Cahiers GUTenberg

LaTeX DANS LES ANNÉES 90

Frank Mittelbach, Rainer Schöpf

Cahiers GUTenberg, nº 6 (1990), p. 2-14.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1990___6_2_0>

LaTeX dans les années 90*

Frank MITTELBACH† et Dr. Rainer SCHÖPF‡

† *Electronic Data Systems (Deutschland) GmbH, Eisenstraße 56, D-6090 Rüsselsheim,* PZF5HZ@DRUEDS2.BITNET

‡ *Institut für Theoretische Physik der Universität Heidelberg, Philosophenweg 16, D-6900 Heidelberg, RFA,* BK4@DHDURZ1.BITNET

Résumé Au cours des trois dernières années, LaTeX s'est largement répandu, y compris dans de nouveaux domaines comme les applications commerciales. Avec l'apparition de nouvelles catégories d'utilisateurs, il faut reconsidérer l'implémentation de LaTeX et certaines de ses fonctionnalités. Dans quelques années, LaTeX 2.09 seul ne suffira plus à satisfaire les besoins croissants de ses utilisateurs. L'un des aspects importants de LaTeX — l'échange de documents — risque d'être sacrifié sur l'autel des changements et améliorations spécifiques à un site donné.

A partir de ces considérations et de notre expérience de plusieurs années de maintenance, nous proposons une réimplémentation de LaTeX. Cette nouvelle version préservera non seulement les interfaces utilisateur actuelles (pour rester compatible avec les anciens fichiers LaTeX), mais tiendra aussi compte des demandes d'extensions déjà formulées, comme des développements futurs.

Abstract *During the last three years, LaTeX has spread widely, even into such new fields as business applications. The fact that there are new classes of users forces one to reconsider the LaTeX implementation and some of its features. Within a few years, LaTeX 2.09 alone will not be sufficient to satisfy the increasing needs of its users. As a consequence one of the important characteristics of the LaTeX concept — the possibility of exchanging documents — is in danger of being sacrificed on the altar of local changes and enhancements.*

Starting from these considerations and from our experiences of several years of LaTeX support, we will present a proposal for a re-implementation of LaTeX. This new version would not only preserve the essential features of the present user interface (in order to be compatible with old LaTeX files), but also take into account already formulated requests, as well as future developments.

*Cet article est la traduction française, réalisée par Éric Picheral (CICB Rennes) et Philippe Louarn (IRISA/INRIA-Rennes), de l'article présenté au dixième congrès du TUG à Stanford en août 1989 et au congrès EuroTeX'89 à Karlsruhe (RFA) en septembre 1989. Cet articles ([MS89b]) est paru dans *TUGboat* vol. 10, (n° 4) p. 681–690, 1989. Cet article est reproduit ici avec l'autorisation des auteurs.

1. Introduction

Depuis quelques années, LaTeX est largement utilisé comme outil de composition de documents. Ses avantages par rapport aux systèmes WYSIWYG et à *plain* TeX — conception logique et commandes de haut niveau pour une édition formatée — permettent, même à une personne sans expérience en composition (l'auteur), de produire facilement des documents bien lisibles. En particulier, pour les ouvrages collectifs, la notion de structure logique permet d'assurer l'homogénéité d'un document.

LaTeX est un langage de marquage qui permet, du moins en théorie, à l'utilisateur de saisir un texte en termes d'entités logiques, par exemple spécifier une partie de texte comme étant une « citation » ou une « liste étiquetée » au lieu de fournir des commandes liées à la structure physique telles que « renfoncer les trois lignes suivantes et les faire commencer par un boulet ».

La traduction en commandes de formatage, par exemple en primitives TeX, est (encore une fois, au moins en principe) bannie du fichier source. Elle sera au contraire effectuée, sans que l'usager s'en rende compte, dans l'un des fichiers de style qui, appliqués au même source, produiront des mises en page différentes. LaTeX offre quatre styles prototypes différents (`article`, `report`, `book`

et `letter`) qui doivent être utilisés pour différents types de fichiers-sources, donc ne devraient pas être interchangeables. Cette contradiction (on ne peut pas passer du style `article` au style `report` par exemple) est le premier malentendu (parmi beaucoup d'autres) pour les utilisateurs de LaTeX et les concepteurs amateurs de style.

Puisqu'il y a un seul style de document officiellement maintenu pour chaque type de document, les utilisateurs de TeX ayant l'expérience de *plain* TeX se sentent souvent incapables de produire la mise en page souhaitée avec LaTeX. Aussi, après quelques tentatives infructueuses, reviennent-ils à *plain* TeX, même s'ils ont alors à faire face à d'autres problèmes (tels que les références croisées, etc.) qui sont facilement résolus avec un langage de marquage comme LaTeX.

Dans ce cas, le cœur du problème réside dans le fait que la plupart des interfaces de LaTeX sont très peu documentées. Même des experts en TeX peuvent donc avoir du mal à produire un certain style, par exemple remplacer le style standard `report`. Du coup, presque tous les documents LaTeX se ressemblent parce que tous les styles de document existants sont peu originaux (variantes de styles prototypes, sans différences notables); ou alors ils sont inutilisables (à cause de nombreuses bogues).

