au vu du résultat de la composition, l'unité de travail minimale serait la *page*.

Le projet VORTEX, à l'université de Berkeley, a pour but le développement d'une version interactive de TEX. Dans cette version, le texte source et le texte composé apparaissent simultanément sur un écran *bitmap*. Il est possible d'effectuer interactivement des modifications sur le texte source comme sur le texte composé. Toute modification apportée à l'une des "représentations" provoque une mise à jour automatique de l'autre représentation. Un prototype[9] est prévu pour 1986.

3.4. Typographie mathématique

C'est essentiellement dans ce domaine que TEX révèle son expertise en matière de typographie, limitant beaucoup de ce fait les risques d'erreur de la part de l'utilisateur. Il existe, dans la tradition typographique, des règles très strictes régissant la largeur de d'espace blanc de part et d'autre d'un signe "=", d'un signe "+",... Que l'on saisisse `y=ax+b` ou `y = ax + b`, TEX générera dans les deux cas : $y = ax + b$.

3.5. Indépendance vis-à-vis du périphérique de sortie

Le degré d'indépendance des formateurs de texte vis-à-vis des périphériques de sortie se manifeste à deux niveaux, qui sont d'ailleurs liés : la *commande des déplacements dans la page* et la *sélection des polices de caractères.*

Déplacements : Dans tous les formateurs de texte permettant de définir des déplacements arbitraires, ces déplacements s'expriment de manière in-

[8] remarquons que ce mode de fonctionnement se rapproche beaucoup de la méthode de travail d'un typographe professionnel

[9] revue TUGboat, 6(3) : 138

terne dans une unité élémentaire discrète : on se déplace de n unités, n étant un nombre entier. Pour *Nroff*, cette unité est le $240^{\text{ème}}$ de pouce; pour *Troff*, le $432^{\text{ème}}$ de pouce. Dans TEX, l'unité élémentaire est le "*rsu*"[10], qui vaut $5.36\ 10^{-3}\mu$ (n est donc nécessairement codé sur 32 bits). Le choix d'une unité si petite est, selon Knuth, indispensable pour éliminer tout risque d'erreur d'arrondi lors de l'accumulation des déplacements (ce problème ne se pose bien sûr que pour les photocomposeuses à très haute résolution).

Dans *Troff* ou *Nroff*, c'est le formateur de texte lui-même qui transforme ces déplacements en unités caractéristiques de la résolution du périphérique, en effectuant un arrondi. Même dans la version DITroff de Troff, les déplacements dans la page sont exprimés dans le fichier de sortie en "unités périphérique", et Troff doit donc connaître la résolution du périphérique au moment où il traite le texte source.

Dans TEX au contraire, les déplacements restent exprimés en *rsu* dans le fichier DVI, et le calcul d'arrondi est effectué par les pilotes.

Ainsi, TEX apparaît actuellement comme le seul formateur de texte pour photocomposeuses et imprimantes à laser dont les fichiers de sortie (DVI) soient réellement indépendants du périphérique.

Polices de caractères : En ce qui concerne les polices de caractères, *Nroff*, *Groff* ou *Troff* utilisent les polices *localement disponibles* sur le périphérique d'impression. Au contraire, TEX utilise un jeu de polices[11] *défini de manière interne* et donc indépendant du périphérique. Même sur les périphériques les plus perfectionnés (type imprimante à laser ou photocomposeuse), TEX utilise ses propres polices qui sont alors téléchargées.

On distingue donc deux conceptions fondamentalement différentes :

- Dans les logiciels de la famille "*roff*", l'idée est d'utiliser au mieux les possibilités locales de chaque périphérique afin de gagner en efficacité. L'aspect de la sortie imprimée dépendra donc assez fortement de ces possibilités (par exemple, des substitutions de polices sont effectuées), mais la quantité d'information à transmettre au périphérique est pratiquement réduite aux codes ASCII des caractères.

- Dans TEX, l'idée est de s'affranchir au maximum
des spécificités et des limitations du périphérique : entre une sortie sur imprimante matricielle d'une résolution de 240 points/pouce et une sortie sur photocomposeuse d'une résolution de 3000 points/pouce, la seule différence sera la finesse du dessin des caractères. TEX ne peut donc être utilisé que sur des périphériques où l'on peut transférer des *bitmaps* et son utilisation est assez coûteuse, tant en transmission qu'en temps de calcul.

