Cahiers GUTenberg

LES FORMATS DE FICHIERS DVI, GF, TFM ET VF : QUE CONTIENNENT-ILS ET COMMENT LES VISUALISER ?

Denis Roegel

Cahiers GUTenberg, nº 26 (1997), p. 71-95.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1997___26_71_0>

Les formats de fichiers DVI, GF, TFM et VF : que contiennent-ils et comment les visualiser ? *

Denis ROEGEL

CRIN (Centre de Recherche en Informatique de Nancy)
Bâtiment LORIA — BP 239 — 54506 Vandœuvre-lès-Nancy cedex
`roegel@loria.fr`

Résumé. Une distribution TEX se compose d'un nombre important de programmes interagissant via des fichiers aux formats très divers. Plusieurs de ces formats sont manipulés couramment sans que nous sachions la plupart du temps ce qui s'y trouve : c'est le cas des fichiers `DVI`, `GF`, `TFM`, `VF`, etc. Nous examinons ici de manière plus détaillée le contenu de ces fichiers en utilisant des programmes de visualisation faisant partie des distributions standard.

Abstract. *A TEX distribution is made of a large number of programs interacting through files in various formats. We manipulate several of these formats commonly, without knowing their contents: it is the case for `DVI`, `GF`, `TFM`, `VF` files, etc. We examine here in a more detailed manner the contents of these files, using visualization programs being part of the standard distributions.*

1. Introduction

TEX [11] est pour beaucoup un logiciel mystérieux. Il est difficile d'avoir une compréhension fine et complète du système. Pendant longtemps, TEX était aussi réputé d'installation assez délicate[1]. La source principale de cette complexité était certainement l'enchevêtrement touffu des fonctions et fichiers en

* Ce texte correspond à un tutoriel donné à Strasbourg dans le cadre des journées Gutenberg, tutoriel auquel ont aussi participé J. André [1] et T. Bouche [2]. Ce texte est en quelque sorte un complément de [3].

1. Avec l'arrivée de distributions publiques clés en main comme `teTeX`, ceci n'est plus vrai.

tout genre, sur lequel devait se plaquer une personnalisation souvent artisanale. L'utilisateur ou l'installateur étaient ou sont encore confrontés à des erreurs ayant de nombreuses origines possibles. Lorsque ces erreurs concernent les fichiers binaires couramment manipulés[2] comme les fichiers `DVI`, `GF`, `TFM` et `VF`, il importe de connaître quelques éléments quant à leur constitution. C'est l'objet de cet article. On ne trouvera pas ici une description complète de ces formats[3], mais suffisamment d'informations pour comprendre comment ces formats un peu particuliers sont employés et pour identifier et résoudre les problèmes qui peuvent se présenter. Le format `PK` ne sera pas présenté ici car c'est un format plus complexe faisant intervenir un algorithme de compactage. Nous n'évoquerons pas non plus les formats des polices PostScript (fichiers `PFA`, `PFB`, `AFM`, etc.).

Nous commencerons par situer les différents types de fichiers puis nous les décrirons et indiquerons quels outils permettent de les manipuler. Dans la suite, nous emploierons TEX pour signifier le moteur TEX, car la description est indépendante d'un format particulier, que ce soit *plain TEX* ou *LATEX*.

2. Organisation des fichiers

La figure 1 montre les relations entre les fichiers évoqués dans cet article. Lorsque TEX est exécuté sur un fichier source, TEX lit bien sûr le fichier source et d'éventuels fichiers de même nature comme les fichiers de macros, etc. Mais TEX lit aussi certains fichiers `TFM` (*TEX Font Metrics*) qui le renseignent sur les caractéristiques dimensionnelles des polices utilisées. À l'issue de ce traitement, TEX produit un fichier `DVI`.

De manière symétrique, lorsque le programme METAFONT est appliqué à un fichier source, un fichier `GF` (*Generic Font*) ainsi qu'un fichier `TFM` sont produits. Le premier de ces fichiers comporte les dessins des caractères utiles aux pilotes (comme `xdvi`, `dvips`, etc.) et le second est utilisé par TEX.

Le format `GF` n'est pas optimisé. Il est converti en un format plus compact, le format `PK` (*Packed Font*).

Les fichiers `GF` ou `PK` peuvent être lus par les pilotes, tels que `xdvi` et `dvips`, afin de produire une représentation fidèle du résultat d'une compilation TEX.

2. C'est-à-dire produits par TEX ou METAFONT [12], ou utilisés par les pilotes. Notons que cet article n'examine pas la structure des fichiers `FMT` qui représentent des versions précompilées d'ensembles de commandes TEX.