2. Interface utilisateur — ou les caractéristiques essentielles de LaTeX

Pour chaque type de document, LaTeX fournit un ensemble de commandes de formatage de haut niveau définies à l'aide de macros internes. Ce code interne est hautement modulaire, souvent ingénieux[1] et il permet une grande diversité de mise en pages à condition que le concepteur de style soit suffisamment familier avec les interfaces.

Vu de l'extérieur, LaTeX a une conception élaborée : à situations similaires, formes syntaxiques similaires, les cas inhabituels étant réglés par l'emploi d'arguments optionnels ... Mais ce n'est pas tout. En tant que standard dans toutes les applications (mais avec différentes mises en page), LaTeX propose :

- la génération automatique de tables des matières, liste de figures, etc.;
- un mécanisme puissant de références croisées avec noms symboliques. Ceci permet d'insérer, supprimer et déplacer des blocs de texte sans avoir à modifier le numéro des équations, etc.;
- la possibilité de ne composer que certaines parties d'un document sans perdre les références croisées, la valeur des compteurs, etc.;
- un concept général d'objets flottants permettant de placer automatiquement les figures, tableaux, etc. indépendamment les uns des autres (mais en préservant la position relative dans chaque classe d'objets);
- avec BibTeX et MakeIndex, il offre des outils puissants pour la création automatique d'index et d'entrées bibliographiques[2];

[1] Il est évident que LaTeX n'est pas implémenté tout à fait correctement. Mais le concept d'ensemble est sagement choisi et n'est pas remis en cause par des bogues de conception ou d'implémentation dans les modules.

[2] Sur un site ayant une installation à jour de LaTeX, ces deux programmes doivent être disponibles. Malheureusement, ce n'est pas toujours le cas.

- des commandes qui permettent de changer la taille des caractères;
- un mécanisme général pour changer la mise en page (en-têtes courants, etc.);
- ...

Alors pourquoi ne pas utiliser LaTeX?

3. Limites de LaTeX v.2.09

Voici quelques limites de la version actuelle de LaTeX. Nous les classons en plusieurs groupes et donnons des exemples au fur et à mesure.

3.1. Erreurs d'implémentation

C'est ici que la plupart de nos exemples se situent. Mais, en quelque sorte, c'est aussi le groupe de problèmes qui est le plus facile à résoudre : il suffit d'éviter les erreurs faites par L. Lamport mais d'utiliser toutes ses bonnes idées.

3.1.1. Commandes « fragiles »

Le concept le plus gênant de cette catégorie est peut-être la notion de commandes *fragiles* et *robustes*. En bas de la page 23, le manuel LaTeX déclare :

« L'argument d'une commande de sectionnement peut être utilisé pour autre chose que simplement produire le titre de la section; il permet de générer une entrée pour la table des matières et un en-tête en haut de la page. [...] Quand il est déplacé de l'endroit où il apparaît dans le fichier d'entrée vers les autres emplacements où il est utilisé, l'argument d'une commande de sectionnement est un peu maltraité. Certaines commandes LaTeX sont fragiles et peuvent se casser quand elles apparaissent dans un argument qui est maltraité de cette façon. Les commandes fragiles sont rarement utilisées dans l'argument d'une commande de sectionnement. [...] Dans les rares occasions où vous devez mettre une commande fragile dans le titre d'une section, vous la protégez simplement à l'aide d'une commande `\protect`. »

Et plus loin (p. 24) :

« Un argument dans lequel des commandes fragiles ont besoin de `\protect` est appelé un argument *mobile*. Les commandes qui ne sont pas fragiles sont appelées *robustes*. Pour chaque commande [...], j'indiquerai si elle est robuste ou fragile. Sauf dans des cas spéciaux [...] une commande `\protect` ne peut être nuisible, aussi est-il presque toujours prudent d'en utiliser une; même si vous n'êtes pas sûr que ce soit nécessaire. »

Simple, n'est-ce-pas? Hélas non, car une commande brisée produira un message d'erreur totalement incompréhensible et, pour achever le tout, non seulement cette erreur pourra se produire à un endroit différent, mais encore ne disparaîtra t'elle pas forcément quand le `\protect` manquant sera finalement trouvé. C'est l'enfer pour les utilisateurs novices de LaTeX, mais même des experts sont parfois perplexes s'ils commettent cette erreur.

La commande `\protect` est souvent utilisée pour supprimer les expansions de macros non désirées et qui causent les erreurs mentionnées plus haut. Voilà la véritable erreur de conception : il serait préférable de supprimer toute expansion par défaut et que l'utilisateur soit autorisé à développer les seules macros pour lesquelles c'est nécessaire.

3.1.2. Récupération d'erreur par LaTeX

Cette partie de l'implémentation peut être résumée en une seule phrase : il n'y a pas de récupération d'erreur. En réalité, il existe des cas où une procédure d'erreur est déclenchée, mais le mécanisme de récupération n'est pas très puissant. En soi, cela ne pose aucun problème car on peut adopter la philosophie « comprendre l'erreur, corriger le source et réexécuter », mais, malheureusement, les messages d'erreur ne sont pas très clair :

```
You're in trouble here. Try typing
<return> to proceed.
If that doesn't work, type X <return>
```

```
to quit.
```

Bref, on est certain qu'il y a des problèmes quand on voit une page entière de messages d'erreur issue du « ventre » de TeX[3] et le conseil d'éteindre l'ordinateur et de rentrer chez soi : ce n'est pas vraiment ce que l'utilisateur attend quand il tape "H" (aide interactive) après une erreur signalée par LaTeX.