3.6. Ecriture de macros

Comme la plupart des formateurs de texte évolués, TEX permet la définition de macro-instructions paramétrables. Dans TEX, les moyens mis à la disposition de

[10] "ridiculously small unit"

[11] plus de 300 en standard, si l'on prend en compte les différentes tailles

l'utilisateur expert pour écrire des macros sont ici particulièrement puissants. Ce mécanisme est très utilisé pour l'adaptation (*customization*) du produit en vue d'une application déterminée.

4. Les critiques

4.1. Critiques générales

Je ne sais plus quel spécialiste du traitement textes a dit, à propos de TEX : "*on peut aussi apprendre le japonais*", mais cette phrase résume bien l'impression générale de la plupart des gens qui se trouvent pour la première fois face à un texte source TEX, surtout si l'auteur du texte a choisi d'utiliser TEX au plus bas niveau, c'est à dire sans l'aide de macros. Cet aspect cryptique des commandes ne devrait cependant pas décourager les utilisateurs potentiels car il est possible, comme on le verra, d'en limiter fortement l'usage.

Toutefois, TEX reste avant tout un *langage de programmation* avec toutes les contraintes que cela comporte, notamment une syntaxe très rigide. TEX manque fréquemment d'"intelligence" ou d'"intuition" en ce qui concerne la compréhension des intentions de l'utilisateur. L'oubli d'une accolade fermante (}) est une erreur des plus fréquentes, et peut avoir des conséquences catastrophiques lorsqu'il n'est pas diagnostiqué par TEX.

De manière plus générale, on peut regretter que TEX nécessite, de la part de l'utilisateur, une certaine connaissance de ses mécanismes de fonctionnement interne. Par exemple, si l'on veut pouvoir formater sans problème des lignes courtes justifiées à gauche et à droite, il faut augmenter le seuil de tolérance concernant les contraintes subies par la "glu" et/ou la fréquence des césures de mots. Là encore, les subtilités de ce genre peuvent en grande partie être prises en charge par un *jeu de macros* associé au *style* du document, préalablement préparé par un spécialiste[12] de TEX.

4.2. Coupures de mots en langue française

Cette critique ne concerne d'ailleurs malheureusement pas seulement TEX, mais plutôt tous les programmes de traitement de texte conçus aux USA. On peut même dire que TEX s'en tire beaucoup mieux que ses concurrents, car la coupure de mots dans TEX se fait selon un algorithme, avec recherche de suffixe, préfixe et étude du radical. Le *dictionnaire* ne sert ici qu'à définir les *exceptions*, c'est à dire les mots auxquels les règles standard ne peuvent pas s'appliquer (à la différence de nombreux programmes de traitement de texte où le dictionnaire sert à définir *toutes* les coupures possibles). Ce dictionnaire peut être facilement enrichi par l'utilisateur — pour la durée d'un texte particulier ou de manière plus générale en regroupant les exceptions dans un fichier — par l'utilisation de la commande `\hyphenation`.

Les régles définies par défaut dans TEX (ou plutôt dans le fichier `HYPHEN.TEX` associé au format par défaut `PLAIN.TEX`) ne s'appliquent pas à la langue française : les préfixes et les suffixes courants ne sont pas les mêmes, les règles de coupures à l'intérieur du radical non plus). Ce problème est "facilement" soluble : il suffit de réécrire un fichier `HYPHEN.TEX` adapté à la langue française et il existe déjà plusieurs versions de ce fichier (dont celle que j'utilise, réalisée par N. Brouard de l'INED).

Mais un problème subsiste : d'une part, l'algorithme ne trouvera pas toutes les

[12] ce que Knuth appelle un "TEXpert"

coupures de mots possibles; d'autre part, la coupure des mots en français est régie beaucoup moins par un ensemble de règles cohérentes que par toute une série d'habitudes et d'exceptions [Buckle 84]. *Or on ne peut pas définir une exception (c.à.d. : préciser les coupures possibles) pour un mot qui contient des caractères accentués!* Ceci provient du fait que la génération des caractères accentués, dans TEX, se fait par une séquence de contrôle (qui, en fait, provoque une superposition) : il ne s'agit donc pas de caractères comme les autres...

La solution de ce problème passe par une modification du programme TEX, ce dont s'occupe activement l'équipe du Laboratoire de Typographie Informatique de l'Université Louis Pasteur à Strasbourg.

En attendant, on peut cependant se demander si le degré d'expertise de TEX en ce qui concerne les césures de mots en français n'est pas déjà largement supérieur à celui de la plupart d'entre nous, et même à celui d'un typographe professionnel après une longue journée de travail. Knuth, dans le TEXbook donne quelques exemples de "perles" relevées dans les colonnes des journaux américains...