3. Les descriptions de référence sont celles de Knuth dans les programmes manipulant ces fichiers.

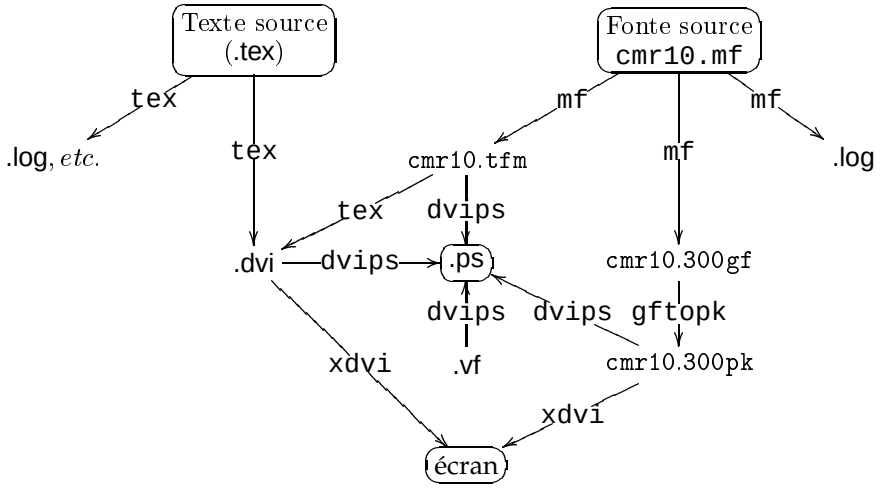

FIGURE 1 – *Organisation des fichiers*

Enfin, le format `VF` (*Virtual Font*) représente une abstraction du format `TFM`. Un fichier `VF` est utilisé par des pilotes tels que `dvips`.

Les noms des fichiers ne sont pas nécessairement les mêmes d'une distribution à l'autre, notamment en raison des contraintes de systèmes d'exploitation différents. Le fichier `cmr10.600gf` correspondant au mode d'imprimante `ljfour` pourra être appelé `cmr10.gf` dans un répertoire `600dpi` sur une autre machine. Nous avons choisi de nous concentrer sur le *contenu* des fichiers, plutôt que sur leur *habillage*. Ainsi, chaque fois que nous parlerons d'un fichier `TFM`, il sera entendu que ce fichier aura en général une extension `.tfm`.

3. Format `DVI`

Le format `DVI` est le format de sortie de TEX, mais en principe d'autres programmes[4] peuvent générer un fichier `DVI`. Ce format a été défini par David R. Fuchs en 1979. Il a l'avantage d'être très général et de nombreux post-processeurs ont été écrits afin de transformer ce format en PostScript, etc.

Un fichier `DVI` est une suite d'octets pouvant être considérés comme des commandes d'un langage machine. Le premier octet de chaque commande est le code de l'opération suivi éventuellement d'octets qui représentent des pa-

4. C'est par exemple le cas de `groff`.

ramètres de la commande. Les paramètres eux-mêmes peuvent consister en plusieurs octets. Certains paramètres représentant des déplacements peuvent dénoter des valeurs négatives en complément à deux.

Nous n'allons pas décrire toutes les instructions[5] du *langage* DVI, mais commenter de manière détaillée un fragment de fichier DVI. Nous utilisons le programme dvitype [4] pour afficher de manière intelligible le contenu[6] d'un fichier DVI. Le programme dvitype, comme d'autres programmes que nous verrons plus loin, s'apparente ainsi à un désassembleur.

Un fichier DVI consiste en un préambule (figure 2), un ensemble de pages (figure 3) et un postambule. Les pages apparaissent dans le fichier DVI dans l'ordre dans lequel elles ont été produites.

3.1. Préambule

Les fichiers au format DVI commencent tous par deux octets dont les valeurs sont 247 (commande *pre*) et 2. Ceci n'est pas reproduit par la sortie de dvitype dans la figure 2. Viennent ensuite deux entiers sur quatre octets chacun. Ces deux entiers expriment une fraction f égale à $\frac{25400000}{473628672}$ dans TEX82. f permet de passer des unités DVI (que nous noterons ud par la suite) à des unités exprimées en multiples de 10^{-7} mètres. Un ud correspondra donc à $f/10^7$ mètres, ce qui correspond à un sp (*scaled point*) dans TEX. Toutefois, on peut imaginer que d'autres programmes ayant une autre unité fondamentale que le sp produisent un fichier DVI, auquel cas la valeur de f serait différente.

Nous avons ensuite sur quatre octets le paramètre *mag* qui correspond à 1000 fois le facteur d'agrandissement d'un document. Cette valeur est positionnée en TEX par la commande \mag. Si la valeur de *mag* est différente de 1000, toutes les dimensions sont multipliées par $mag/1000$. dvitype indique aussi la correspondance entre un ud et un pixel, mais cette information ne figure pas dans le fichier DVI et c'est dvitype qui fait l'hypothèse que la définition est de 300dpi.[7] Le préambule continue par un commentaire d'au plus 255 caractères, lequel indique ici la date de la compilation du fichier TEX. Enfin, il s'achève par les définitions de polices.