Les messages d'erreur sont même parfois ni plus ni moins qu'inexacts ; les lignes suivantes :

```
\begin{center}
␣
\end{center}
```

produisent par exemple le message d'erreur :

```
Something's wrong--perhaps a missing
\item
```

Dans neuf cas sur dix, cette erreur n'est pas causée par un `\item` manquant ; l'utilisateur ne sait donc pas quoi corriger (en fait, l'environnement `center` s'attend à trouver du texte, c'est-à-dire quelque chose en mode horizontal et non pas une ligne blanche). Il peut être utile de consulter le manuel, car les messages d'erreur y sont expliqués de façon plus détaillée. Mais les messages d'erreur produits par LaTeX restent quand même incompréhensibles par l'utilisateur.

3.1.3. L'environnement générique de liste

L'environnement générique de liste est un des modules centraux de LaTeX. Il est utilisé de façon interne par la plupart des environnements standard fournis par LaTeX ; même des environnements comme `center` sont traités comme des cas particuliers de liste (avec une commande `\item` vide).

Pour autoriser une telle variété d'applications, l'environnement de liste possède une vingtaine de paramètres qui peuvent être modifiés pour changer la mise en page. De plus, les valeurs par défaut de ces paramètres (définies dans le fichier de style de document) dépendent du niveau d'imbrication ; c'est-à-dire que le style de document peut offrir différents espacements par défaut pour les listes emboîtées dans d'autres listes.

Mais il reste quelques problèmes d'implémentation :

- Une première erreur de conception est la décision d'ajouter la valeur de `\parskip` à tous les paramètres d'espacement vertical, même quand cet espacement est utilisé à des endroits où aucun paragraphe ne se termine.

  La modification de ce paramètre agit donc sur la mise en page à des endroits inattendus ce qui implique que d'autres paramètres qui ne devraient pas être modifiés, doivent être ajustés pour compenser cet effet de bord indésirable.

- Par ailleurs, le paramètre `\parskip` est accessible à l'utilisateur alors que les paramètres affectés peuvent être modifiés uniquement dans le fichier de style. L'utilisateur peut changer, par exemple, le paramètre `\topsep`, mais sa valeur par défaut, définie dans le fichier de style, devra être restaurée ultérieurement.

- La restauration des paramètres à leurs valeurs par défaut (si rien

[3] La production d'horribles listes d'erreurs est effectivement un problème de TeX qui devrait être catalogué comme « limitation TeXnique ». Un produit comme LaTeX, implémenté comme un grand ensemble de macros, doit avoir de nombreux niveaux d'expansion et il n'est pas possible de supprimer la partie intermédiaire de la pile historique quand TeX repère une erreur. A notre avis, il faudrait que TeX ait une commande `\tracing...` pour contrôler la sortie de la pile historique. Don Knuth a accepté cette proposition pour TeX 3.0 : cette nouvelle primitive s'appelle `\errorcontextlines`.

d'autre n'est spécifié) est quelque peu arbitraire. Certains des paramètres importants (par exemple les valeurs de pénalité pour les coupures de page avant et après une liste) héritent leur valeur de la liste englobante ce qui est plus qu'un inconvénient pour un concepteur de style[4].

- Un autre choix de l'implémentation empêche de définir des listes avec vignettes de la largeur de la page, par exemple des vignettes placées seules sur la ligne. Ces listes ne peuvent pas être emboîtées correctement (voir par exemple les commentaires d'un article sur l'implémentation de l'environnement théorème étendu [Mit89c]).

Comme LaTeX est utilisé dans des domaines de plus en plus nombreux, un environnement de liste plus général devient indispensable.

3.2. Limites de conception

Il est certainement impossible de tracer une frontière nette entre les désastres d'implémentation et les limites de la conception. Les premiers sont des problèmes introduits quand les macros ont été écrites, les secondes viennent de décisions ou d'omissions pendant la phase de conception. Néanmoins, ces deux phases vont souvent ensemble.

3.2.1. Sélection de polices

Il est certainement impossible de tracer une frontière nette entre les désastres d'implémentation et les limites de la conception. Les premiers sont des problèmes introduits quand les macros ont été écrites, les dernières viennent de décisions ou d'omissions pendant la phase de conception. Néanmoins, ces deux phases vont souvent ensemble.

3.2.2. Sélection de polices

Une de ces limitations vient de la décision de Leslie Lamport de reprendre de *plain* TeX la méthode de sélection des différents styles de polices : il a disposé les polices dans une grille à deux dimensions, l'une spécifiant la taille, l'autre le style. C'était raisonnable au moment de l'implémentation, car il y avait seulement quelques styles différents utilisables avec TeX. Depuis pourtant, de plus en plus de polices sont devenues disponibles, rendant inadéquat le choix ci-dessus. Par exemple, l'utilisateur ne peut pas, sans toucher au style, passer de façon simple des polices de Knuth aux polices résidentes d'une imprimante PostScript à l'intérieur d'un même document.

