Cahiers GUTenberg

VERS UNE VERSION ARABISÉE DE TEX

Oussama Boughaba, Seifeddine Boutalbi, Michel Fanton

Cahiers GUTenberg, nº 10-11 (1991), p. 25-44.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1991___10-11_25_0>

Vers une version arabisée de TEX

Oussama BOUGHABA, Seifeddine BOUTALBI et Michel FANTON

Centre d'Etudes et de Recherche en Traitement Automatique des Langues
équipe de recherche de l'INALCO associée au CNRS et au Collège de France
adresse : CERTAL - INALCO, 2 rue de Lille 75007 Paris
fax : [33] (1) 49 26 42 99,
courier électronique : UILO007@FRORS31 *(EARN/bitnet).*

Abstract. *This paper presents the state of development of an arabicized version of TEX.*

Résumé. Cet article présente l'état actuel du développement d'une version arabisée de TEX sous DOS.

Mots clef : TEX, arabe

1. Introduction

Cet article vise à présenter l'état d'avancement d'un projet d'arabisation de TEX. Nous n'insisterons pas ici sur les difficultés inhérentes à l'utilisation d'un logiciel fondamentalement pensé pour les langues écrites en caractères latins et, même plus précisément pour l'anglais. Ce thème fait l'objet d'une autre communication donnée à ce même colloque [Fanton91], dans laquelle sont décrits les différents niveaux possibles de compromis nécessaires à une telle entreprise.

L'une des conclusions de cet article est que l'obtention d'une typographie conforme aux traditions typographiques arabes demanderait l'écriture d'un programme dont certains aspects seraient très différents de ceux que l'on trouve dans TEX.

Le projet que nous présentons ici est donc d'un tout autre ordre.

1.1. Motivation et définition du projet

L'objectif poursuivi dans la phase actuelle du projet est beaucoup plus modeste. TEX et surtout LATEX sont devenus des standard de fait pour un

nombre important de publications scientifiques internationales. Or, même si la langue utilisée dans ces publications est, la plupart du temps, l'anglais, le contenu scientifique des articles peut faire appel à d'autres langues.

Le domaine de recherche du CERTAL, le traitement automatique des langues, fait partie de cette catégorie. Des considérations identiques ont conduit les Japonais à développer JTEX et JLATEX, d'autres chercheurs à s'intéresser au grec ou aux alphabets cyrilliques.

A notre connaissance rien d'achevé ou en cours d'achèvement n'existant pour la langue arabe[1], nous avons décidé de lancer un projet allant dans cette direction.

1.2. Différentes phases du projet

Les opérations à effectuer pour composer un texte mixte (comportant des parties en caractères arabes et des parties en caractères latins) peuvent être décomposées en 5 étapes:

1. saisie du texte dans un code normalisé pouvant faire l'objet d'une relecture et d'une correction aisée
2. conversion de ce fichier dans une forme manipulable par TEX
3. modification du programme source TEX pour qu'il puisse manipuler des textes mixtes gauche-droite droite-gauche.
4. création une police arabe avec METAFONT.
5. écriture d'un pilote, pour au moins une imprimante, capable d'imprimer ces textes mixtes.

1.3. Utilisation des travaux antérieurs

Pour parvenir à des résultats tangibles dans des délais raisonnables nous avons délibérément pris le parti de la simplicité à tous les niveaux. Cette position de principe nous a conduit à avoir recours à un certain nombre de travaux antérieurs:

[1] Nous avons consulté sur ce point le Pr. MacKay, éminent spécialiste à la fois de l'arabe et de TEX

— les travaux de A. Lakhdar-Ghazal et P. MacKay en matière de typographie arabe[2].

— les recommandations de D. Knuth et P. MacKay pour la manipulation de textes bi-directionnels[3] avec TEX.

— les sources de TurboTEX et les programmes de traduction

$$\text{WEB} \rightarrow \text{PASCAL} \rightarrow \text{C}$$

développés par la société Kinch[4].

— l'arabisation du système DOS développée initialement par la société ALIS et désormais distribuée par la société MICROSOFT.

— notre propre expérience d'arabisation d'un système de composition numérique professionnel[5].

La machine cible choisie pour cette première implémentation est le compatible IBM-PC fonctionnant sous système DOS. La raison de ce choix est la possibilité de disposer, sous le même système, d'une version de TEX et de METAFONT, de la saisie directe en arabe et d'une imprimante à laser HEWLETT-PACKARD, pour laquelle on peut aisément écrire un pilote, ou modifier l'un des nombreux pilotes préexistant.

1.4. Etat d'avancement actuel du projet (avril 1991)

saisie Prototype d'analyse contextuelle en cours de réalisation

formatage Première version TEX–XET réalisée. A tester.

impression Première version du pilote d'imprimante

fonte Première version de la partie alphabétique de l'ASV-CODAR entièrement réalisée. Une seconde est en cours de réalisation.