4.3. Insertion de figures

Il n'existe dans TEX aucun mécanisme standard permettant l'insertion de graphiques ou d'illustrations dans le texte. Il est bien sûr possible de réserver de la place (un rectangle de dimension et position quelconques), et ceci de manière assez souple (insertions "flottantes"). Il existe aussi la possibilité de redéfinir la macro `\output` qui contrôle le stockage dans le fichier DVI de la page composée, et d'insérer une marque qui sera repérée par le pilote. Ce pilote peut alors prendre toute initiative appropriée, comme provoquer la combinaison du *bitmap* correspondant à la page formatée avec un *bitmap* correspondant à une figure (créé par un logiciel graphique).

C'est cette approche qui a été utilisée au CIRCE pour combiner, dans la chaîne de traitement *Sapristi*, les logiciels TEX, GPGS, et *Uniras*. Les pilotes commercialisés pour certaines imprimantes à laser qui utilisent le code intermédiaire *PostScript* permettent le même genre de choses; c'est le cas des pilotes pour *Apple Laser Writer*, *QMS 800/2400*, *Sun Laser Writer*, *Appolo Domain Laser26*...

Ces solutions restent cependant dépendantes du périphérique, et la plupart des utilisateurs de TEX souhaitent une extension du programme permettant l'insertion de graphiques, comme c'est le cas pour *mint*.

4.4. Critiques injustifiées...

Hormis ces trois points, la plupart des autres critiques formulées à l'encontre de TEX (niveau d'abstraction, manque d'interactivité,...) sont le fait des partisans de l'approche WYSIWYG, qui n'ont pas compris ou ne veulent pas comprendre la supériorité de TEX par rapport à un éditeur pleine page amélioré (ces critiques pourraient d'ailleurs s'appliquer à tous les formateurs de texte évolués). A ceux-ci, que répondre, si ce n'est qu'"on a rien sans rien"...

5. Possibilités d'amélioration des interactions avec l'utilisateur

Diverses approches complémentaires sont envisageables pour faciliter la mise en œuvre de TEX, dans l'optique d'une utili-

sation systématique pour le traitement de textes mathématiques ou non.

5.1. Les jeux de macros

TEX reconnaît 300 commandes "natives", auxquelles s'ajoutent les 600 macros du jeu de macros par défaut `PLAIN.TEX`, écrit par Knuth. Il existe actuellement plusieurs jeux de macros plublics dont les plus connus sont :

- $\mathcal{AMS}$TEX : c'est un ensemble de macros qui facilitent la saisie des formules mathématiques. Réalisé par Michael Spivak en 1982, $\mathcal{AMS}$TEX est publié par l'American Mathematical Society. Ce jeu de macros est compatible avec le format `PLAIN.TEX`.

- LATEX : c'est un important jeu de macros, écrit par Leslie Lamport en 1984, qui transforme TEX en un véritable "système de préparation de documents". Il s'agit de faciliter la préparation des documents scientifiques longs (thèses, rapports). LATEX permet, entre autres :

  - la structuration du texte en chapitres, sections, paragraphes de divers niveaux avec numérotation automatique et génération de tables des matières;
  - la gestion des citations et références bibliographiques (un module complémentaire, BIBTEX, permet d'aller chercher automatiquement les références dans une base de données);
  - la construction de tableaux, la numérotation automatique des tableaux et figures, la création d'une liste des figures et tableaux;
  - le *formatage* d'un texte sur plusieurs colonnes;
  - la génération d'index.

  LATEX n'est pas compatible avec le format `PLAIN.TEX`. Il s'agit donc d'une version "spéciale" de TEX, très enrichie par rapport à la version de base. Hormis la taille mémoire nécessaire pour charger l'ensemble des macros (qui interdit, par exemple, l'utilisation de LATEX dans la version 512 K de TEX sur PC *MS-DOS*), et quelques "*bugs*" qui subsistent dans les dernières versions, il n'y a aucun inconvénient à utiliser systématiquement LATEX au lieu de TEX. C'est le choix fait par de nombreux utilisateurs de TEX au CIRCE.

  SLITEX est un jeu de macros associé à LATEX qui permet la création de transparents pour rétroprojection.

- HPTEX : c'est un ensemble de macros créé chez Hewlett-Packard en 1984 pour faciliter l'utilisation de TEX sur la série de méga-micros HP 200. Moins riche (mais moins encombrant!) que LATEX, il en reprend les caractéristiques les plus couramment utilisées. HPTEX est compatible avec `PLAIN.TEX`.