3.2.3. L'environnement de liste

Malgré sa généralité, l'environnement de liste présente certaines faiblesses de conception :

- Il est impossible de générer une liste *run-in*, c'est-à-dire une liste où le premier élément s'accroche au texte précédent. Cette fonctionnalité existe dans $\mathcal{A}\mathcal{M}\mathcal{S}$-TeX ;
- plus généralement, puisque la présentation des types de liste standard est fixée par le style de document choisi, il n'y a aucun moyen pour l'utilisateur de choisir différentes sortes de présentation pour le même type de liste (par exemple liste énumérée, liste simple) sans définir ses propres environnements. Il serait préférable

[4] Une conséquence désagréable est que l'option de style de LaTeX `fleqn` (qui permet d'appuyer à gauche les équations au lieu de les centrer comme c'est l'habitude) favorise les coupures de page juste avant les équations *hors-texte* (cette bogue est corrigée dans la version de LaTeX du 24 mai 1989).

d'offrir un moyen de spécifier des attributs tels que *compact*, *stream* (voir par exemple [Won85]), *run-in*, etc.;

- l'espace vertical avant et après les listes est contrôlé par le même paramètre;
- l'espace vertical précédant le premier élément ne dépend pas de la longueur de la dernière ligne du paragraphe précédent (comme dans le cas des équations hors-texte).

3.2.4. Concept d'attribut

LaTeX ne permet pas à l'utilisateur de spécifier des attributs comme dans Script-DCF par exemple. Pourtant ce concept vaut la peine d'être examiné, au moins pour une prise en compte partielle dans LaTeX (voir la sous-section 3.2.3).

3.2.5. Arguments produisant du texte

Le mécanisme de TeX qui balaye les arguments des macros fixe les `\catcodes` des éléments qu'il lit. En conséquence, certaines commandes LaTeX (telles que `\verb`) ne peuvent pas être utilisées à l'intérieur des `\parboxes`, `\footnotes`, etc. On peut éviter cela en utilisant un schéma plus élaboré; voir par exemple l'implémentation de `\footnote` dans *plain* TeX [Knu86, p. 363] ou l'article sur les *mottos* de chapitre [Won86].

3.2.6. Les mathématiques

En plus des possibilités offertes par *plain* TeX, LaTeX ne propose que les environnements `eqnarray` et `array`. Pour composer des textes de mathématiques avec LaTeX, ce n'est certainement pas suffisant. $\mathcal{AMS}$-TeX offre les fonctionnalités nécessaires; pourtant leur insertion dans LaTeX n'a pas encore été réalisée[5]. Au lieu de cela, différentes personnes ont écrit des options de style non publiées, chacune implémentant des sous-ensembles limités.

Par ailleurs, lors du congrès de Karlsruhe, Barbara Beeton a annoncé que Michael Spivak préparait une nouvelle version d'$\mathcal{AMS}$-TeX, comprenant la plupart des caractéristiques de LaTeX[6].

3.2.7. `array` *et* `tabular`

L'environnement `tabular` de LaTeX permet de composer des alignements sophistiqués. Cela signifie que l'on a pas besoin de maîtriser parfaitement TeX pour « savoir comment faire des tableaux avec filets » [Knu86, p. 245]. Par contre, plusieurs possibilités faciles à implémenter ne sont pas disponibles. Voir par exemple la nouvelle implémentation décrite dans [Mit88, Mit89a].

3.2.8. Images

Pour dessiner, LaTeX offre surtout des commandes élémentaires de positionnement qui devraient en fait être utilisées de façon interne pour définir des interfaces de haut niveau. Des exemples de telles interfaces sont donnés dans [Pod86, Sch89].

3.2.9. La procédure de sortie (output routine)

La procédure de sortie est un module très complexe et sophistiqué qui offre une large variété de possibilités pour composer une page. Pourtant, certaines décisions de mise en page (placement des notes de bas de page et des objets flottants) ne peuvent

[5] Cette situation a maintenant évolué; l'*American Mathematical Society* va distribuer un fichier de style $\mathcal{AMS}$-TeX qui sera une option utilisable avec n'importe quel fichier de style LaTeX. Il y aura aussi des versions de `book.sty` et `article.sty` qui implémenteront des styles livre et revue définis par l'AMS. Cette information a été aimablement fournie par Regina Girouard, responsable des services de composition de l'AMS.

[6] Cette version, baptisée L-$\mathcal{AMS}$-TeX, est actuellement commercialisée par *The TeXplorators Corporation*, Houston, Texas. ndt.

être implémentées dans les fichiers de style sans une redéfinition des macros internes de LaTeX. Ceci pose des problèmes de compatibilité. Voir aussi la section 4.

3.3. Interfaces inconnues

Comme nous l'avons déjà signalé, de nombreuses interfaces ne sont pas documentées proprement. Cela entraîne des solutions inutilement compliquées pour certains problèmes de mise en page. Pire encore : on aboutit parfois à la conclusion qu'il n'y a pas de solution du tout.

De tels problèmes peuvent facilement être résolus en utilisant les macros internes de LaTeX de façon correcte. Considérez par exemple une mise en page décalée où tous les titres de section doivent s'étendre dans la marge gauche qui est elle-même assez large pour les contenir. Ceci peut être réalisé avec la commande générique de sectionnement `\@startsection` qui permet de spécifier un paramètre de renfoncement. Si on lui donne une valeur négative, on obtient des titres de section qui débordent à gauche.