[2] cf. [Lakhdar-Ghazal83], [MacKay85], [MacKay86]
[3] cf. [Knuth-MacKay87]
[4] cf. [Kinch-Vollbrecht88]
[5] cf. [Fanton89]

2. Saisie de textes bi-directionnels arabe-latin

Le problème qui se pose est de pouvoir saisir des textes comportant des parties en arabe destinés à être formatés ultérieurement par TEX. L'utilisation d'une transcription en caractères latins de l'arabe ne peut être envisagée sérieusement, car, quel que soit le système utilisé, les textes saisis sont difficiles à relire et à corriger[6]. Il faut donc pouvoir créer directement, de préférence avec un éditeur standard, des textes mixtes. Ce problème a donné lieu à des études nombreuses dans la dernière décennie. Sur le marché, on peut trouver deux solutions bénéficiant d'une diffusion commerciale significative.

1. ARABIX, version bi-alphabétique et bi-directionnelle de UNIX développée par la société IMT. Malheureusement ce système n'existe pas pour toutes les versions d'UNIX et, encore moins pour toutes les machines[7].

2. ARABDOS version bi-alphabétique et bi-directionnelle de MS-DOS 3.3.

Ces deux logiciels permettent d'utiliser les éditeurs standard monolingues des systèmes considérés[8]. Notre choix s'est porté sur ARABDOS, dont nous disposions antérieurement pour d'autres applications.

ARABDOS permet d'utiliser plusieurs variantes du code arabe standard. Les choix effectués par les organismes arabes de normalisation impliquent certaines contraintes que nous allons aborder au paragraphe suivant.

2.1. Code arabe standard

2.1.1. Origine du code

Le code arabe standard est l'aboutissement de longues recherches menées par le Pr. Lakhdar-Ghazal et son équipe de recherche de l'IERA[9]. Le code sur 7 bits qui en est issu a d'abord porté le nom de CODARU[10], puis après

[6] Nous en avons fait l'expérience douloureuse au début de nos travaux sur la langue arabe. Note de M. FANTON

[7] En particulier, il n'existe pas de version pour SUN, machines UNIX auxquelles nous pouvions avoir accès.

[8] Les spécialistes donnent le nom d'arabisation externe à ce type d'approche. Notons que cette technique fonctionne pour les éditeurs de programme mais pas pour les traitements de textes pour les raisons que nous allons voir par la suite.

[9] Institut d'Etudes et de Recherche pour l'Arabisation, Rabat Maroc

[10] COD-e AR-abe U-nifié

quelques modifications destinées à obtenir le consensus de l'ensemble des états arabes CODARU-FD[11]. Ce système a été adopté par l'association arabe de normalisation ASMO et par l'ISO. Les codes 8 bits qui ont vu le jour par la suite n'en sont que des variantes.

2.1.2. Philosophie du code arabe standard

L'objectif principal de ce système de codage est de réduire le plus possible le nombre de codes nécessaires pour représenter l'alphabet arabe. Cet objectif a été atteint par l'élimination des informations redondantes, chaque fois que cela était possible. Par redondance, les concepteurs entendaient: *informations aisément calculables.*

Dans [Fanton91], on a vu que les lettres arabes utilisées en imprimerie, pouvaient, héritage de la calligraphie manuelle, prendre jusqu'à 4 formes différentes: initiale, médiane, finale, isolée (plus si l'on prend en compte les ligatures et les formes allongées servant à la justification des textes). Ces formes dépendent de leur position dans le mot et de l'obligation de lier ou de ne pas lier les lettres entre elles.

Il leur est apparu que le principal phénomène de redondance provenait de l'affectation de codes distincts aux différentes formes que peuvent prendre les lettres en arabe.

Ce choix est très pertinent d'un point de vue linguistique et économique sur le plan informatique: il permet de placer les codes nécessaires à un alphabet arabe complet dans les 96 positions disponibles que comporte l'ASCII 7 bits normalisé alors que cela serait impossible si des codes distincts étaient attribués aux différentes formes de lettres.

2.1.3. Codes 8 bits

Le système de codage que nous venons d'évoquer est un code 7 bits. Il était utilisé pour les terminaux de machines fonctionnant sous système UNIX. Les caractères arabes occupant la place des caractères latins dans cette version, un caractère signalant le passage de l'arabe au latin ou du latin vers l'arabe devait figurer dans tous les fichiers mixtes. La valeur du code de changement de table n'était pas précisé par la norme et donc variait de terminal en terminal.

[11] CODARU-Forme Définitive

L'apparition du code 8 bits des IBM-PC a permis de trouver une solution plus confortable, en déplaçant le code arabe dans la *partie haute de la table.* La première version a consisté en une simple translation de la table 7 bits. Les autres correspondent à divers compromis sur la cohabitation entre caractères arabes, caractères latins accentués et caractères graphiques.

