Cahiers GUTenberg

POLICES, TEX ET CIE

Alain Cousquer, Éric Picheral

Cahiers GUTenberg, nº 9 (1991), p. 3-31.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1991___9_3_0>

Polices, TEX et Cie*

Alain COUSQUER† et Éric PICHERAL‡

† LIFL, *Université de Lille I, F-59655* VILLENEUVE D'ASCQ *Cedex, France*
‡ CICB, *Campus de Beaulieu, F-35042* RENNES *Cedex, France*

Résumé. Le but de cet article est de présenter les principes de gestion des polices de TEX, leurs modes d'utilisation ainsi que le modèle de fonte employé, qui est plus simple que celui de PostScript. Nous montrons également les mécanismes complexes de sélection qui n'apparaissent pas en usage courant et nous terminons par une présentation des principaux fichiers utilisés par TEX.

Abstract. *The purpose of this paper is to introduce the principles of the handling of the fonts in TEX together with their usage, and the pattern of the fonts which is more straightforward than in PostScript. We also explain the complex mechanisms of selection which do not usually appear, and finally we present virtual fonts as well as various files used with TEX.*

Parmi les aspects les plus spectaculaires et les plus séduisants de TEX pour le néophyte figure sans conteste la possibilité de modifier à volonté le style et la taille de l'écriture ; c'est d'ailleurs une des premières démonstration que fait D.E. KNUTH des capacités de son logiciel dans *The TEXbook* (p. 13 et suivantes). À ce stade, les choses paraissent d'une simplicité exemplaire. La gourmandise venant en mangeant, comme chacun sait, l'utilisateur un peu curieux en arrive rapidement, au fil de ses expérimentations, à découvrir que tout n'est pas aussi facile qu'il avait pu le penser et qu'il existe des limites bien réelles à ce que l'on peut faire dans un environnement donné.

Nous nous proposons dans cet article d'explorer un peu ces limites, d'en fixer les origines et de voir comment les dépasser pour pouvoir tirer le meilleur parti de la puissance de TEX dans ce domaine.

1. De nouveaux principes dès 1978

Rappelons tout d'abord que TEX est avant tout un outil de composition typographique et que son objectif fondamental est le même que celui d'un

*Cet article a fait l'objet d'une présentation lors de la journée *Fontes* organisée par l'association GUTenberg le 4 décembre 1990 à l'école supérieure Estienne, à Paris.

typographe traditionnel : disposer dans des objets appellés *pages* d'autres objets nommés *lignes*, *figures*, *caractères*, etc. et ceci de la façon la plus agréable possible pour le lecteur. La vue très générale qu'avait KNUTH dès cette époque de ce que devait être la typographie informatique le conduisit à y introduire deux principes fondamentaux qui ont montré depuis toute leur pertinence et font encore aujourd'hui sa force, à savoir l'indépendance totale de la mise en page vis-à-vis du matériel de sortie d'une part, et vis-à-vis de la résolution utilisée d'autre part.

Cette séparation radicale des métriques des caractères et de leur dessin est à la base de la puissance et de l'universalité de TEX — « ça tourne sur tout » — mais aussi de nombreux déboires rencontrés par des utilisateurs de bonne foi qui se trouvent abreuvés de messages d'erreurs alors qu'ils n'avaient apparemment rien fait d'anormal (nous employons à dessein un langage moral pour refléter l'état de culpabilité où l'on se trouve souvent dans une telle situation[1] : l'analyse des rapports entre l'utilisateur et son progamme est encore à faire...)

La seule chose que fait TEX, en effet, est de produire des fichiers appelés couramment *fichiers dvi* ou simplement `.dvi`[2] qui fournissent une description de la façon dont sont composées les pages finales. On a souvent coutume de dire que ces fichiers ne contiennent que des boîtes bien dimensionnées ; la réalité est un peu plus complexe, comme on le verra, mais l'image est somme toute assez bonne.

La conséquence logique de ceci est que TEX n'a besoin, pour pouvoir travailler, que des dimensions que doivent occuper les différents objets qu'il doit mettre en place, dimensions exprimées en unités standard que l'on trouvera dans le tableau 7. En ce qui concerne les caractères qui sont les objets que TEX juxtapose pour constituer les mots et lignes de ses pages, leurs dimensions ou *métriques* sont regroupées dans des fichiers particuliers, les *fichiers tfm*, les dessins des caractères étant eux-mêmes renvoyés dans des fichiers de divers types (*fichiers gf*, *fichiers pk*, etc.) utilisés uniquement

[1] Nous sommes ainsi abreuvé, lors du contrôle de ce texte sur écran, des messages suivants :
```
texx: cannot get font cmcsc10 scaled 800:
No such file or directory
texx: (wanted, e.g., "./cmcsc10.240pk")
texx: 1 missing font prevents output (sorry)
```
alors que nous n'avons jamais demandé explicitement une telle police. Nous savons cependant où est l'erreur et ce qu'il faudrait faire pour y remédier : créer le fichier des dessins de caractères `cmcsc10.240pk` et le mettre dans un endroit approprié pour que la visionneuse (`texx`) le trouve.

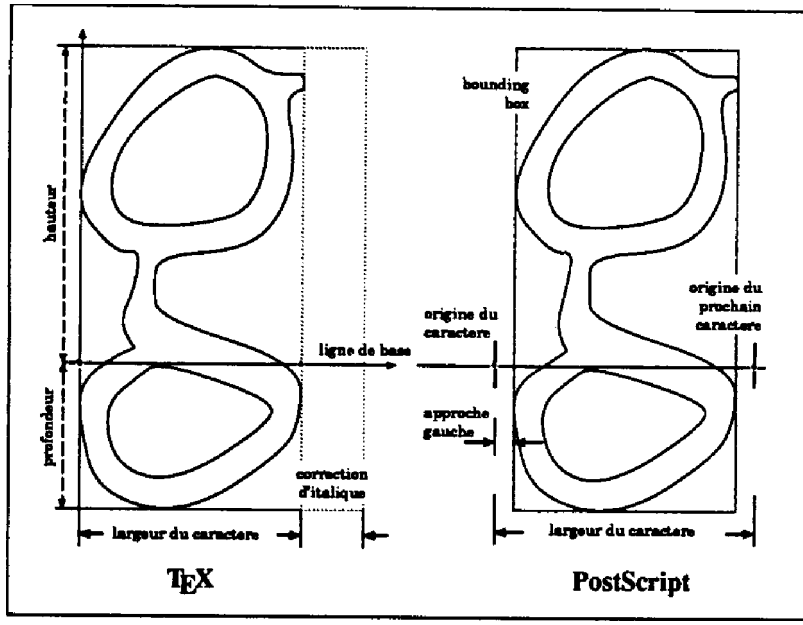
[2] On se reportera au tableau 6 pour avoir la signification des différentes extensions des noms de fichiers en TEX.

par les logiciels de sortie (pilotes d'écran ou d'imprimante, traducteurs vers d'autres langages, etc.), les noms de fichier (sans extension) ayant les mêmes racines dans tous les cas. Ainsi, pour la famille *Computer Modern*, en style romain et corps 10, on aura les fichiers `cmr10.mf` produit par METAFONT, `cmr10.tfm` pour les métriques et `cmr10.300pk`, `cmr10.329pk`, `cmr10.360pk` et `cmr10.432pk`...pour les dessins des caractères, entre autres.