3.4. Limitations TeXniques

Dans ce groupe, nous citons des limites du programme TeX lui-même. La limite sans doute la plus grave provient du mécanisme d'insertion de TeX. Après avoir prématurément émis une page (de sorte que la procédure de sortie puisse examiner son contenu), le matériel inséré est enlevé de la liste verticale principale et rassemblée dans certaines boîtes. De ce fait, il n'est plus possible de réinsérer de façon inchangée le contenu de la page[7]. Cela fait qu'il est presque impossible de balancer la hauteur de plusieurs colonnes quand des insertions comme les notes de bas de page ou les objets flottants sont en cause [Mit89d]. D'autres problèmes TeXniques proviennent de la façon dont TeX insère les pénalités, ou découpe les paragraphes en lignes.

[7] Cette proposition a été prise en compte par Donald Knuth, qui l'a implémentée dans TeX 3.0.

4. Nouvelles demandes

Quand on leur demande leur avis, beaucoup d'utilisateurs de LaTeX répondent :

> « LaTeX est quelque chose de magnifique, mais il manque ci ou ça. »

Si nous éliminons les demandes qui sont réglées par la lecture du manuel, il reste trois groupes de souhaits :

- Les fonctionnalités qui peuvent être implémentées dans la version actuelle de LaTeX. Se pose ici le problème du trop petit nombre de personnes qui connaissent la structure interne et les interfaces de façon suffisante pour faire le travail. Ces interfaces sont très chichement documentées et de mauvaises interprétations ou utilisations peuvent conduire à des résultats catastrophiques. Même si le travail est fait correctement, le problème de la compatibilité des différentes options de style demeure.
- Les fonctionnalités pour lesquelles il manque les interfaces indispensables. Il faut ici changer les macros internes de LaTeX pour les implémenter.
- Les demandes entièrement nouvelles qui conduisent à des modifications de grande ampleur du code ou même des concepts. Pire encore : certaines choses ne peuvent absolument pas être implémentées car TeX ne peut pas les traiter.

Plutôt que de faire la liste des nombreuses demandes, nous donnons maintenant quelques exemples.

4.1. Fonctionnalités faciles à implémenter

Nous avons déjà invoqué le problème des titres débordant sur la marge gauche. D'autres demandes peuvent facilement être réglées dans un fichier de style, comme les commandes `\pagestyle` spéciales (comme celles qu'utilise Leslie Lamport dans son livre), les conventions de numérotation (par exemple les équations à l'intérieur des théorèmes ...) et les mises en page spéciales pour les titres (par exemple des titres centrés, ou bien `\chapter` sans le mot « Chapter »).

Une question très importante est la prise en compte des langues nationales. LaTeX standard ne propose rien dans ce domaine. Pourtant, on peut facilement l'implémenter par une option de style comme le montre le fichier `german.sty` qui constitue le standard allemand auquel tous les sites de langue allemande adhèrent depuis 1987. Voir [Par88] pour une description de ses fonctionnalités.

Certaines demandes ressortent du domaine des applications commerciales; par exemple le besoin de marquer chaque page d'un document avec la date, l'heure, le nom du fichier d'entrée et éventuellement des renseignements de sécurité.

4.2. Fonctionnalités qu'on peut implémenter avec un effort modéré

Comme exemple de ce deuxième groupe, considérons la nouvelle implémentation des environnements `array` et `tabular` décrite en [Mit88, Mit89a]. En plus de la possibilité de créer de beaux tableaux avec filets, cette nouvelle implémentation permet à l'utilisateur de spécifier l'apparence d'une colonne à un endroit précis.

On peut aussi considérer le défi lancé par David F. Rogers dans [Rog88]. Il demandait une macro pour placer une figure, qui remplisse au moins trois conditions parmi cinq, et affirme que « les commandes LaTeX d'insertion d'objets flottants ne fonctionnent pas non plus ». Ceci n'est que partiellement vrai. Deux conditions sont automatiquement remplies dans LaTeX standard; celle qui reste (les figures numérotées doivent être insérées après la première référence à la figure) peut facilement être implémentée en changeant seulement une ligne de code dans *une* macro interne de la procédure de sortie de LaTeX; dans la macro `\@addtocurcol`, l'appel à `\addtotoporbot` doit être changé en un appel à `\@addtobot`. Évidemment, une option de style qui implémente cette modification doit prendre également garde aux paramètres flottants car leur affectation par défaut tend à décourager le flottement par le bas.

Comme autre exemple, on peut prendre une composition en deux colonnes où les notes de bas de page apparaissent en bas de la colonne de droite. De nouveau, ceci peut être résolu en redéfinissant *seulement* la macro interne `\@makecol`.

4.2.1. Prise en compte des imprimantes PostScript

« Nous devons reconnaître l'importance de PostScript comme standard de facto. [...]. Nous devons nous rendre compte que la compatibilité avec PostScript est cruciale si nous voulons être pris au sérieux par le reste du monde »[Cla89, p. 153]. La question de la conversion du contenu des fichiers `.DVI` en PostScript a déjà été prise en considération, mais ici nous nous inté-

ressons au problème de l'incorporation des polices PostScript. Il est facile de modifier `lfonts.tex` pour utiliser les polices internes des imprimantes PostScript au lieu de celles définies par Knuth. Pourtant cette modification doit être fait à la création du fichier `.fmt` (instruction `\dump`) et l'utilisateur ne peut donc pas la mettre en œuvre. Il faudrait un schéma de sélection de police pendant l'exécution de TEX, comme nous l'avons décrit dans [MS89a][8].