5.2. Les préprocesseurs de TEX

Lorsqu'on saisit un texte en français, il est très pénible de devoir taper `\'e`, par exemple, pour coder un "é", en l'attente d'une version francisée de TEX. De même, il est fastidieux de saisir la séquence `{\bf maigre}` pour saisir le mot "**maigre**" en caractères gras. L'idée principale dans la réalisation d'un *préprocesseur* de TEX est de rendre le texte source moins "verbeux" et plus lisible.

N'importe quel traitement de texte perfectionné sur micro peut constituer un tel préprocesseur. Ainsi, à partir du programme PC-WRITE et du module de sortie associé[13], j'ai réalisé très rapidement un préprocesseur simple de TEX qui permet au moins la saisie normale des caractères accentués sur un PC, et qui génère automatiquement les commandes de changement de police, de soulignement, d'encadrement...Un mot en caractères gras apparaît effectivement en caractères gras à l'écran, de même pour un mot souligné; un mot encadré apparaît en vidéo inverse. Ces "attributs" sont codés, dans le *buffer* de l'éditeur, par des caractères de contrôle, qui sont transcodés en commandes TEX par le module de sortie.

Au niveau au dessus, le préprocesseur *stratec* réalisé à l'Université de Strasbourg[14] offre le même type de possibilités, et profite en plus d'une caractéristique particulière du *Victor Sirius* qui permet d'afficher des caractères définis par l'utilisateur. Les lettres grecques et divers caractères spéciaux sont saisis en utilisant les touches surnuméraires du clavier ou des combinaisons de touches, et sont visualisés directement à l'écran.

Enfin, le logiciel *Mathor*[15] permet la saisie directe et l'affichage à l'écran des formules mathématiques, avec transcodage ultérieur en commandes TEX. *Mathor* était conçu avant tout au départ comme un logiciel indépendant, possédant ses propres pilotes, et non comme un préprocesseur de TEX. Cependant, les concepts de mise en page des formules mathématiques dans *Mathor* étant voisins de ceux utilisés dans TEX (boîtes enboîtées), il fut possible de réaliser une interface, qui décuple l'intérêt de *Mathor*. Actuellement, l'interface se limite à traduire de manière assez "figée" les codes *Mathor* en commandes TEX, de manière à obtenir après traitement par TEX une présentation aussi proche que possible de celle obtenue directement par *Mathor*. Ainsi, le source TEX généré contient des commandes du type `\vskip 8mm` ou `\noindent\hskip 12mm` en tête de chaque paragraphe. Il s'agit donc d'une "réduction" de TEX aux possibilités de *Mathor*, qui ne laisse à l'utilisateur que très peu de possibilités d'enrichir le source TEX généré (à part définir un facteur de "magnification" globale).

Cette approche ne pose pas de problème pour le texte mathématique, mais paraît insuffisante en ce qui concerne le texte "normal", et surtout semble assez illusoire puisque, de toutes façons, TEX redistribuera les mots entre les lignes d'un paragraphe et les coupera à sa façon. Les paragraphes n'auront donc pas le même nombre de lignes, et ne seront pas distribués de la même manière entre les pages. Il serait donc souhaitable d'"assouplir" cette interface en remplaçant les commandes de déplacement rigides par des appels de macros (du genre `\debutpar` au début de chaque paragraphe). Ces macros pourraient être définies par défaut par *Mathor* en tête du fichier (de manière à rester compatible avec la version actuelle), mais elles pourraient être facilement modifiées par l'utilisateur de manière a modifier globalement le style de mise en page du document.

5.3. La pré-visualisation à l'écran

C'est, à mon avis, cette possibilité qui apporte le plus de confort dans l'environ-

[13] ce programme, distribué par la société Quicksoft à Seattle est en *freeware*, c'est à dire que sa diffusion entre les utilisateurs est encouragée

[14] Laboratoire de Typographie Informatique

[15] société Novedit - versions VICTOR S1 et compatibles IBM-PC; peut-être prochainement SM90

nement de TEX. Il est toujours difficile, surtout pour les néophytes, de savoir exactement quelle mise en page finale sera provoquée par une série de commandes. Or la sortie sur imprimante (même à laser) du résultat final reste un processus assez long (de l'ordre de la minute par page). La prévisualisation sur écran *bitmap* haute résolution permet l'affichage d'une page composée en quelques secondes, ce qui améliore grandement l'interactivité. Des programmes de pré-visualisation existent pour les versions de TEX sur IBM-PC et sur postes de travail *UNIX*. Dans ce dernier cas, le système de multi-fenêtrage permet l'affichage simultané du source TEX et du résultat composé.