2. Le modèle de fonte

Le modèle de fonte défini par TEX est relativement simple, comparé à celui utilisé par PostScript [Andre et Bur 90] comme on peut le voir sur la figure 1, ou plus encore autrefois par Mint (le déplacement du point de base est toujours horizontal par exemple, ce qui n'est pas le cas pour les deux autres). La métrique comporte, pour chaque caractère, sa *chasse*, sa *hauteur*

Figure 1. Modèles de caractères de TEX et PostScript



et sa *profondeur* par rapport à la *ligne de base*, la *correction d'italique* à appliquer éventuellement ainsi que des indications pour les ligatures ou les crénages qu'il peut avoir avec les caractères suivants. On présentera un peu plus loin la façon dont ceci apparait dans les fichiers `.tfm`

D'autres caractéristiques sont communes à tous les caractères: ce sont respectivement la *taille* pour laquelle la fonte a été dessinée (*designsize*), l'*inclinaison* (*slant*) qui sert pour le placement des accents, l'*espace* inter-mot (*space*), la *dilatation* maximale de cet espace (*shrink*) ou son *rétrécissement* (*stretch*) utilisés tous deux pour la justification des lignes dans un paragraphe, la *hauteur* du « x » (*xheight*), la *largeur* du cadratin (*quad*) et l'*espace supplémentaire* après un point (*extraspace*, c'est une habitude américaine). Voir [Andre89] à ce sujet. On peut trouver tous ces renseignements dans les fichiers `.tfm` ou dans les fichiers `.pl` qui sont une traduction des fichiers `.tfm` à usage des pauvres humains incapables de décoder sans effort les codes « binaires » des `.tfm`.

Le fichier `cmr10.tfm` fournit ainsi, après conversion, les indications suivantes pour la police `cmr10`:

```
(FAMILY CMR)
((DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 11374260171)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.333334)
   (STRETCH R 0.166667)
   (SHRINK R 0.111112)
   (XHEIGHT R 0.430555)
   (QUAD R 1.000003)
   (EXTRASPACE R 0.111112)
   )
```

où l'on voit par exemple qu'une *espace* vaut 3,33334 pt et la hauteur des minuscules 4,30555 pt, ceci pour la taille nominale de 10,0 pt. Pour les caractéristiques particulières de chaque caractère, on voit sur l'exemple du « y » pour cette police que ce caractère a une chasse de 5,27781 pt, une hauteur, une profondeur et une valeur de correction d'italique et que les valeurs de crénage avec « o », « e », « a » ainsi qu'avec les caractères de rang (octal) 54 et 56 (c'est-à-dire virgule et point) sont négatives.

```
(CHARACTER C y
   (CHARWD R 0.527781)
   (CHARHT R 0.430555)
   (CHARDP R 0.194445)
   (CHARIC R 0.013888)
```

```
(COMMENT
   (KRN C o R -0.027779)
   (KRN C e R -0.027779)
   (KRN C a R -0.027779)
   (KRN O 56 R -0.083334)
   (KRN O 54 R -0.083334)
   )
)
```

Il n'y a pas de ligature avec « y » dans cette police. La commande pour générer ces fichiers est :

`tftopl <tfm-file> <pl-file>`

Comme la commande inverse `pltotf` existe, rien ne nous interdit de modifier les métriques de vos fontes, sous votre entière responsabilité bien sûr !

3. Une utilisation aisée...

Pour l'utilisateur de TEX[3], le passage d'une police à une autre s'accomplit à l'aide de commandes simples. La première de toutes est d'ailleurs l'absence de toute commande : dans ce cas, TEX utilise par défaut la police `cmr` dans le corps 10 correspondant au fichier de métriques `cmr10.tfm`. Le changement de police se fait, à tout instant, à l'aide des commandes `\rm`, `\sl`, `\it`, `\tt` et `\bf`. En fait, ces commandes changent uniquement le style, mais non la taille des lettres (on obtient respectivement les styles Romain, *Romain penché*, *Italique*, `Machine-à-écrire` et **Gras**). Il n'y a pas de commande en TEX (à la différence de LATEX) pour changer uniquement le corps ; cette modification passe par la désignation explicite d'une autre police, que celle-ci existe effectivement dans la taille voulue (par exemple `\sevenrm`, commande qui fait référence au fichier `cmr7.tfm` pour du Romain en corps 7) ou que cette police (c'est à dire, pour TEX **uniquement la métrique** de la police) ait été créée par agrandissement à partir d'un fichier de métriques existant avec la commande `scaled` : vous pouvez ainsi par exemple utiliser la commande `\twelveandhalf` si vous avez pris la précaution de « créer » cette police par la commande `\font\twelveandhalf=cmr10 scaled 1250` et si vous avez l'idée de travailler en 12,5 points.

En fait, si on regarde les choses d'un peu plus près, on s'aperçoit que la méthode fondamentale est la seconde ; les commandes des cinq styles présentées au début ne sont en fait que des macro-instructions en TEX

[3] Il en est de même en LATEX, même si les commandes sont un peu différentes

(définies dans le fichier `plain.tex`), ayant pour effet final l'appel d'une autre police[4]. Dans le cas de notre « police » `\twelveandhalf`, nous avons parlé de « création » en un sens virtuel : au moment de composer lignes et pages, TEX calculera les dimensions des caractères de cette police à partir des métriques de la police `cmr10.tfm`. Les dessins correspondants n'existent pas pour autant, mais ça, vous ne vous en apercevrez que plus tard, quand vous voudrez passer aux actes, c'est-à-dire « sortir » sur un périphérique quelconque, écran ou imprimante.

4. ...en usage courant...

4.1. Les polices standard

Quelles sont donc les polices existant de façon standard dans toute implémentation de TEX utilisant le format `plain.fmt`? Il y en a en tout et pour tout 16 directemenent utilisables, c'est-à-dire que l'utilisateur peut appeler par leur nom (tableau 1), et 25 supplémentaires « préchargées », qui nécessitent une déclaration de nom pour pouvoir être utilisées (tableau 2).

Table 1. Polices chargées par le format `plain.tex`

cmr10	cmr7	cmr5	cmmi10	cmmi7	cmmi5	cmex10	cmsl10
`\tenrm`	`\sevenrm`	`\fiverm`	`\teni`	`\seveni`	`\fivei`	`\tenex`	`\tensl`
cmbx10	cmbx7	cmbx5	cmsy10	cmsy7	cmsy5	cmti10	cmtt10
`\tenbf`	`\sevenbf`	`\fivebf`	`\tensy`	`\sevensy`	`\fivesy`	`\tenit`	`\tentt`

Les noms d'appel des polices chargées sont écrits dans le tableau 1 en style machine-à-écrire dans le corps 8, ce qui a nécessité le baptême de cette fonte par la commande

```
\font\stt=cmtt8        (en TeX)
\newfont{\stt}{cmtt8}        (en LaTeX)
```

puisqu'il s'agit d'une police préchargée, puis le codage de chaque nom dans le tableau avec `{\stt \char'\\tenrm}` .

On peut se poser également la question de savoir si l'existence d'une même police en plusieurs corps différents est justifiée, puisque les mécanismes

[4] Pour rentrer dans le détail des *familles* de fontes, on se reportera au livre de R. SÉROUL, *Le petit livre de TEX, p. 54 et suivantes* pour des explications plus détaillées sur ces mécanismes.

Table 2. Polices préchargées par le format `plain.tex`

cmr9	cmr8	cmr6	cmmi9	cmmi8	cmmi6
cmbx9	cmbx8	cmbx6	cmsy9	cmsy8	cmsy6
cmsl9	cmsl8	cmtt9	cmtt8	cmti9	cmti8
cmss10	cmssq8	cmssi10	cmssqi8		
cmr7 scaled 2074		cmtt10 scaled 1440		cmssbx10 scaled 1440	

d'agrandissement de TEX et METAFONT permettent de générer tous les corps nécessaires à partir d'un même fichier de départ ; la raison en est simple : le dessin des lettres (et leurs métriques) varie en fonction de leur taille nominale :

voici du romain de taille nominale 10 points

et du romain 5 points agrandi à 10 points

L'épaisseur, la chasse, l'espacement ne sont pas les mêmes d'un caractère à l'autre. Rappelez vous que la taille nominale est celle qui (en principe !) est la plus agréable et la plus lisible.