4.3. Problèmes difficiles

Maintenant que nous avons vu que la composition de pages complexes peut facilement être traitée par LATEX, on peut se demander quelles demandes (à notre avis) appartiennent au troisième groupe. Et bien, la composition de page par exemple; des demandes vraiment très complexes telles que la mise en page utilisée par la revue *Scientific American* : trois colonnes, les figures chevauchant le texte sur une à trois colonnes avec des légendes placées dans la colonne voisine si nécessaire. Comme cela a déjà été mentionné, la primitive `\insert` de TEX ne peut pas être utilisée pour un tel travail. *Tout* faire à la main est possible (TEX est une machine de Turing, comme Leslie Lamport l'a noté) mais il y a des limitations dans l'espace et le temps disponibles[9].

Comme nous l'avons déjà noté, nous pensons que le balancement des colonnes appartient au troisième groupe si des notes de bas de page sont en jeu, en dépit du fait que le *TEXbook* [Knu86, p. 417] dit implicitement que ceci est possible. Nous serions seulement trop heureux de voir un algorithme qui ne faillisse pas dans des conditions normales (c'est-à-dire sans restrictions).

[8] L'interface de ce schémas de sélection avex LATEX a réalisé depuis la rédaction de cet article. Il est décrit dans [MS90]

[9] Néanmoins, nous travaillons à ce projet à Mainz, en espérant prouver que ce problème appartient au deuxième groupe.

5. Propositions pour une nouvelle implémentation

La version courante de LATEX est composée essentiellement des fichiers suivants :

`lplain.tex` — fonctionnalités de *plain* TEX utilisées par LATEX

`lfonts.tex` — définitions de polices

`hyphen.tex` — motifs de césure

`latex.tex` — l'essentiel des fonctionnalités de LATEX ou les macros internes pour les construire dans un fichier de style

`*.sty` — fichiers de style de document et options du style de document

Les quatre premiers sont tous chargés quand un fichier de format est généré. Cela justifie qu'une grande quantité de la mémoire interne de TEX est utilisée pour stocker les définitions contenues dans ces fichiers, même s'il y a seulement de très rares occasions où elles sont utilisées toutes ensembles. Si nous envisageons d'ajouter de plus en plus de fonctionnalités à LATEX, nous sommes conduits à nous demander : quelles parties de LATEX sont essentielles pour la plupart des types de document (appelées le « noyau ») et lesquelles sont seulement utilisées par des applications spéciales (appelées « la partie périphérique »).

5.1. Le noyau de LATEX

Avant de pouvoir parler de récrire le noyau, nous devons le séparer de la partie périphérique. Les modules suivants sont considérés comme faisant partie du noyau :

- Fonctionnalités de base issues de *plain* TeX (définies dans `lplain.tex`)
- Sélection de police
- Constantes utilisées par le reste du programme
- Structure de contrôle du programme
- Concept de semi-paramètre
- Procédures de base pour le traitement des erreurs
- Espacement, coupure de ligne et de page
- Traitement des fichiers
- Gestion des compteurs
- Commandes mathématiques de base
- Environnement générique de liste et ses applications de base
- Commandes de construction de boîte (y compris `parbox` et `minipage`)
- `array` et `tabular`
- Interface pour l'environnement `théorème`
- Commandes génériques de sectionnement
- Interface de génération des tables de matières, liste de figures, etc.
- Bibliographie
- Objets flottants
- Notes de bas de page
- Procédure de sortie (*output routine*)

Certaines de ces parties ont besoin d'être revisées, d'autres doivent être entièrement ré-implémentées : l'environnement de liste, l'écriture dans les fichiers `.aux` pour supprimer la fonctionnalité `\protect` (déjà fait), le mécanisme des compteurs (fait), les références hiérarchiques, les commandes `array` et `tabular` (fait), ou encore une procédure de sortie nouvelle quant à sa conception (c'est certainement la partie la plus difficile). Toutes les parties ci-dessus devraient être rendues plus modulaires pour obtenir un plus haut degré de flexibilité.

Pour fournir un meilleur contrôle au moyen des styles ou des extensions, plusieurs *hooks*, tels que `\everypar`, devront être introduits : on peut penser à `\everylist`, `\everysection`, `\everybegindocument`, `\everyenddocument`, etc.

5.2. Fonctionnalités périphériques

Les parties périphériques ne devraient pas être incluses dans le fichier format. Elles peuvent facilement être déplacées dans plusieurs fichiers qui sont chargés à la demande durant l'exécution de LaTeX.

Nous considérons que les parties suivantes sont périphériques :

- Mathématiques de plus haut niveau. Les seules fonctionnalités actuellement disponibles sont `eqnarray` et `array`. Une nouvelle implémentation devrait fournir les mêmes fonctionnalités que $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TeX, chacune étant chargée séparément.
- Un nouvel environnement `\verbatim` avec une taille illimitée de texte `verbatim` et une commande pour entrer un texte `verbatim` depuis un fichier (fait).
- Un environnement `picture` amélioré (fait sur le plan des concepts, partiellement implémenté : epic, PiCTeX)
- Des améliorations devraient être discutées, par exemple l'empilement ou dépilement de plusieurs environnements `tabbing`.
- La prise en compte d'options spéciales pour les épreuves, montrant les étiquettes symboliques, les entrées d'index là où elles apparaissent, l'impression de l'heure et de la date, etc.
- Index : plus précisément, l'interface avec un programme d'index tel que MakeIndex.