2.1.4. Contraintes liées à l'utilisation du code arabe standard

Cependant le prix à payer pour ces avantages indéniables, est la nécessité de procéder à une *analyse contextuelle*[12] pour déterminer la forme exacte de la lettre à représenter sur le périphérique de sortie. Tout dispositif destiné à imprimer ou à visualiser un texte arabe saisi à l'aide de cette technique doit opérer une traduction du code standardisé vers le code du périphérique de sortie, qui, lui, peut ne pas être standard.

2.2. Pré-traitement des fichiers à formater

La nécessité de procéder à une analyse contextuelle pose immédiatement le problème de la place que cette analyse doit occuper dans la séquence des opérations d'impression et d'affichage. La solution habituelle consiste à reporter sur le pilote de l'imprimante ou de l'écran le traitement impliqué par le codage standard de l'arabe.

Cette technique est inadaptée lorsqu'on veut faire manipuler un fichier par un formateur de texte comme TEX. Dans ce cas toutes les informations nécessaires au formatage doivent se trouver dans le fichier ou être calculées par le formateur lui-même. Un tel traitement impliquerait, sauf à modifier profondément l'algorithme de justification, deux *passes*, la première procédant à la détermination de la forme des lettres, au traitement des ligatures et de la voyellation dans les parties arabes du texte, la seconde procédant au formatage.

Il nous est apparu plus simple, au niveau d'une première réalisation, de procéder à ces manipulations en dehors de TEX par le biais d'un pré-traitement.

[12] cf. [Fanton91] pour la description de l'algorithme

3. Formatage de textes bi-directionnels

TEX a été conçu pour composer des documents qui s'écrivent et se lisent de gauche à droite et de haut en bas. Les langues sémitiques (hébreu, arabe), s'écrivent de droite à gauche. C'est également le cas de toutes les langues qui utilisent les caractères arabes[13]. Ce problème est donc très important et justifie largement d'envisager la création d'une version de TEX capable de traiter des documents comportant un mélange de texte droite-gauche et gauche-droite.

Cette question est abordée rapidement dans [MacKay86] et traitée plus complètement dans [Knuth-Mackay87]. Les solutions proposées varient selon le type de textes à formater.

3.1. Techniques de formatage des textes droite-gauche

3.1.1. Textes gauche-droite comportant de courtes insertions droite-gauche

L'idée la plus simple consiste à renverser le segment droite-gauche à l'aide d'une macro-commande. [Knuth-MacKay87] en donne un exemple permettant de traiter des mots isolés. Dans [Boughaba90] ce principe a été repris et développé: on y trouve une macro-commande permettant de traiter des textes courts. L'auteur fait remarquer que le caractère récursif des traitements exécutés par ce type de macro-commande demande beaucoup de place mémoire, ce qui constitue une sérieuse limitation.

3.1.2. Textes intégralement droite-gauche

Dans le cas de documents entièrement écrits en arabe, on pourrait envisager d'écrire un programme qui imprime le contenu d'un fichier DVI en mode miroir. Cette technique est rendue possible par l'indépendance, voulue par D. Knuth, des étapes de formatage et d'impression des documents.

3.1.3. Textes mixtes

Aucune des deux techniques qui viennent d'être mentionnées n'étant réellement satisfaisante, [Knuth-MacKay87] en propose une plus générale, qui permet de faire alterner textes gauche-droite et droite-gauche avec un

[13] **Le persan, l'ourdou (langue du Pakistan) et le pashto (langue de l'Afghanistan) ne sont pas des langues sémitiques, mais utilisent l'alphabet arabe complété par des caractères qui leur sont propres**

niveau d'imbrication quelconque. D. Knuth propose de donner le nom de TEX-XET[14] au programme incorporant les modifications suggérées. Selon lui on ne peut qualifier le programme ainsi modifié de nouvelle version de TEX dans la mesure où la compatibilité n'est qu'ascendante: tous les programmes TEX pourront être traités par TEX-XET mais le contraire n'est évidemment pas vrai. Nous respecterons le souhait de D. Knuth en utilisant désormais le nom TEX-XET pour ce système[15].

3.2. Description du système TEX-XET

3.2.1. Méthode générale

La méthodologie suggérée dans [Knuth-MacKay87] consiste à ajouter quatre nouvelles primitives au programme TEX :

— *beginL* : début de texte gauche-droite

— *endL* : fin de texte gauche-droite

— *beginR* : début de texte droite-gauche

— *endR* : fin de texte droite-gauche

qui servent à délimiter les différentes parties d'un document bi-directionnel.