4.2. Les commandes de taille

Le mécanisme d'agrandissement peut se faire de trois façons différentes en TEX :

```
\font\fiverm=cmr5 at 10pt
\font\fiverm=cmr5 scaled 2000
\font\fiverm=cmr5 \magstep4
```

Dans les trois cas, TEX a besoin du fichier `cmr5.tfm` et fournit des résultats très voisins ; les pilotes d'imprimantes ou d'écran se plaignent sans doute un peu plus s'ils ne trouvent pas les bons dessins correspondants ; on trouvera dans le tableau 3 les relations entre les diverses formes d'agrandissements demandés, les corps résultants et les suffixes des polices `.pk` nécessaires pour une police de corps nominal 10 `pt` et des résolutions d'imprimante de 300 et 406 `dpi`. Les agrandissements indiqués dans ce tableau sont ceux généralement utilisés dans la générations des fichiers des dessins des caractères ; vous avez donc de bonnes chances de les retrouver sur votre site. Ces agrandissements (sauf `\magstehalf`) sont en progression géométrique de raison 1,2.

En LATEX, les commandes de style et de taille sont séparées ; ces dernières

Table 3. Correspondances entre agrandissements et fichiers pk

magstep	agrand.	corps (approx.)	suffixes .pk	
			300 dpi	406 dpi
0	1000	10pt	300	406
half	1095	11pt	329	445
1	1200	12pt	360	497
2	1440	14pt	432	585
3	1728	17pt	518	702
4	2074	20pt	622	842
5	2488	25pt	746	1010
	2986	30pt	895	
	3583	36pt	1075	
	4300	43pt	1290	
	5160	52pt	1548	
les agrandissements de 2986 à 5160 sont utilisés par SLiTEX				

s'obtiennent avec des instructions comme `\tiny` pour obtenir de tout petits caractères ou `\Large` pour de plus gros. On trouvera la correspondance entre ces commandes et les corps obtenus dans le tableau 4 ; il faut remarquer que ces corps dépendent de l'option de *taille* du style choisi ; par défaut, la taille est 10 pt ; elle peur être mise à 11 pt ou 12 pt. Comme TEX, LATEX fait une distinction entre polices chargées et polices préchargées ; pour ces dernières, la commande d'initialisation est

`\newfont{`*commande*`}{`*nom-de-police*`}`

où *commande* est la commande d'appel de la nouvelle police (`\stt` par exemple) et *nom-de-police* est le nom du fichier, sans suffixe, que vous voulez charger (`\cmtt8` si vous voulez utiliser le style machine-à-écrire en corps 8). Le tableau 5 indique quelles sont les polices disponibles de façon standard, celles qui, préchargées, peuvent être utilisées après déclaration et celles qui ne sont pas disponibles. Il y a cependant certaines incohérences dans la gestion des polices sous-jacente de LATEX : ainsi `\Large\it` *donne de l'italique en grande taille* alors que `\it\Large` donne du romain !

Pour conclure ce bref tour d'horizon des polices disponibles, il faut retenir que vous ne pouvez pas vous affranchir de deux limites : la première est l'existence d'un fichier de métriques (`.tfm`) au départ, la deuxième est l'existence (ou la création pour les plus hardis) des fichiers de dessins (en

Table 4. Les corps utilisés par LaTeX

commande	taille par défaut 10 pt	11 pt	12 pt
`\tiny`	5	6	8
`\scriptsize`	7	8	8
`\footnotesize`	8	9	10
`\small`	9	10	11
`\normalsize`	10	11	12
`\large`	12	12	14
`\Large`	14	14	17
`\LARGE`	17	17	20
`\huge`	20	20	25
`\Huge`	25	25	25

général `.pk`) convenant à la fois à l'agrandissement défini lors de la création et à la résolution de la sortie utilisée. Une autre possibilité existe cependant : elle consiste à sortir partiellement du monde TEX en utilisant des polices dans un autre format ; TeX ne joue plus alors que le rôle d'un logiciel de mise en page très élaboré ; la sortie sur imprimante (ou photocomposeuse) fait appel à d'autres polices, PostScript par exemple. Il faut cependant, pour que TEX puisse remplir son rôle, qu'il dispose des métriques de ces polices. Un utilitaire, `afmtotfm` dans le cas de PostScript, se charge de cela : à partir des métriques PostScript, il construit un fichier de même nom et de suffixe `.tfm` qui contient les métriques au format requis par TEX. Il existe ensuite des myriades de programmes (GUTenberg distribue `dvips` de Tom ROCKIKI avec ses bandes SUN) qui se chargent de traduire en bon PostScript le fichier dvi ainsi obtenu.

En résumé, une police pour TEX, c'est un *nom*, une *taille* et un *agrandissement* ; si vous voulez tirer pleinement parti des polices de TEX, il vous faut connaître les *noms* des polices désirées, noms souvent mystérieux (à quoi peut bien ressembler du `cmitt10` ou du `cmssdc10` ?), et surtout les corps et résolutions existants sur votre site. Si, de plus, vous voulez utiliser les polices résidentes de votre imprimante, vous allez devoir travailler encore un peu plus... Heureusement, les mécanismes de mise en service de nouvelles polices sont suffisament simples pour autoriser une bien plus grande variété ; ce n'est pas aussi compliqué que ça en a l'air au premier abord, et ça en vaut

Table 5. Les polices utilisables avec LaTeX

	style						
taille	\rm	\it	\bf	\sl	\sf	\sc	\tt
5 pt	S	P	–	–	–	–	–
6 pt	S	P	–	–	–	–	–
7 pt	S	P	–	–	–	–	–
8 pt	S	P	P	P	P	P	P
9 pt	S	S	P	P	P	P	P
10 pt	S	S	S	S	S	P	S
11 pt	S	S	S	S	S	P	S
12 pt	S	S	S	S	S	P	S
14 pt	S	S	S	P	P	P	P
17 pt	S	S	S	P	P	P	P
20 pt	S	P	P	P	P	P	P
25 pt	S	–	P	–	–	–	–

S : Standard
P : Préchargée, à nommer
– : Non disponible

souvent la peine.

5. ... qui cache une mécanique complexe

Derrière cette simplicité toute relative se cache une machinerie complexe qui concerne TeX, LaTeX, les pilotes d'imprimantes ou d'écran, voire METAFONT ou PostScript et leurs divers outils. Commençons d'abord par un peu de sémantique sur les noms et les types des divers fichiers utilisés ; le tableau 6 donne les principales extensions de ces fichiers :

5.1. L'origine

Toutes les fontes de TeX[5] ont été produites par METAFONT, système crée lui aussi par D.E. KNUTH. Ce système permet de générer des familles de polices et de fournir en sortie deux types de fichiers, l'un pour les métriques

[5]Enfin presque toutes ; cite[cousquer] nous explique comment créer des fontes chinoises à partir des seuls dessins (*bit-map*) des caractères.

Table 6. Principaux fichiers utilisés par TEX

`.mf`	metafont	fichier source pour Metafont
`.gf`	generic file	bit-map des caractères d'une fonte
`.tfm`	tex font metric	métriques d'une fonte
`.pk`	packed file	bit-map des caractères
`.pxl`	pixel file	bit-map des caractères
`.tex`		source pour TEX
`.fmt`	format	fichier format (TEX ou LATEX) en binaire
`.dvi`	device independant	fichier produit par TEX
`.log`		compte rendu d'exécution (ou `.lis`)
`.pl`	property list	forme lisible des `.tfm`
`.vpl`	virtual property list	forme lisible des `.vf` et `.tfm`
`.vf`	virtual font	description binaire d'une fonte virtuelle

(fichiers `.tfm`), les autres pour les dessins créés pour une résolution donnée (fichiers `.gf`, `.pk` ou `.pxl`). En entrée, chaque caractère est décrit dans un langage (défini dans *The* METAFONT *book*) assez ardu à utiliser correctement ; en gros ce sont les contours des dessins qui sont décrit et stockés dans un fichier `.mf`.

Dans sa version d'origine, TEX82, les polices `cm`[6] comprenaient 128 caractères par fonte ; avec la version 3.0 qui travaille sur 8 bits, elles pourront en avoir jusqu'à 256. Dans la version 7 bits, qui est encore la version la plus répandue, comme dans les fontes qui l'accompagnent, les caractères accentués sont obtenus par crénage (c'est le rôle du paramètre global *slant*[7] de fournir à TEX le décalage relatif de l'accent par rapport à sa position « normale » dans le cas d'un style penché par exemple). Les futures polices auront, elles, toutes les lettres accentuées nécessaires au moins pour les langues européennes.