Comme nous l'avons mentionné précédemment, une installation de LaTeX est complète seulement si elle comprend BibTeX et MakeIndex. L'intégration de ces programmes, spécialement de MakeIndex, c'est-à-dire les interfaces, devrait être améliorée.

5.3. Styles de document

5.3.1. Un concept de marquage standard (SGML)

Pour une ré-implémentation de LaTeX, il faut aussi reconsidérer les styles de document standard. Comme nous l'avons exposé dans l'introduction, il y a différents types de documents, par exemple les

livres, manuels, articles, rapports, lettres, compte-rendus, etc. Ces types ne peuvent pas être échangés puisque chacun a ses propres concepts logiques : les lettres n'ont pas de chapitre tandis qu'il n'y a pas besoin d'entête de lettre ou de signature dans un livre. Étant donné un type de document (par exemple un rapport), il y a de nombreuses façons différentes de faire le formatage. Le style **report** de LaTeX courant est ainsi une instance parmi beaucoup d'autres du meta-style **rapport**.

Le point important, ici, est que tous les styles de type rapport doivent utiliser les mêmes concepts logiques de telle sorte qu'un document LaTeX écrit proprement soit indépendant de l'instance spécifique de style utilisée. Un site allemand (disons l'université de Mainz) peut décider d'offrir un style spécial (appelé par exemple **uni-mainz-report**) pour formater des rapports suivant ses propres conventions. Mais un rapport écrit à Stanford peut alors être composé à Mainz en utilisant ce style *sans* modifier le fichier d'entrée LaTeX — mais évidemment le document sortira avec des espacements différents, etc.

Ainsi, une des tâches importantes d'une nouvelle implémentation consiste à reconsidérer les concepts logiques utilisés jusque là, puis à décider de supprimer ou modifier l'un d'entre eux, ou d'en ajouter de nouveaux. Par exemple, il n'y a actuellement aucune différence entre les styles prototypes **report** et **book**. Évidemment ceci ne peut être considéré comme normal. C'est bien sûr une question à débattre entre experts.

5.3.2. Prise en compte des langues nationales dans un contexte linguistiquement indépendant

Il y a ici un point à noter : on ne doit pas induire de ce que nous avons dit plus haut que la représentation textuelle des en-têtes (par exemple « Contents », « Litteratura ») doive être fixée par le style. A cet égard, nous ne sommes pas d'accord avec Leslie Lamport. Au contraire, puisqu'il se peut fort bien qu'un site désire composer des documents écrits dans différentes langues, mais tous dans le même style, il est raisonnable de fournir un moyen pour changer les noms d'en-têtes. En fait nous proposons d'en faire un concept logique du meta-style spécifique. Ceci signifie qu'il doit exister des commandes telles que **\chaptername** et **\refname** qui puissent être modifiées par le document (en utilisant **\renewcommand**) ou par les options de style. Il est tout à fait autorisé (bien que pas forcément raisonnable) pour un style spécifique d'ignorer ces commandes et de composer tous les en-têtes de la même façon.

6. Considérations institutionnelles

Leslie Lamport a mis un copyright sur LaTeX et fixé le statut quo. Même si ceci est le meilleur moyen d'assurer l'uniformité de LaTeX pour un vaste groupe d'installations et d'utilisateurs, on court le risque que des développements nouveaux et des améliorations deviennent incompatibles. Par conséquent notre proposition d'une nouvelle implémentation est vouée à l'échec si elle devient seulement une de plus parmi beaucoup d'autres (avec seulement un nouveau nom). Nous pensons qu'un tel projet ne devrait être entrepris

que s'il est pris en charge par une organisation qui pourra conduire les développements futurs.

6.1. Maintenance

Au vu de la taille du code source de LaTeX, il est clair qu'il doit être maintenu. Cela ne signifie pas seulement qu'il doive exister quelqu'un qui corrige toute bogue détectée; ce n'est qu'une partie de l'histoire, et pas forcément la plus importante. Une autre tâche nécessaire consiste à développer le logiciel au fur et à mesure que de nouvelles demandes émergent. La conception de logiciel est un travail difficile. Il est encore plus difficile de revenir sur des décisions de conception prises auparavant, parce que, en plus des aspects de développement de logiciel, il faut prendre en considération les questions de compatibilité.

Pour la maintenance d'un système de la taille de LaTeX, il faut un groupe de personnes qui puisse répondre aux demandes et souhaits et décider ce qui peut et doit être réalisé. Ceci signifie que ce groupe doit comprendre des spécialistes de LaTeX et d'autres de styles de document. Ces personnes ne doivent pas forcément être elle-mêmes les implémenteurs. Elles doivent établir les normes et non pas écrire les macros. Néanmoins, elles doivent être suffisamment familières avec LaTeX.

6.2. Suggestions pour un groupe de standardisation de LaTeX

Ceci est certainement possible si Leslie Lamport désire partager le contrôle de LaTeX. Un autre point important est la garantie que ce groupe continue à réaliser sa tâche même si les individus changent. L'endroit où logiquement il faudrait organiser ceci est le Groupe d'utilisateurs de TeX (TUG). Nous pensons que le TUG devrait constituer une commission pour discuter de ces problèmes.