L'exemple ci-dessous illustre le contenu d'un fichier TEX comprenant un texte mixte : \beginR *Ceci est un texte droite-gauche* \beginL ceci est un texte gauche-droite \endL *qui contient un texte gauche-droite* \endR

3.2.2. Impact des nouvelles primitives sur les fichiers DVI

Cette extension de l'ensemble des primitives de TEX, nécessite la définition de deux nouveaux codes opérations :

— *begin_reflect*

— *end_reflect*

générés au niveau du fichier DVI. Les fichiers DVI incluant les codes opérations supplémentaires générés par le système TEX-XET seront dénommés DVI-IVD dans la suite du texte.

[14] Dans [Knuth-MacKay87] XET est imprimé avec une fonte inverse dont nous ne disposons pas

[15] L'adoption du nom choisi par D. Knuth est d'autant plus justifiée que les modifications propres à l'arabe interviennent en dehors du programme TEX-XET

Ainsi, pour chaque primitive *beginL* ou *beginR* qui sera insérée au niveau du document TEX, on insérera dans le fichier DVI–IVD correspondant le code d'opération *begin_reflect*. De même, à chaque primitive *endL* ou *endR* du fichier TEX correspondra, dans le fichier DVI–IVD, le code opération *end_reflect*.

Cette extension du système TEX produit des fichiers DVI–IVD qui renferment le code opération *begin_reflect* au début de tout texte écrit en arabe, et *end_reflect* à la fin de ce texte. Le programme d'édition peut ainsi détecter le changement de direction et imprimer cette partie du texte à partir de la droite.

D'une manière plus générale, les modifications décrites plus haut permettent au système TEX–XET de composer des documents mixtes en différenciant, à l'aide des nouvelles primitives, les parties du texte qui s'écrivent de droite à gauche de celles qui s'écrivent de gauche à droite. Elle permet ainsi de produire des documents, où des textes droite-gauche et gauche-droite s'imbriquent de manière très complexe.

3.3. Implémentation du système TEX–XET

3.3.1. Choix de TurboTEX

Pour implémenter cette solution nous avons opté pour le système TurboTEX qui est une implémentation en langage C, sous UNIX et MS-DOS, du programme TEX écrit en langage WEB par D. Knuth [Kinch-Vollbrecht88].

TurboTEX permet non seulement d'obtenir le programme de TEX en langage Pascal, après son traitement par le module TANGLE. Il le traduit aussi en langage C à l'aide du programme PASCHAL. De plus, il offre la possibilité, en sélectionnant les options adéquates, de compiler ce même programme sous UNIX et sous MS-DOS.

La manière dont TurboTEX a été conçu, présente pour nous les avantages suivants :

— la possibilité de pouvoir implémenter notre système TEX–XET dans un environnement MS-DOS ;

— bénéficier des nombreux pilotes d'imprimantes déjà disponibles dans un environnement MS-DOS, ce qui simplifie l'écriture des programmes capables de lire et d'imprimer les fichiers délivrés par TEX–XET ;

— obtenir une meilleur performance du système TEX–XET, grâce à son implémentation en langage C, qui lui permet d'être beaucoup plus rapide qu'une implémentation réalisée en Pascal [Kinch-Vollbrecht88].

3.3.2. Difficultés d'implémentation

Utilisation de deux environnements différents L'obligation de travailler simultanément dans deux environnements différents (MS-DOS et et UNIX) accentue les difficultés d'implémentation du système TEX–XET. En effet, les programmes TANGLE et PASCHAL et les fichiers qu'ils ont à traiter occupent une place mémoire importante et ne peuvent être exécutés, voire pour certains compilés sous MS-DOS, au moins avec les compilateurs dont Kinch et nous-mêmes disposions à ce moment à savoir MICROSOFT C 5.1. Il est possible que ce problème soit désormais surmontable avec les compilateurs couplés à un DOS EXTENDER capable d'exploiter les ressources du mode protégé du processeur 386. Ce test fait partie de nos projets à court terme. S'il était positif, l'intégralité du processus de génération de la nouvelle version pourrait être réalisée dans un environnement DOS, à condition de disposer d'un 386 à 32 bits et ... de suffisamment de mémoire.

Problèmes de portage des programmes fournis par TurboTEX Il est souvent nécessaire de modifier partiellement ou complètement tous les fichiers *makefile* qui permettent de compiler le programme TEX dans les deux environnements MS-DOS et UNIX pour obtenir une compilation correcte. La plupart des erreurs sont causées par des problèmes classiques de portage d'un programme dans des environnements différents.

En effet, les difficultés peuvent provenir du fait qu'un compilateur supporte ou ne supporte pas le modèle de compilation HUGE, qu'il accepte ou non une déclaration du type UNSIGNED ou VOID, ou encore de la représentation que la machine utilise pour coder des entiers ou des pointeurs.

Cycle de mise au point très long Le nombre élevé d'opérations que nécessite l'obtention de la version exécutable sous DOS de TEX–XET à partir du programme-source en langage WEB, rend très fastidieux et très long le processus habituel de mise au point pour ce programme. Ce processus pourrait être accéléré en utilisant uniquement l'environnement MS-DOS, à l'aide des compilateurs mentionnés au paragraphe précédent.