Notons cependant que la résolution d'un écran *bitmap* dit "haute-résolution" est de l'ordre de 100 points/pouce, ce qui est insuffisant pour afficher de manière lisible une page en format A4 à sa taille réelle. Il est donc nécessaire de "magnifier" la page lors de l'affichage, et seule une partie est visible a l'écran.

Références bibliographiques

[Kern. 84] B.W. KERNIGHAN, *The Unix Document Preparation Tools — A Retrospective* in "PROTEXT 1 : Proceedings of the First International Conference on Text Processing Systems", pp. 12–25, Dublin, Ireland, 24–26 Octobre 1984, J.H. MILLER, ed, Boole Press Pub., Dublin.

[Kern. 81] B.W. KERNIGHAN, *Why Pascal is not my favourite programming language* in Science Technical Report n ° 100, Bell Laboratories, 1981.

[Auzou. 84] A. AUZOU-CONNES, *Etude de TEX et* METAFONT, Diplôme d'Ingénieur, CNAM, Paris, 1984.

[Buckle 84] N. BUCKLE, *Word Hyphenation in French*, in "PROTEXT 1 : Proceedings of the First International Conference on Text Processing Systems", pp. 108–113, Dublin, Ireland, 24–26 Octobre 1984, J.H. MILLER, ed, Boole Press Pub., Dublin.

Pour des informations complémentaires, on pourra consulter les références suivantes, non citées dans le texte :

[FR 85] D. FOATA, Y. ROY, *TEX/SM 90 et le langage WEB. Manuel d'utilisation* Publ. I.R.M.A. Strasbourg 258/D-07 et 261/D-10, Laboratoire de Typographie Informatique, Université Louis Pasteur, Strasbourg, 1985.

[André 82] J. ANDRÉ, *Bibliographie analytique sur les "manipulations de texte"* in T.S.I. vol. 1, n ° 5, sept.-octb. 82, Dunod ed.
Bibliographie très complète, mais déjà un peu périmée...

[Knuth 85] D.E. KNUTH, *The TEXbook*, Addison-Wesley, 1985.
Tous les détails sur l'utilisation de TEX. Assez illisible, mais très complet...

[Knuth 84] D.E. KNUTH, *TEX : The Program*, Addison-Wesley, 1984.
Description des algorithmes utilisés dans TEX.

[Roy 84] Y. ROY, *TEX/WEB et le traitement de textes mathématiques.*
Thèse décrivant les principales caractéristiques de TEX et ses diffi-

cultés d'implantation sur UNIVAC 1110*; description du langage WEB*

Conclusion en forme de commentaire

Cet article de François Chahuneau a été écrit en avril 1986; nous avons décidé de le publier car il fait une présentation assez complète des différents types de logiciels de traitement de textes et il situe bien TEX par rapport à l'ensemble des autres produits.

La destination première de ce papier était d'aider à la réflexion en vue de mettre en place une politique cohérente en matière de logiciel de traitement de texte dans le schéma directeur de l'INRA. L'orientation matériel était à l'époque la SM90, mise au point par le CNET, cela explique le nombre de références à cette machine.

Pour répondre aux critiques, *"on peut aussi apprendre le japonais"*, c'est plutôt TEX qui a appris le japonnais car il existe maintenant une version de TEX qui traite des textes en japonnais! Mais sur le problème qui nous concerne plus particulièrement, les coupures de mots en français, la version multilangue de TEX, MLTEX, sait parfaitement gérer les césures de mots français, accentués ou non, il est même possible de mélanger dans un même texte, par exemple un résumé en anglais, avec des césures anglaises, et le texte en français, avec des césures françaises. MLTEX a été développé au Canada (ils étaient bien placés pour le faire) par M. Ferguson de l'INRS.

Pour ce qui est de l'enrobage de TEX, l'article de Laurent Siebenmann dans ce même numéro fait une bonne mise au point sur les préprocesseurs; il existe enfin maintenant des logiciels de prévisualisation à l'écran pour une large gamme de matériels, ce point n'est donc plus un problème.

O. Nicole

PUBLICITÉ

Vient de paraître

- Jacques ANDRÉ, Richard FURUTA & Vincent QUINT
  Structured Documents
  The Cambridge Series on Electronic Publishing, Cambridge University Press, Cambridge 1989, 220 pages. ISSBN 0 521 36554 6.
  [Cet ouvrage correspond au cours organisé par l'INRIA en 1987 à Aussois et fait la synthèse sur la question : concepts, modèles et systèmes de documents ; normes ; représentation, reconnaissance, archivage, transmission de documents structurés ; aspects typographiques et linguistiques ; bibliographie de plus de 250 titres.]