7. Postscriptum

Pendant et après la conférence de Stanford, des discussions avec Leslie Lamport ont eu lieu concernant les suites à donner au présent article. Il est clair maintenant qu'à l'origine, il s'attendait à ce que LaTeX soit remplacé par un autre système de préparation de document peu d'années après son apparition. Ceci n'est pas arrivé, et lui aussi constate maintenant le besoin de nouveaux développements.

Il est d'accord avec les principes contenus dans ce papier qui concernent l'avenir de LaTeX, mais il ne souhaite pas être directement impliqué dans leur implémentation. Pendant les discussions, le plan de développement suivant en deux étapes a été suggéré :

1. Re-concevoir l'interface fichiers de style et la documentation : ceci implique la publication d'un manuel sur l'architecture des fichiers de style. La version de LaTeX correspondante serait la 2.10. Cette version pourrait contenir quelques améliorations mineures visibles par l'utilisateur, mais tous les fichiers d'entrée qui utilisent les fonctionnalités documentées dans le manuel LaTeX actuel devraient fonctionner avec la version 2.10.
2. Produire une implémentation de LaTeX entièrement nouvelle, version 3.0, suivant les principes mis en évidence dans ce papier. Elle devrait être compatible par le haut en ce sens qu'il devrait être possible de traiter n'importe quel document existant grâce à l'ajout d'une option de style.

Le calendrier d'un tel travail ne peut pas être fixé actuellement puisque nous

ne savons pas encore clairement quelle proportion de notre temps nous pourrons consacrer à ce projet, ni quelle aide viendra des différentes parties prenantes.

Références bibliographiques

[Bar88] Frederick H. BARTLETT, « Automatic Page Balancing Macros Wanted. », in *TUGboat* vol. 9 (n ° 1), p. 83, 1988.

[Cla89] Malcolm W. CLARK, « International Standards and TEX. », in *TUGboat* vol. 10 (n ° 2), p. 153–156, 1989.

[Knu86] Donald E. KNUTH, *The TEXbook. Computers and Typesetting* Vol. A. Reading, Massachusetts: Addison Wesley, 1986.

[Lam85] Leslie LAMPORT, LATEX: *A Document Preparation System. User's Guide and Reference Manual*. Reading, Massachusetts: Addison Wesley, 1985.

[Mit88] Frank MITTELBACH, « A new implementation of the array- and tabular- environments. », in *TUGboat* vol. 9 (n ° 3), p. 298–314, 1988.

[Mit89a] Frank MITTELBACH, « A new implementation of the array- and tabular- environments — addenda. », in *TUGboat* vol. 10 (n ° 1), p. 103–104, 1989.

[Mit89b] Frank MITTELBACH, « The doc option. », in *TUGboat* vol. 10 (n ° 2), p. 245–273, 1989.

[Mit89c] Frank MITTELBACH, « An Extension of the LATEX theorem environment. », in *TUGboat* vol. 10 (n ° 3), p. 416–426, 1989.

[Mit89d] Frank MITTELBACH, « An environment for multi-column output. », in *TUGboat* vol. 10 (n ° 3), p. 407–415, 1989.

[MS89a] Frank MITTELBACH et Rainer SCHÖPF, « A new font selection scheme for TEX macro packages — the basic macros. », in *TUGboat* vol. 10 (n ° 2), p. 222–238, 1989.

[MS89b] Frank MITTELBACH et Rainer SCHÖPF, « With LATEX into the Nineties. », in *TUGboat* vol. 10 (n ° 4), p. 681–690, 1989.

[MS89c] Frank MITTELBACH et Rainer SCHÖPF, « LATEX limitations and how to get around them », 1989, (à paraître).

[MS90] Frank MITTELBACH et Rainer SCHÖPF, « A new font selection scheme for TEX macro packages — the LATEX interface. », in *TUGboat* vol. 11 (n ° 1), 1990.

[Par88] Hubert PARTL, « German TEX. », in *TUGboat* vol. 9 (n ° 1), p. 70–72, 1988.

[Pod86] Sunil PODAR, Enhancements to the Picture Environment of LATEX. Dept. of Computer Science, S.U.N.Y. at Stony Brook, Technical Report 86-17, version 1.2, juillet 1986.

[Pri87] Lynne A. PRICE, « SGML and TEX. », in *TUGboat* vol. 8 (n ° 2), p. 221–225, 1987.

[Rog88] David F. ROGERS, « A Page Make-up Challenge. », in *TUGboat* vol. 9 (n ° 3), p. 292–293, 1988.

[Sch89] Rainer SCHÖPF, « Drawing histogram bars inside the LATEX picture environment. », in *TUGboat* vol. 10 (n ° 1), p. 105–107, 1989.

[Won85] Reinhard WONNEBERGER, « Stream lists and related list types for LATEX. », in *TUGboat* vol. 6 (n ° 3), p 156–157, 1985.

[Won86] Reinhard WONNEBERGER, « Chapter Mottos and Optional Semi-Parameters in General and for LATEX. », in *TUGboat* vol. 7 (n ° 3), p. 177–185, 1986.

[ZR89] Joost ZALMSTRA et David F. ROGERS, « A Page Make-up Macro. », in *TUGboat* vol. 10 (n ° 1), p. 73–81, 1989.