Une version complète de TEX–XET est disponible actuellement. Toutes les sources des modifications et des adjonctions écrites en langage WEB et ajoutées aux programmes initiaux de TEX, sont décrites dans [Boughaba90].

4. Impression de textes bi-directionnels

4.1. Choix d'un pilote

Afin de construire les pilotes d'imprimantes capables de lire les fichiers DVI–IVD et d'interpréter les nouvelles commandes générées par TEX–XET, nous avons utilisé les programmes sources des pilotes d'imprimantes écrits en langage C, pour traiter les fichiers DVI composés par TEX. Ces programmes font partie du logiciel TurboTEX et sont une implémentation conforme à la description donnée dans [Beebe87].

Pour effectuer les tests, nous avons intégré au pilote de l'imprimante HEWLETT-PACKARD LASERJET les fonctions et procédures nécessaires pour effectuer les traitements spécifiques aux fichiers DVI–IVD. Ce pilote modifié nous a permis d'imprimer le texte présenté en annexe.

4.2. Méthode de construction

[Knuth-Mackay87] ne donne qu'une esquisse de la démarche à suivre. Nous avons complètement développé et implémenté la solution dont le principe est décrit ci-dessous:

Les codes d'opérations *begin_reflect* et *end_reflect* permettent de détecter tout changement de direction au niveau du fichier DVI–IVD. Ainsi si on est dans un mode d'écriture gauche-droite, et qu'on rencontre le code d'opération *begin_reflect*, on sauvegarde la position courante P_0 dans le fichier DVI–IVD, et on continue de parcourir ce fichier sans imprimer de texte ni exécuter de commande, et ce jusqu'à la rencontre du code opération *end_reflect* ou d'un nouveau code opération *begin_reflect*. Dans ce cas, on sauvegarde encore une fois la position courante P_1 dans le fichier DVI–IVD, et on parcourt à nouveau ce fichier dans le sens $P_1 \rightarrow P_0$ en imprimant cette fois les textes et en exécutant les commandes du fichier DVI–IVD. Une fois la position P_0 atteinte, on revient à la position P_1 et on continue de traiter le fichier DVI–IVD selon le même principe.

4.3. Difficultés d'implémentation

Comme le code opération *begin_reflect* peut être généré aussi bien par un *beginL* que par un *beginR* le programme d'impression doit être capable de savoir quand il doit imprimer de gauche à droite et quand il doit imprimer

de droite à gauche. Le même problème doit être réglé pour le *endL* et le *endR* qui génèrent tous les deux un *end_reflect*.

L'ordre dans lequel les commandes TEX sont exécutées est très important. En effet, si, par exemple, on exécute une commande W_0 suivie d'une commande W_1 l'effet obtenu est différent de celui qu'on obtiendrait en exécutant les commandes dans l'ordre $W_1 - W_0$. D'où la nécessité, dans un texte droite-gauche où l'impression est inversée, de prendre toutes les précautions pour composer chaque ligne, en respectant les effets produits par chaque commande, comme si elle intervenait dans la composition d'un texte gauche-droite.

La majorité des changements que nous avons effectués portent sur le contenu du fichier *prtpage.h* [Beebe87]. En effet, c'est dans ce fichier que se trouve la majeure partie du programme qui sert à interpréter les commandes d'un fichier DVI. Nous avons, dans un premier temps, ajouté les deux nouveaux codes-opération *begin_reflect* et *end_reflect*, puis nous avons tenté, dans la mesure du possible de ne procéder, au niveau de ce programme, qu'à des appels de fonctions. Les définitions de ces fonctions ont été reportées dans un autre fichier.

5. Création d'une fonte arabe pour TEX–XET

5.1. Choix d'une technique

Lorsqu'on veut créer une fonte utilisable par TEX, deux techniques sont envisageables:

1. Employer une fonte existante et créer les fichiers *.tfm* et *.pxl* correspondants. C'est la méthode décrite dans [Cousquer90] pour les caractères chinois.
2. Créer une fonte avec un logiciel adapté:
   (a) En définissant les caractères point par point, ce qui peut devenir très long et fastidieux lorsqu'on souhaite obtenir une résolution élevée.
   (b) Par numérisation, à l'aide d'un scanner, d'un dessin de caractère.
   (c) En utilisant une définition mathématique des caractères, par points de contrôle reliés par des courbes de lissage d'épaisseur variable.

C'est cette dernière technique que D. Knuth a implémentée dans le programme METAFONT dont il s'est servi pour créer les fontes *cmr* qui accompagnent TEX.

Dans l'esprit de D. Knuth, la technique par points de contrôle et courbes permet de simuler le mouvement d'une plume. Elle présente des difficultés certaines pour représenter des caractères dont la forme est issue des inscriptions lapidaires romaines et non d'une adaptation de l'écriture manuscrite.