Un autre inconvénient des polices actuelles va également disparaître : il s'agit du codage interne des symboles ; en effet, comme on peut le voir dans *The TEXbook*, pp. 427 et suivantes, ce codage suit le code ASCII pour les caractères dits « imprimables » dans les polices `cmtt`, les caractères de contrôle étant occupés par des lettres grecques et différents signes diacritiques. Pour les autres polices, par contre, sept codes changent

[6] `cm` pour computer modern

[7] Voir dans la section *les fichiers pxl* ci après une brève description du rôle des divers paramètres.

d'affectation : ce sont ceux des symboles « < », « > », « \ », « _ », « { », « | », « } » remplacés respectivement par « ¡ », « ¿ », « “ », « ´ », « – », « — », « ” ». De plus, si les lettres grecques sont toujours aux mêmes emplacements, d'autres codes de contrôle (et même le code 32, ESPACE) sont occupés par les dessins des ligatures de la police, différents accents et signes diacritiques, quelques lettres spécifiques à certaines langues (ß, æ, œ, ø, Æ, etc.). Tout ceci est assez fâcheux du point de vue de la cohérence ! Pour les mathématiques, la situation est encore plus compliquée puisqu'il y a mélange de lettres latines, grecques et de symboles ; les fontes utilisées dans ce mode sont les `cmmi`, `cmsy` et `cmex`. En gros, pour la première (`cmmi`), seules restent à leur place les lettres majuscules et minuscules ; les autres positions sont utilisées par des lettres grecques majuscules et minuscules ou des symboles. Avec la deuxième (`cmsy`), il ne reste plus que les majuscules calligraphiées, tout le reste contenant des symboles divers. La troisième (`cmex`) est utilisée exclusivement par des symboles (ou morceaux de symboles pour ceux qui sont extensibles).

Les polices qui vont accompagner désormais TEX 3.0 suivront une autre philosophie ; le codage reprendra les normes dites *ISO-latin 1* et *ISO-latin 2*[8] en ajoutant dans les positions « libres » les accents, la ponctuation ainsi que quelques rares caractères manquants (œ a ainsi été « oublié » par l'ISO !). Un tel codage qui concerne avant tout les pays européens a été approuvé lors du congrès européen de TeX en 1990 ; certaines questions de compatibilité avec les règles de codage préconisées pour PostScript ou TrueType sont peut-être encore à discuter : il s'agit de la non-occupation des 32 premières positions et de la 128ème (recommandations pour PostScript, mais exigences pour TrueType). Une fois ces questions réglées et les polices correspondantes fabriquées, un avenir sans nuages s'ouvrira : plus de problèmes de césure des mots accentués (à condition de créer les tables de césure...), écriture de toutes les langues (européennes...), échange internationnal aisé de documents, etc, etc. Rêvons...

Outre la cohérence retrouvée et les lacunes comblées, il y a dans la nouvelle définition des fontes de la version 3.0[9] quelques modifications dignes d'intérêt (de nouvelles possibilités sont ainsi offertes pour les ligatures dont le nombre passe de 256 à plus de 32 500 par police) et une véritable nouveauté, le caractère spécial, *cwm*[10], qui fournira un mécanisme pour changer la forme

[8] Norme ISO-8859, voir aussi la norme ISO-6937 ; cf [Marti90].

[9] ou plus exactement des *fontes créées par* METAFONT 2.0 *et utilisées par* TEX 3.0

[10] *cwm* pour compound word mark.

de certaines lettres dans des conditions particulières, aux extrémités de mots ou dans un environnement donné par exemple ; c'est un peu le zéro des fontes.

Pour LaTeX, dans ses versions 2.10 et la future 3.0, il est prévu un nouveau schéma de sélection des fontes ; alors que la grille actuelle de sélection est organisée selon deux critères, le *style* et la *taille*, avec quelques incohérences comme on l'a vu, la future grille prendra en compte quatre variables :

1. la *famille* (`cmr`, `cmss`, `cmtt`, respectivement romain, sans-sérif, machine-à-écrire...),
2. la *série* (medium, bold...),
3. la *forme* (italique, penché, petites capitales, souligné...)
4. et enfin la *taille* (10 `pt`, 11 `pt`...).

Le mécanisme de sélection comprendra des instructions du type

```
\family{cmr}
\series{bf}
\shape{it}
\size{9}{11pt}
```

puis l'utilisation de commandes `\selectfont`... pour la mise en service de nouvelles polices. La sélection d'une seule caractéristique sera possible en conservant les autres inchangées par des commandes du type

`\rm` (pour du romain)
`\bf` (pour du gras)
`\sl` (pour du penché)
`\tiny` (pour la taille, comme auparavant)

De nouvelles macros comme `\rmdefault`, `\bfdefault` ou `\itdefault` seront disponibles. Ainsi, pour composer tout un document en sans sérif, il suffira de mettre dans le préambule

```
\renewcommand{\rmdefault}{cmss}
\renewcommand{\itdefault}{sl}
```

et le tour sera joué...

La définition des fontes externes associées sera elle aussi simplifiée ; ainsi, pour définir la combinaison d'une famille (`cmss`) avec une série (`bx`) et une forme (`n`) pour les tailles de 5 à 25, on utilisera la macro

```
\new@fontshape{cmss}{bx}{n}{
<5>cmssbx5%
<10>cmssbx10%
...
```

```
<25>cmssbx10 at 24.88pt%}
```

6. Les fontes virtuelles

C'est un concept nouveau introduit par D.E. KNUTH dans TEX 3.0 pour permettre de s'adapter à des fontes d'origine quelconque ; il n'a pas encore reçu beaucoup d'applications, mais il faut quand même citer le pilote `dvips` de T. ROCKIKI dans ses dernières versions.

Une *fonte virtuelle* existe logiquement, mais pas physiquement ; cela n'est pas génant pour TEX qui n'a pas besoin de savoir d'où viennent réellement les caractères. Les pilotes trouvent ensuite dans les fichiers `.vf` quels caractères réels correspondent aux caractères que TEX imaginait être présents lors de la compilation. Ces caractères réels peuvent être déplacés, agrandis et combinés avec des caractères d'autres fontes. Une fonte virtuelle est donc en quelque sorte un dispositif d'indexation pour les caractères : elle peut même utiliser des caractères appartenant à une fonte virtuelle, y compris elle-même !

Les fontes virtuelles peuvent être utilisées pour faire de l'épreuvage quand les fontes de l'imprimante finale n'existent pas sur l'imprimante ou l'écran d'épreuvage. Elles existent sous deux formes : les fichiers `.vf` qui sont codés de façon similaire aux fichiers `.pk` et `.dvi` et les fichiers `.vpl` (*virtual property list*), similaires aux fichiers `.pl` issus des fichiers `.tfm`, mais avec des extensions. C'est la forme lisible des `.vf`

6.1. Utilisation des fontes virtuelles par les pilotes

Pour chaque fonte citée dans le fichier `.dvi`, le pilote doit

1. rechercher dans une table spéciale si la fonte est *device resident* (c'est le cas des fontes PostScript par exemple) et si oui, charger le fichier `.tfm` ; si tel n'est pas le cas, on passe à étape suivante ;
2. rechercher un fichier `.gf` ou d'un `.pk` et, s'il existe, le charger ; si il n'existe pas, on passe à étape 3 ;
3. rechercher un fichier `.vf` ; si il le trouve, cela peut conduire à une autre fonte virtuelle (le processus de recherche est récursif) ou à une fonte réelle (cas 1. ou 2.). si il n'y a pas de fichier `.gf` ou `.pk` ou `.vf`, le fichier `.tfm` est chargé et les caractères seront laissés en blanc, mais avec les

largeurs correctes ; on pourra ainsi avoir une idée de la mise en page si du moins il n'y a pas trop de caractères manquants.