P. MacKay[16] fait remarquer que ce qui constitue peut-être un inconvénient pour les caractères latins devient un atout pour l'écriture arabe d'imprimerie construite sur la base de la calligraphie manuscrite.

Ne disposant pas de fonte arabe pré-existante, nous avons décidé d'employer METAFONT pour en créer une pour TEX–XET.

5.2. Création de caractères avec METAFONT

La création d'un caractère numérisé avec METAFONT implique la suite d'opérations suivantes:

1. Création du dessin du caractère à numériser.
2. Analyse mathématique du caractère à numériser:
   (a) Détermination des points de contrôle.
   (b) Interpolation des points de contrôle à l'aide de courbes.
3. Ecriture du programme METAFONT : le programme doit spécifier dans le détail l'analyse effectuée au point précédent.
   (a) Choix de la plume.
   (b) Fixation des points de contrôle.
   (c) Définition des coordonnées des points de contrôle.
   (d) Marquage des points de contrôle sur le graphe.

En ce qui concerne le premier point, n'étant pas typographes professionnels, nous avons dû chercher un modèle de fonte à numériser.

[16] cf. [MacKay85] et [MacKay86]

5.3. Choix d'un modèle

Dans [Fanton91] on trouve une discussion des critères que devrait respecter un système typographique conforme aux traditions typographiques arabes. Mais, comme cela est dit dans l'introduction du présent article, l'objectif poursuivi par le projet décrit ici est tout autre : il s'agit de pouvoir composer des articles scientifiques comportant des fragments plus ou moins importants de texte en arabe. Cet objectif peut s'étendre jusqu'à la composition de documents complets en arabe, pour peu qu'on ne soit pas trop respectueux des traditions typographiques mentionnées plus haut.

Notre choix s'est porté sur la fonte ASV-CODAR[17] créée par le Pr. Lakhdar-Ghazal.

Cette fonte présente trois avantages essentiels[18] pour les objectifs que nous nous sommes fixés :

1. Cette fonte permet de composer des textes arabes avec un nombre très réduit de caractères: 84 pour le *système pur* qui inclut la ponctuation et les chiffres, 23 de plus pour le *système total* qui comprend une deuxième forme pour certaines lettres destinée à améliorer l'esthétique des textes composés et des lettres étrangères (persanes) utilisées dans la transcription arabe des mots étrangers.
2. La forme des caractères a été simplifiée.
3. Elle permet une voyellation simple et lisible des textes.

Les deux premiers points sont essentiels pour l'obtention rapide d'une fonte complète. Le premier point réduit le *nombre*, le second la *complexité* des programmes METAFONT à écrire. Le troisième est fondamental pour des textes à but scientifique. La voyellation est souvent utile pour lever les ambiguïtés de certains énoncés. Elle est indispensable dans les applications pédagogiques et lorsqu'il s'agit de traiter de linguistique arabe.

[17] Arabe Standard Voyellé
cf. [Lakhdar-Ghazal83] pour la description complète de la fonte et [Fanton91] qui contient nos commentaires.
Nous avons conscience des implications juridiques de cette méthode. Nous avons pris contact avec le Pr. Lakhdar-Ghazal pour l'informer de la réalisation de ce prototype.

[18] Il s'agit là des objectifs explicitement poursuivis par le Pr. Lakhdar-Ghazal lorsqu'il a défini les spécifications de l'ASV-CODAR

5.4. Numérisation de la fonte ASV-CODAR

5.4.1. méthode utilisée

Le processus de numérisation d'une fonte avec METAFONT est long, mais il possible de tirer parti de certaines caractéristiques de METAFONT et de l'ASV-CODAR pour accélérer le processus:

1. METAFONT est un langage structuré permettant de programmer, dans des procédures, des formes partielles qu'on peut retrouver dans différents caractères.
2. Les concepteurs de l'ASV-CODAR ont, par ailleurs, exploité au maximum les ressemblances qui pouvaient intervenir entre les différentes lettres et signes orthographiques de l'arabe.

Dans cette fonte, par exemple, beaucoup de caractères ne diffèrent que par la présence d'un, deux ou trois points au-dessus ou au-dessous d'une forme commune.

Ces groupes de un deux ou trois points peuvent faire l'objet de procédures appelables pour numériser les différentes lettres qui les utilisent.

5.4.2. Exemples

Dans l'ASV-CODAR, 5 lettres ont une même forme de base :

— *ba* qui correspond au *b*
— *ta* qui correspond au *t*
— *tha* lettre qui se prononce comme le *th* anglais de *thing*
— *nun* qui correspond au *n*
— *ya* qui correspond au *y*

Ces lettres se différencient par :

— *1 point* sous le *ba*
— *2 points* sur le *ta*
— *3 points* sur le *tha*
— *1 point* sur le *nun*

— *2 points* sous le *ya*

Ces 5 lettres peuvent donc être programmées à l'aide d'une procédure pour la forme commune et par appel des procédures générant les groupes de points. Avec cette technique il est possible de réaliser les 33 caractères de la partie alphabétique avec seulement 16 formes communes différentes.

5.4.3. Programmes METAFONT *correspondant*

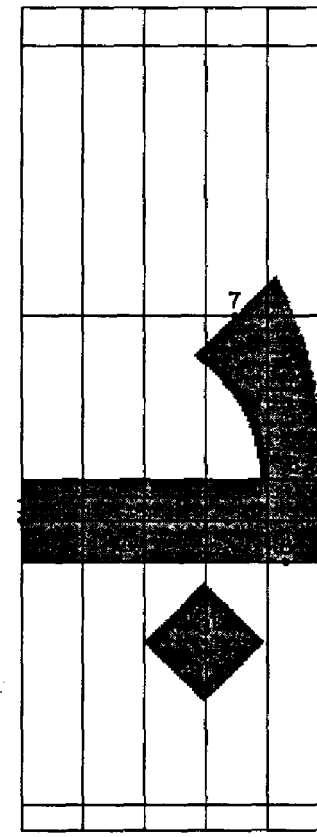
Extrait du fichier contenant les macros-commandes

```
% Fichier de base FMABASE.MF
% extension du fichier Plain.mf
                ...
% Definition des macros Commandes
...
% macro nokta (point mesure)
  def nokta(suffix $) =
fill    ((x.$-u), y.$)--
( x.$,(y.$-u))--
((x.$+u), y.$)--
( x.$,(y.$+u))--
cycle;
enddef;

% macro deux_nokta (deux points cote a cote)
  def deux_nokta(suffix $,  $$) =
z.$$=z.$-(2u,0);
nokta($);
nokta($$);
  enddef;

% macro trois_nokta_sup (trois points)
  def trois_nokta_sup(suffix $, $$, $$$) =
deux_nokta($,$$);
z.$$$=z.$+(-u,u+p_3o);
nokta($$$);
  enddef;
...
% macro wasl (trait de liaison)
  def wasl(suffix $, $$) =
numeric angle;
angle=90;
penpos$(Qalem, angle);
penpos$$(Qalem, angle);
penstroke z.$e--z.$$e;
  enddef;
```

```
% macro cinah (baa, noon, ta, tha ...)
  def cinah(suffix $, $$, @@) =
numeric angle[];
angle.$=angle.@@=0;
angle.$$=45;
penpos$(Qalem, angle.$);
penpos@@(coef1*Qalem, angle.@@);
penpos$$(round(2*cosd(angle.$$)*Qalem), angle.$$);
penstroke z.$e..{up}z.@@e..z.$$e;
  enddef;
...
```

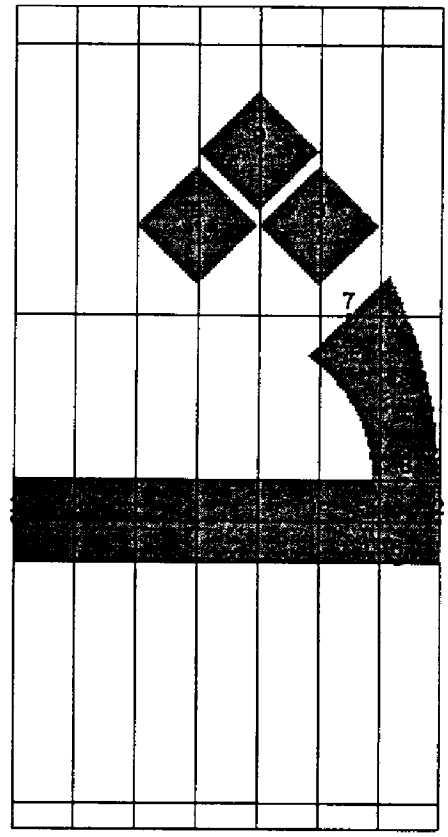
Programme de creation du caractère ba

```
% Appel macro
%   debut_car_arab(CODE,CHASSE)
 debut_car_arab(2,5);

% Choix de la plume d'ecriture
        pickup plum_cir;

% Definition des coordonnees
z1=(0,qaida);
z2=(w,qaida);
z3=(w-2u, Nokta_inf);
x6r=x8r=w; y6=Qalem_inf;
x7=w-3/2u; y7=cinah_ht;
y8=Qalem_sup;

% Appel des macros definies dans fmabase
wasl(1,2);
cinah(6,7,8);
nokta(3);

% Marquage des points de controles
labels(1,2,3,6,7,8);

fin_car;
```