6.2. Format des fichiers .vpl

C'est l'analogue des fichiers `.pl`, mais pour les fontes virtuelles. La première partie est fondamentalement la même (*nom de fonte*, *paramètres généraux*, *table des ligatures*, etc.) ; la partie spécifique aux fichiers `.vpl` se présente comme suit :

```
(MAPFONT D 0 (FONTNAME Times-Roman))
(MAPFONT D 1 (FONTNAME Symbol))
(MAPFONT D 2 (FONTNAME cmr10)(FONTAT D 20))
(CHARACTER O 0 (MAP (SELECTFONT D 1)(SETCHAR C G)))
(CHARACTER O 76 (MAP (SETCHAR O 277)))
(CHARACTER D 197 (MAP
   (PUSH)(SETCHAR C A)(POP)
   (MOVEUP R 0.937)(MOVERIGHT R 1.5)(SETCHAR O 312)))
(CHARACTER O 200 (MAP (MOVEDOWN R 2.1)(SETRULE R 1 R 8)))
(CHARACTER O 201 (MAP
   (SPECIAL ps: /SaveGray currentgray def .5 setgray)
   (SELECTFONT D 2)(SETCHAR C A)
   (SPECIAL ps: SaveGray setgray)))
```

On rencontre ainsi successivement la définition de trois noms de police pour les fontes de numéro (décimal) 0, 1 et 2, la définition du caractère de rang (octal) 0 comme étant le caractère « G » de la fonte 1 (police « `Symbol` »), le remplacement du caractère (octal) 76 par le 277 dans la même police, la création pour le caractère de rang (décimal) 197 d'un « A » majuscule accentué, l'accent étant le caractère 312 (octal), etc. Tout ceci procure une souplesse d'utilisation inconnue jusqu'à présent.

Une application intéressante des fontes virtuelles est l'épreuvage des documents : il est en effet exceptionnel de disposer sur écran, sur imprimante laser ou sur photocomposeuse des mêmes fontes aux résolutions voulues. Supposons que l'on ait un écran avec les fontes `cmr`, une laser-writer 300 dpi avec polices PostScript, une photocomposeuse avec la police Univers (sans sérif). Pour épreuver le document avant photocomposition, on utilisera la police `cmss10` sur l'écran et `Helvetica` sur la LaserWriter. La démarche à suivre sera la suivante :

1. création du fichier `univers-aps.vpl`, à partir de son fichier `.tfm` avec l'utilitaire `tftopl` ;

2. édition de ce fichier en `univers-lw.vpl` pour faire la correspondance des caractères `helvetica`, avec la métrique des `univers` ;
3. même chose pour *l'écran : `univers-ecran.vpl` avec la fonte `cmss10`;*
4. exécution de `vptovf` sur les 3 fichiers `.vpl`, ce qui fournira trois fichiers `.tfm` identiques et trois fichiers `.vf` distincts ;
5. exécution du pilote approprié, avec le fichier `.vf` correspondant.

7. Les fichiers de TEX

7.1. Dimensions et unités

TEX connait les unités de longueur couramment utilisées en typographie (voir tableau 7). Comme le dit Knuth, « les unités de mesure de TEX sont fermement fondées sur le système métrique... ».

Table 7. Unités utilisées par TeX

`in`	pouce	1 *pouce* = 2,54 *cm*
`cm`	centimètre	
`mm`	millimètre	
`pt`	point	*point typographique :* 1 `in` = 72,27 `pt`
`bp`	big point	1 `in` = 72 `bp`
`pc`	pica	1 `pc` = 12 `pt`
`dd`	didot	1157 `dd` = 1238 `pt`
`cc`	cicero	1 `cc` = 12 `dd`
`sp`	scaled point	65 536 `sp` = 1 `pt`

Pour définir une police nous avons besoin de son *corps* ; TEX appelle cette taille (qui pour lui est un nombre réel compris entre 1,0 et 256,0) la *taille nominale* (`designsize`) exprimée en points ; il utilise de façon interne une autre unité : le `FIX` ; 1 `FIX` est égal au corps nominal multiplié par 2^{-20}. Le `FIX` est ainsi une unité relative ; c'est dans cette unité que sont exprimées de façon interne les dimensions diverses d'un caractère telles que la chasse, la profondeur des jambages, la hauteur au dessus de la ligne de base, etc. C'est elle aussi qui règlemente les positionnements absolus ou relatifs des diverses composantes d'un fichier `.dvi` dans la page.

Dans les descriptions suivantes relatives aux fichiers de TEX, un *mot* indique toujours une suite de quatre octets. On trouve des données codées sur un mot, un demi-mot, un octet ou une fraction d'octet. Ces descriptions sont intéressantes à plus d'un titre : outre la satisfaction de la curiosité légitime de l'utilisateur, elles permettent de mieux comprendre le fonctionnement de certains mécanismes comme les ligatures, la sélection et le nombre des fontes disponibles, etc.

Suivant en cela les règles classiques de la dissertation, les fichiers sont divisés en trois grandes parties, un *préambule* qui donne des informations générales sur le fichier lui-même, un *corps* et une conclusion qui récapitule un certains nombre de données essentielles. Il y a dans ces fichiers deux types d'informations : des *données* comme les dimensions, déplacements, etc., fournies explicitement ou par indexation dans une table interne, et des instructions dans des langages de commande propres comme nous allons le voir pour les fichiers `.dvi`, `.tfm` ou `.gf`.

7.2. Les fichiers `.dvi`

Ce sont les fichiers résultant de l'action de TEX sur le texte saisi par l'utilisateur ; l'indépendance dont ils se réclament vis-à-vis de la sortie est pratiquement totale, compte-tenu des quelques remarques que l'on fera à la fin sur le mécanisme de désignation et de sélection des polices.

Il s'agit avant tout d'un flot d'octets qui peuvent être considérés comme les commandes d'un petit langage du type *assembleur*, commandes pour lesquelles le premier octet est le code de l'opération à effectuer ; nous le désignerons dans la suite par *opcode n* où *n* sera sa valeur décimale ; cet octet est suivi de zéro, un ou plusieurs autres octets, selon les cas, qui sont les paramètre de la commande considérée. La structure générale du fichier est constituée d'un *préambule* suivi d'une ou plusieurs *pages* et se termine par une *conclusion*. Plusieurs paramètres des commandes sont des *pointeurs* qui donnent une adresse (rang d'un octet) dans le fichier.

7.2.1. Préambule

Le préambule est constitué d'une commande unique, *pre*, qui définit les dimensions utilisées dans le fichier ; elle a la forme

pre : 247 *i*[1] *num*[4] *den*[4] *mag*[4] *k*[1] *x*[*k*]

où

i est un identificateur de type pour les fichiers `.dvi` (actuellement, l'entier 2) ;

num [4] et *den* définissent les unités de mesure du fichier : ils constituent le numérateur et le dénominateur d'une fraction par laquelle il faut multiplier toutes les dimensions qui apparaissent dans le fichier pour les obtenir comme des multiples de 10^{-7} mètres. TEX82 leur donne les valeurs *num*= 25400000 et *den*= 473628672 ($= 7226 \times 2^{16}$) pour exprimer le fait qu'il y a 7227 `pt` dans 1 `cm`.

mag est l'agrandissement global du document telle qu'elle a été fixée par TEX. C'est la seule chose qui change dans le `.dvi` quand vous recompilez un document en changeant seulement le paramètre de la commande `\magnification`. Certain utilitaires travaillant à partir d'un `.dvi` peuvent modifier cette valeur.

k et *x* peuvent être utilisés pour insérer un commentaire dans le préambule.

7.2.2. Page

Chaque *page* commence par une commande *bop* et se termine par une commande *eop* ; une commande *eop* ne peut être suivie que par une commande nulle *nop*, une autre commande *bop*, une commande de déclaration de fonte *fnt_def* ou la commande *post* qui introduit la dernière partie du fichier.

Quand un utilitaire lit un fichier `.dvi`, il doit conserver la trace de plusieurs paramètres :

1. la valeur f de la police courante (c'est un entier), valeur qui ne peut être modifiée que par les commandes *fnt* et *fnt_num*,
2. les coordonnées h et v de la position du point courant ($(0,0)$ pour le coin supérieur gauche de la feuille),
3. quatre nombres représentant des espacements horizontaux, w et x, et verticaux, y et z (comptés positivement vers le bas).