Programme de creation du caractère tha

```
% Appel macro
% debut_car_arab(CODE,CHASSE)
 debut_car_arab(1,7);

% Choix de la plume d'ecriture
pickup plum_cir;

% Definition des coordonnees
z1=(0,qaida);
z2=(w,qaida);
z3=(w-2u, Nokta_sup);
x6r=x8r=w; y6=Qalem_inf;
x7=w-3/2u; y7=cinah_ht;
y8=Qalem_sup;

% Appel des macros
wasl(1,2);
cinah(6,7,8);
trois_nokta_sup(3,4,9);

% Marquage des points de controles
labels(1,2,3,4,6,7,8,9);
 fin_car;
```



Références bibliographiques

[André85] Jacques ANDRÉ, "Paramètres typographiques", in actes de la conférence *Typographie & informatique*, INRIA, Rennes 1985

[Boughaba90] Oussama BOUGHABA, "Une étude du système TEX et de son arabisation", mémoire de DEA de Traitement Automatique des Langues, INALCO Paris 1990.

[Beebe87] Nelson H. F. Beebe, "A TEX DVI Driver Family", revision 2.07, Center for Scientific Computation, University of Utah, Salt Lake City, 1987.

[Boutalbi90] Seifeddine BOUTALBI, "Une méthodologie pour la création d'une fonte numérique arabe", mémoire de DEA de Traitement Automatique des Langues, INALCO Paris 1990.

[Cousquer90] Alain Cousquer, "En chinois dans le TEXte", in *Cahiers GUTenberg*, no 6, Juillet 1990.

[Fanton89] Michel FANTON, "Arabisation d'un système de photocomposition numérique", communication donnée au colloque *informatique et langue arabe* organisé par l'Institut

du Monde Arabe et la Commission Economique et Sociale des Nations Unies pour l'Asie Occidentale à Paris en octobre 1989.

[Hamm75] Roberto HAMM, "Pour une typographie arabe", Sindbad, Paris, 1975.

[Kinch-Vollbrecht88] Richard KINCH and Jennifer L. VOLLBRECHT, "A new port in C for UNIX and MS-DOS", in *TUGboat*, volume 9 no 1, 1988.

[Knuth83] Donald E. KNUTH, "The WEB system of structured documentation", Stanford Computer Science Department, 1983, STAN-83-980.

[Knuth86-1] Donald E. KNUTH, "The METAFONT book", Addison-Wesley 1986.

[Knuth86-2] Donald E. KNUTH, "Computer modern type faces", Addison-Wesley 1986.

[Knuth86-3] Donald E. KNUTH, "TEX the program", Addison-Wesley 1986.

[Knuth86-4] Donald E. KNUTH, "The TEX book", Addison-Wesley 1986.

[Knuth-Mackay87] Donald E. KNUTH and Pierre A. MACKAY, "Mixing right-to-left texts with left-to-right texts", *TUGboat*, Volume 8 (1987), no. 1, pp. 14–25.

[Knuth-Plass81] Donald E. KNUTH and Michael PLASS, "Breaking paragraphs into lines", in *Software Practice and Experience*, volume 11, 1981

[Lakhdar-Ghazal83] Ahmad LAKHDAR-GHAZAL, "l'alphabet arabe et les machines", in *Applied Arabic Linguistics and Signal & Information Processing, proceedings of the 1st Fall Session of Arab School on Science & Technology*, Rabat (Morocco) 1983, Volume 2 pp. 233–257.

[MacKay85] Pierre A. MACKAY, "Modern font design and typesetting", in *Informatics and Applied Arabic Linguistics, proceedings of the 7th Summer Session of Arab School on Science & Technology*, Zabadani Valley (Syria) 1985, pp. 2A2-1–2A2-7.

[MacKay86] Pierre A. MACKAY, "Typesetting Problem Scripts", *Byte*, February 1986, pp. 201–216.

[Microsoft88] Microsoft, "MS-DOS User's Guide. Arabic Supplement", 1988.

[Rubinstein88] Richard RUBINSTEIN, "Digital typography", Addison-Wesley 1988.

[Southall85] R. SOUTHALL, "METAFONT & the problems of type design", in actes de la conférence *Typographie & informatique*, INRIA, Rennes 1985

[Southall] R. SOUTHALL, "Designing new typefaces with METAFONT", Stanford Computer Science Report no 1074

Fonte Arabe: Corps 10 points.

Magstep 1 .....

Soit l'équation mathématique (المعادلة الرياضية) suivante:

$$2x^2 + 3x + 4 = 0$$

حاول حل هذه المعادلة و نشرح الجواب باستعمال الطريقة الموضحة في الدرس او المنهجية المقررة في كتاب التماريين للسنة الثانية علمي (consulter la page 30 du paragraphe[I]).

Magstep 2 .....

Soit l'équation mathématique (المعادلة الرياضية) suivante:

$$2x^2 + 3x + 4 = 0$$

حاول حل هذه المعادلة و نشرح الجواب باستعمال الطريقة الموضحة في الدرس. او المنهجية المقررة في كتاب التماريين للسنة الثانية علمي (consulter la page 30 du paragraphe[I]).