Une pile contenant les six valeurs h, v, w, x, y et z doit pouvoir être gérée, le fichier `.dvi` possédant des instruction d'empilage et de dépilage de ces six valeurs pour les conserver en mémoire. Il n'y a pas, par contre, de mécanisme analogue pour les fontes qui sont toujours appelées directement et ne peuvent pas être empilées.

Les principales commandes sont :

set_char_0 : opcode 0 à *set_char_127 : opcode* 127
imprimer le caractère de code *opcode* n ; la valeur de h est incrémentée de la largeur du caractère après cette commande.

set_char_128 : opcode 128 $c[1]$
même chose pour les caractères de rang $128 \leq c < 256$.

...

La police numéro f est utilisée pour toutes ces impressions.

set_rule : opcode 132 $a[4]$ $b[4]$
imprimer un rectangle de noir de hauteur a et largeur b, le coin inférieur gauche étant situé en (h, v) ; rajoute ensuite b à h (b peut êtrenégatif; si h est négatif, ne fait rien).

...

put_rule : opcode 137 $a[4]$ $b[4]$
même chose, sans modification de h.

nop : opcode 138
ne rien faire...

bop : opcode 139 $c_0[4]$ $c_1[4] \ldots c_9[4]$ $p[4]$
début de page ; les six valeurs h, v, w, x, y et z sont mises à 0 et la pile doit être vidée. Le numéro de police f est indéfini. Les dix paramètres c_i contiennent les valeurs des compteurs `\count0` à `\count9` au moment où la commande de génération de page `\shipout` a été rencontrée. p est un pointeur vers le début de la page précédente, ce qui peut autoriser le parcours rapide d'un fichier `.dvi` à partir de la dernière page.

eop : opcode 140
fin de page.

push : opcode 141
empiler les six valeurs h, v, w, x, y et z.

pop : opcode 142
dépiler les six valeurs h, v, w, x, y et z.

Viennent ensuite 28 commandes de déplacements et de modification ou d'affectation des six valeurs précédentes, permettant de nombreuses combinaisons ; cette façon de faire a été choisie pour des raisons de compacité. La première est ainsi :

right1 : opcode 143 $b[1]$
déplacement à droite de b unités et affectation à b de la valeur $b + h$ (b peut être négatif, ce qui entraine un déplacement vers la gauche).

fnt_num_0 : opcode 171 à *fnt_num_63 : opcode* 234
mise en service ($f \leftarrow i$) de la police numéro i ($0 \leq i < 63$) qui doit avoir préalablement été définie par une commande *fnt_def*.

fnt1 : opcode 171 $k[1]$
même chose pour les police de numéro compris entre 64 et 255.

Il existe aussi dans la définition du langage des fichiers `.dvi` des commandes *fnt2*, *fnt3* et *fnt4* qui permettraient de gérer respectivement jusqu'à 65 536, $2^{24} - 1$ et *beaucoup*, *beaucoup*[11] de polices, mais TEX ne les utilise pas ; on est donc limité à 256 polices différentes dans un fichier `.dvi`.

7.2.3. Conclusion

Restent enfin les commandes de définition des polices ; elles sont de la forme
fnt_def1 : opcode 243 $k[1]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$
où

- k est le numéro de la fonte que l'on définit,
- c est le code de contrôle que TEX a trouvé dans le fichier `.tfm` lorsqu'il a créé le fichier `.dvi` (ce qui peut poser des problèmes de compatibilité et être un frein à l'échange de fichiers `.dvi` si ce code n'est pas nul ou s'ils ne coïncident pas entre les deux sites).
- s est un facteur d'échelle appliqué à toutes les largeurs de caractère dans la fonte de numéro k,
- d est la taille nominale, sans correction de `\magnification`,
- le reste donne le nom du répertoire et le nom de la fonte sous la forme d'une chaîne de $a + l$ caractères ASCII, a pour le nom du répertoire et l pour celui de la fonte.

Il reste enfin les commandes de début de la dernière partie qui peut contenir, elle aussi, des commandes de définition de fonte, mais récapitule surtout des paramètres relatifs au fichier tout entier : hauteur plus profondeur de la plus grande page, plus grande largeur, pointeur vers le *début-de-page* *bop* de la dernière page, hauteur maximum de la pile, etc.

7.3. Les fichiers `.tfm`

On peut considérer qu'un fichier `.tfm` est formé de deux parties : un *en-tête* de six mots pour des informations générales suivi de dix tableaux de

[11] Exactement $2^{32} - 1$ si vous voulez tout savoir.

valeurs diverses.

7.3.1. En-tête

Les six premiers mots correspondent à douze paramètres entiers ; ils donnent des informations sur le fichier lui-même et sur les éléments qui le composent ; ces douze valeurs sont :

`lf` : longueur du fichier en mots
`lh` : longueur du tableau en-tête (`header`) en mots
`bc` : index du premier caractère disponible dans la fonte
`ec` : index du dernier caractère disponible
`nw` : nombre de mots dans le tableau `width`
`nh` : nombre de mots dans le tableau `height`
`nd` : nombre de mots dans le tableau `depth`
`ni` : nombre de mots dans le tableau `italic`
`nl` : nombre de mots dans le tableau `lig_kern`
`nk` : nombre de mots dans le tableau `kern`
`ne` : nombre de mots dans le tableau `exten`
`np` : nombre de paramètres de la fonte (tableau `param`)

On constate en particulier que le nombre de largeurs distinctes disponibles pour les symboles de la police est en nombre limité ; il en est de même pour les hauteurs des hampes, profondeurs des jambages ou valeurs de correction d'italique.

7.3.2. Tableaux

À la suite de cet en-tête viennent en principe les dix tableaux de données dont nous venons de voir les dimensions ; ces dix tableaux sont respectivement :

`header` : Tableau de douze mots en général[12] dont seuls les deux premiers sont réellement utiles et utilisés : `header[0]` contient un identificateur de contrôle de conformité (`checksum`) entre les fichiers `.tfm` et `.pxl` ou `.pk` ; une valeur nulle signifie qu'aucun contrôle ne sera effectué ; le deuxième mot contient la taille nominale de la police (*designsize*), taille pour laquelle elle a été conçue et dessinée. Les dix mots suivants peuvent contenir n'importe quoi : à raison de quatre caractères par mot, on peut y mettre le nom de la police, la date de création, le nom et l'âge du capitaine, etc.

[12] Sauf sur VAX qui en demande plus, sans raison ...

char_info : tableau de `bc-ec` mots, à raison d'un mot pour chacun des caractères de la police ; les quatre octets composant ces mots représentent :

- *octet 1 :* **width_index**, l'index de la largeur du caractère dans le tableau **width** (voir infra ; il peut y avoir jusqu'à 255 largeurs non-nulles dans une police).
- *octet 2 :* **height_index** et **depth_index**, les index, codés chacun sur un demi octet, soit 15 valeurs non-nulles possibles, des hauteurs et profondeurs à prendre dans les tableaux **height** et **depth**.
- *octets 3 et 4 :* les index correspondant aux tableaux **italic**, **lig_kern**, **kern** et **exten** selon le mécanisme suivant :
  - **italic_index** : 6 bits et **tag** : 2 bits dans l'octet *3*
  - **remainder** : 8 bits dans l'octet **4** dont l'interprétation dépend de la valeur de **tag**.

  La valeur **italic[italic_index]** de la correction italique est ajoutée à la largeur du caractère si l'utilisateur l'a spécifiée en faisant suivre le caractère par « \/ », ou bien si on est en mode mathématique. **tag** peut prendre quatre valeurs qui indiquent comment interpréter le quatrième octet, **remainder** :

  tag = 0 : **remainder** est inutilisé.
  tag = 1 : le caractère a un programme de ligature-crénage qui débute à l'adresse **lig_kern[remainder]** dans le tableau **lig_kern**.
  tag = 2 : le caractère fait partie d'une suite de taille croissante et n'est pas le dernier : le suivant a pour index **remainder** (**remainder<bc**) dans la police.
  tag = 3 : le caractère est extensible et est composé par morceaux (il peut prendre n'importe quelle taille) ; les morceaux sont donnés dans **exten[remainder]**.

width, height, depth : trois tableaux formés de **nw**, **nh** et **nd** mots chacun et indiquant les largeurs, hauteurs et profondeurs disponibles, en points par rapport à la ligne de base, pour les boîtes contenant les dessins des caractères ; la première valeur dans ces tableaux est toujours 0.

italic : tableau de **ni** mots donnant les corrections à apporter à la chasse des caractères dans certaines conditions (formules mathématiques par exemple) ; il y a 64 valeurs possibles ; la première valeur est toujours 0.

lig_kern et **kern :** le tableau **lig_kern** contient les séquences d'instructions d'un petit langage de commande qui permet de gérer les ligatures ou

les crénages associés à un caractère et à une police ; on peut donc avoir des polices sans ou avec ligatures, sans modifications du texte source en TEX. Ce langage de commande fonctionne en conjonction avec le taleau **kern** ; chaque mot est une commande de quatre octets ayant la signification suivante :

- *octet 1 :* **stop_bit** ; indique que le mot est le dernier pas d'une séquence d'instructions du langage de commande si sa valeur est supérieure à 128.
- *octet 2 :* **next_char** ; si le caractère suivant dans le fichier source est **next_char**, effectuer l'opération indiquée par l'octet 3.
- *octet 3 :* **op_bit** ; indique une ligature si sa valeur est inférieure à 128, un crénage sinon.
- *octet 4 :* **remainder**, utilisé par le langage de commande.

Dans le cas d'une ligature, le caractère courant et **next_char** sont remplacés par le caractère de code **remainder** ; pour un crénage, un espacement de valeur **kern [remainder]** (le plus souvent négative) est inséré entre les deux caractères.

exten : tableau de données pour les caractères extensibles, composé de mots : chacun de ceux-ci est formé de quatre octets appelés dans l'ordre **top**, **mid**, **bot** et **rep**. Ce sont les codes-caractères des morceaux utilisés pour construire les symboles de taille variable.

param : c'est le dernier tableau du fichier **.tfm**. Il contient les paramètres globaux de la police, c'est-à-dire sept mots contenant :

- *mot 1 :* le décalage relatif dû à l'inclinaison, utilisé pour le placement correct des accents ; cette valeur est, en fait, un rapport entre le déplacement horizontal et le déplacement vertical : 0,25 indique un déplacement horizontal de 0,25 unité pour un déplacement vertical de 1 unité. Cette valeur n'est pas affectée par la valeur du **designsize**, ce qui n'est pas le cas des autres valeurs.
- *mot 2 :* l'espace entre les mots, en **FIX**es.
- *mot 3* et *mot 4 :* indiquent, en **FIX**es, l'expansion ou la compression possible de l'espace entre mots utilisée pour la justification des lignes.
- *mot 5 :* la hauteur des minuscules, utilisée pour le positionnement des accents.
- *mot 6 :* la valeur, en **FIX**es, du cadratin (**\quad**).

– *mot 7:* espace additionnel en fin de phrase à rajouter au deuxième paramètre (pour les américains).

7.4. Les fichiers `.pxl` ou `.pk`

Pour les fichiers `.pxl` ou `.pk`, les choses sont plus simples, puisqu'ils doivent contenir essentiellement les dessins (sous forme de matrices de points) des caractères. Les nom de ces fichiers sont de la forme *nomDSIZE.MAG* où *nom* est le nom de la police, *DSIZE* la taille nominale (`designsize`) et *MAG* un facteur d'échelle fonction de la définition de l'imprimante utilisée et de l'agrandissement auquel a été produit le fichier par rapport à la taille nominale. Nous ne donnerons que quelques brèves indications sur ce qu'ils contiennent à partir de la forme `.pxl` (forme obsolète aujourd'hui), les fichiers `.pk` en étant seulement une forme compressée. Il comprend les quatre parties suivantes :

— un identificateur de contrôle (sur un mot), `pxl_id`, dont la valeur est 1001,

— une suite de mots contenant les dessins de tous les caractères codés sous forme *bit-map* : 1 = un pixel noir, 0 = un pixel blanc,

— le répertoire de la fonte proprement dite qui contient les informations relatives à chaque caractère codées sur quatre mots; il y a autant de paquets de quatre mots qu'il y a de caractères potentiels dans la police, les caractères absents ayant leurs valeurs mises à zéro. On a ainsi dans chacun des quatre mots :

mot 1 : deux entiers de 16 bits contenant la largeur l et la hauteur h (en pixels) de la plus petite boîte englobante de la matrice de pixels.

mot 2: deux entiers de 16 bits donnant la position (en pixels) du point de référence par rapport au coin supérieur gauche de la boîte précédente: déplacement horizontal dans le premier demi-mot, compté positivement vers la droite, déplacement vertical dans le deuxième, compté positivement vers le bas.

mot 3: adresse dans le fichier du premier mot du tableau des dessins du caractère.

mot 4 : valeur de la largeur, en `FIX`es, de la boîte contenant le caractère (y compris les approches) ; on obtient ainsi le déplacement du point courant. Au facteur d'agrandissement près, cette valeur doit être la même que celle contenue dans le fichier `.tfm`.

Le répertoire de la fonte est en quelque sorte le tableau de bord et le poste d'aiguillage du fichier `.pxl` ; l'accès aux informations relatives à un caractère, à commencer par son existence dans la fonte, se fait par son intermédiaire. L'adresse du début de ce répertoire est fournie elle-même dans la quatrième et dernière partie du fichier.

— Une dernière partie formée de cinq mots :

`checksum` : code de contrôle de conformité entre le fichier `.tfm` et le fichier `.pxl` ; mis à 0 pour invalider ce contrôle.

`magnification` : valeur du facteur d'agrandissement de la fonte, multiplié par 1000.

`designsize` : corps de la fonte, en `FIX`es.

`dir_ptr` : pointeur vers le premier mot du répertoire de la fonte.

`pxl_id` : code de contrôle, égal à 1001 (voir le premier mot du fichier).

7.5. Les fichiers .gf

Ce sont les sorties « normales » fournies par METAFONT des dessins des caractères d'une fonte. À partir d'eux, on peut produire les fichiers `.pk` ou `.pxl`.

Un fichier `.gf` peut être considéré comme une suite d'octets constituants les instructions d'un petit langage de commande décrivant le remplissage d'une matrice de pixels de n lignes et m colonnes. Il est composé de trois parties : un *préambule* suivi d'une série de *caractères* et enfin une *partie finale*. Pour les instructions du langage de commande, le premier octet est le code de l'instruction elle-même ; elle peut être suivie d'un certain nombre d'octets formant les paramètres de cette instruction. Pour toute commande, on présume que le programme utilisateur conserve en mémoire les valeurs m et n respectivement de la colonne et de la ligne courante ; tous les déplacements sont calculés à partir de cette position.

La plupart des commandes sont des commandes de déplacement dans la ligne courante de la matrice appelées *paint* ; ce déplacement s'effectue en noircissant éventuellement les pixels balayés selon la valeur actuelle (*noir* ou *blanc*) d'un « pinceau » (*paint_switch*). À l'issue d'une commande *paint*, la valeur de *paint_switch* est inversée.

Le point de référence (intersection de la ligne de base avec le bord gauche de la boîte du caractère) correspond au coin inférieur gauche du pixel $(0,0)$ de la matrice de pixels ; les numéros n de ligne sont comptés positivement

vers le haut et ceux m des colonnes positivement vers la droite. Les lignes (*resp.* colonnes) en dessous (*resp.* à gauche) du point de référence ont des valeurs de n (*resp.* m) négatives. Les commandes sont telles que la valeur de la colonne courante m ne puisse jamais décroître dans un déplacement dans la ligne courante, et que le numéro n de la ligne courante ne puisse jamais croître au cours de la description d'un caractère.

7.5.1. Les commandes

Afin de fixer les idées, nous présentons quelques instructions du langage de commande ; pour chacune d'entre elles, on donne son nom symbolique en italique suivi de la valeur décimale n de l'octet qui la code sous la forme

nom : opcode n <liste des paramètres>

où <liste des paramètres> est de la forme

$param_1[n_1] \ldots param_i[n_i]$

Chaque nom de paramètre est suivi de sa longueur en octets. Les commandes présentées ici le sont dans l'ordre croissant de leur code opération (*opcode*).

paint_0 : opcode 0 commande *paint* de longueur nulle, ce qui a pour seul effet d'inverser la couleur (*noir* ou *blanc*) du « pinceau ».

paint_1 : opcode 1 à *paint_63 : opcode* 63 : peinture de pixels ;
commandes *paint* de longueur 1 à 63 pixels qui, selon la couleur du « pinceau », noircissent ou non de 1 à 63 pixels dans la ligne courante à partir de (et y compris) la colonne courante.
...(d'autres commandes du type *paint*)

boc : opcode 67 *c*[4] *p*[4] *min_m*[4] *max_m*[4] : début de caractère ;
commande de dessin pour le caractère de code *c* ; *p* est l'adresse du précédent caractère ayant le même code *modulo* 256 ($p = -1$ si il n'y en a pas) ;
min_m, *max_m*, *min_n* et *max_n* sont les rangs minimum et maximum des colonnes et des lignes de la matrice de pixels.
...

eoc : opcode 69 : fin de caractère ;
entre cette commande et les commandes *boc* suivantes ou la commande *post* finale, on ne peut trouver que des commandes vides *no_op*.

skip0 : opcode 70 : passage à la ligne suivante ; diminue de 1 le numéro de ligne, remet le numéro de colonne m à *min_m* et la couleur du pinceau *paint_switch* à *blanc*.
...

new_row_0 : opcode 74
comme *skip0*, mais la couleur du pinceau devient *noir*.

...

xxx1 : opcode 239 *k*[4] *x*[k]
commandes indéfinies (ou à définir pour le programme utilisateur) ;

...

no_op : opcode 244 : pas d'opération, commande vide.

char_loc : opcode 245 *c*[1] *dx*[4] *dy*[4] *w*[4] *p*[4]
commande qui apparaît dans la partie finale et donne l'adresse *p* dans le fichier de la commande *boc* du caractère (ou de la première des commandes du type *xxx* (...) ou *no_op* qui la précèdent ;
c est le reste (*i.e.*modulo 256) du code du caractère (les caractères de même reste sont supposés avoir les mêmes métriques) ;
w est la largeur et *dx* et *dy* les déplacements horizontal et vertical, en fractions de pixels (représentées sous une forme similaire à celle des *scaled points* avec 16 bits de partie décimale) ; dans les fichier `.dvi`, *dy* est égal à 0 : le déplacement est toujours horizontal.

...

pre : opcode 247 *i*[1] *k*[1] *x*[*k*]
i est un identificateur de type pour les fichiers `.gf` ; sa valeur est l'entier 131.
Les $k + 1$ octets suivants (l'octet *k* est un entier qui fournit la longueur du paramètre *x*) sont des commentaires et ne sont pas utilisés.

post : opcode 248 *p*[4] *ds*[4] *cs*[4] *hppp*[4] *vppp*[4] *min_m*[4] *max_m*[4] *min_n*[4] *max_n*[4] : début de la dernière partie du fichier `.gf` ;
p est l'adresse du premier octet suivant la dernière commande *eoc* de la partie *caractères* du fichier.
ds et *cs* sont respectivement les *designsize* et *checksum* du fichier `.tfm` associé,
hppp et *vppp* sont les définitions horizontale et verticale, en pixels par point (`dpi`) exprimées sous forme réelle (*i.e.* multipliées par 2^{16}), de la sortie.
min_m (*max_m*) et *min_n* (*max_n*) sont les valeurs minimales (maximales) que peuvent prendre ces variables dans toutes les commandes *boc*.

post_post : opcode 249 *q*[4] *i*[1] *c*[$\geq$ 4] :]
dernière commande de la *partie finale* du fichier `.gf` ;
q est l'adresse de la commande *post* précédente ;

i est un octet d'identification (égal à 131) et *c* est une série d'au moins 4 octets identiques (égaux à 223) qui terminent le fichier de façon à ce que sa longueur totale soit un multiple de 4.

Références bibliographiques

[Andre89] Jacques ANDRÉ, « Métriques de fontes en typographie traditionnelle » , in *Cahiers GUTenberg* n° 4, p. 10–22, 1989.

[Andre et Bur 90] Jacques ANDRÉ et Justin BUR, « Métrique de fontes PostScript » , in *Cahiers GUTenberg* n° 8, p. 29–50, 1991.

[Child88] Bart CHILD, « TEX Fonts and Suggested Magnifications » , in *TUGboat* vol. 9 (n° 2), p. 129–130, 1988.

[Cousquer90] Alain COUSQUER, « En chinois dans le TEXtefguill , in *Cahiers GUTenberg* n° 6, p. 15–24, 1990.

[Dardaillier89] Daniel DARDAILLER, « Normes et fontes » , in *Cahiers GUTenberg* n° 4, p. 2–8, 1989.

[Ferguson90] Michael J. FERGUSON, « Fontes latines européennes et TEX » , in *Cahiers GUTenberg* n° 7, p. 29–31, 1990.

[Fuchs88a] David FUCHS, « TEX Font Metric Files » , in *TUGboat* vol. 2 (n° 1), p. 12–16, 1988.

[Fuchs88b] David FUCHS, « The format of PXL files » , in *TUGboat* vol. 2 (n° 3), p. 8–12, 1988.

[Knuth82] Donald E. KNUTH, « The Concept of a Meta-Font » , in *Invisible language* vol. 16 (n° 1), p. 3-27, 1982. Reading, Massachusetts : Addison-Wesley, 1986. Traduction française : « Le concept de méta-fonte » ,???

[Knuth86] Donald E. KNUTH, *The TEXbook. Computers and Typesetting* Vol. A. Reading, Massachusetts : Addison-Wesley, 1986.

[Knuth90] Donald E. KNUTH, « Virtual Fonts : More Fun for Grand Wizards » , in *TUGboat* vol. 11 (n° 1), p. 13–23, 1990.

[Lamport86] Leslie LAMPORT, LATEX *User's guide & Reference manual*, Reading, Massachusetts : Addison-Wesley, 1986.

[Louarn et Trepos90] Philippe LOUARN et Raymond TRÉPOS, « Serveurs de fichiers et de fontes pour TEX » , in *Cahiers GUTenberg* n° 8, p. 21–28, 1991.

[Marti90] MARTI, Usa, 1990. **Trouver la ref!!!**

[Mittelbach et Schopf89] Frank MITTELBACH et Rainer SCHÖPF, « A new font selection schem for TEX macro packages — the basic macros » , in *TUGboat* vol. 10 (n° 2), p. 222–238, 1989.

[Mittelbach et Schopf90] Frank MITTELBACH et Rainer SCHÖPF, « The new font family selection — User interface to standard LATEX » , in *TUGboat* vol. 11 (n° 2), p. 297–305, 1990.

[Rokicki85] Tomas ROKICKI, « Packed (PK) font file format » , in *TUGboat* vol. 6 (n° 3), p. 115–130, 1985.

[Schrod88] Joachim SCHROD, « LaTeX Fonts and Suggested Magnifications » , in *TUGboat* vol. 9 (n° 3), p. 241–243, 1988.

[Seroul89] Raymond SEROUL, *Le petit livre de TEX*. Paris : InterEditions, 1989.

Pour tous les articles du *TUGboat* qui concernent la description des fichiers `.tfm`, `.pk`, `.pxl`, `.gf`, `.vf` et `.vpl`, on se reportera de préférence aux fichiers `.web` définissant les produits suivants : `TFtoPL`, `PKtype`, `GFtoPXL`, `GFtype`, `VFtoVP`, `VPtoVF`. On pourra obtenir ces fichiers sur les sites ftp décrits dans [Louarn et Trepos90]. Pour obtenir une sortie lisible, on les traitera par le programme `weave` qui donnera en sortie un fichier `.tex`.