Chaque police k comporte une somme de contrôle (*checksum*) sur quatre octets provenant du fichier TFM lu par le programme (en général TEX) ayant généré le fichier DVI. Cette somme sera comparée par un pilote qui lit le fichier DVI à

5. Le détail de toutes les instructions est donné dans [9] ou [16].

6. Notons que dvitype ajoute certaines informations qui ne figurent pas explicitement dans le fichier DVI.

7. Le nombre de pixels par ud est $\frac{f}{10^7} \Big/ \frac{2{,}54}{100 \cdot 300} = \frac{300}{4736286{,}72} \approx 6{,}334 \cdot 10^{-5}$.

```
Options selected:
  Starting page = *
  Maximum number of pages = 1000000
  Output level = 4 (the works)
  Resolution = 300.00000000 pixels per inch
numerator/denominator=25400000/473628672
magnification=1000;       0.00006334 pixels per DVI unit
' TeX output 1997.04.02:2101'
Postamble starts at byte 8410.
maxv=42908243, maxh=30207812, maxstackdepth=6, totalpages=4
Font 40: logosl10--loaded at size 655360 DVI units
Font 39: cmtt10 scaled 1095--loaded at size 717619 DVI units
 (this font is magnified 109%)
...
Font 7: cmr10--loaded at size 655360 DVI units
Font 6: cmr7--loaded at size 458752 DVI units
```

FIGURE 2 – *Exemple de préambule de fichier* `DVI` *(sortie* `dvitype`*)*

celle provenant du fichier `TFM` correspondant. Cette somme n'est pas affichée par `dvitype`. Chaque déclaration de police comporte son nom, le corps de référence (pour la police 7, c'est 10pt, soit 655360 ud) et un facteur d'agrandissement (1095 dans le cas de la police 39). Si le facteur d'agrandissement est différent de 1000, la taille d'utilisation de la police est indiquée en ud.

3.2. Représentation d'une page dans un fichier `DVI`

À tout moment, la position sur la page est donnée par deux nombres appelés les coordonnées horizontale h et verticale v. Ces deux nombres sont exprimés en ud et sont nuls au coin supérieur gauche de la page et croissent vers la droite et le bas. Les valeurs de w, x, y et z représentent des espacements horizontaux (w, x) et verticaux (y, z) qui seront détaillés plus loin.

La sortie de `dvitype` indique, en regard de la position par rapport au début du fichier, les différentes instructions. À la position 42, on trouve l'instruction *bop* de début de page. Cette instruction a pour effet d'initialiser le sextuplet (h, v, w, x, y, z) à $(0, 0, 0, 0, 0, 0)$. Il est possible de repérer chaque page par 10 nombres. Le choix des valeurs dépend du programme générant la sortie `DVI` et dans le cas de TEX il s'agit des valeurs de `\count0`, ..., `\count9`. Un dernier paramètre du début de page pointe vers le début de la page précédente.

La première instruction après le début de la page 1 est *down4* (qui tient sur 5 octets) et consiste à se déplacer vers le bas de 42908243 ud. La nouvelle valeur de v est indiquée. La valeur correspondante en pixels est vv (par exemple, $42908243 \cdot 0{,}00006334 \approx 2718$). La seconde instruction est *push* qui a pour ef-

fet d'empiler (donc de mémoriser) la position courante, plus exactement les valeurs des variables du sextuplet. Les valeurs mémorisées sont indiquées. L'instruction suivante se déplace à nouveau, négativement vers le bas, c'est-à-dire vers le haut. Et ainsi de suite jusqu'à la commande de l'octet 104 qui effectue un déplacement négatif vers la droite. Ces commandes de déplacement sont dues à des déplacements dans TEX, par exemple des commandes `\kern`, `\vskip`, etc, mais aussi à certains crénages.

À la position 108, la police 22 est redéfinie (elle l'était déjà dans le préambule) et elle est sélectionnée à la position 128. Suivent ensuite une série de positionnements de caractères. Ainsi, *setchar66* est la commande qui positionne le caractère de code 66 de la police courante, ce qui en l'occurrence sera un «B». La largeur de «B» est 394314 ud, valeur qui est ajoutée à la valeur courante de h. La valeur correspondante en pixels est hh (par exemple, $21379 \cdot 0{,}00006334 \approx 1$). Le mot qui est formé des octets 129 à 137 est «Brouillon». La commande qui suit, *w3* effectue un déplacement de 185688 ud à droite et correspond à l'espace qui suit le mot. De plus, cette commande mémorise l'espacement dans la variable w, de telle sorte qu'il puisse être réutilisé, ce qui est le cas en particulier aux positions 151 et 163, où le même espace inter-mot intervient. La commande *w0* a pour effet un déplacement horizontal de w. Les commandes *x0*, *x1*, *x2* et *x3* sont analogues mais modifient x. Les commandes $w \ldots$ ou $x \ldots$ sont normalement utilisées soit pour mémoriser des espaces inter-mots, soit des crénages. Les commandes verticales équivalentes sont d'une part *y0*, *y1*, *y2* et *y3* (qui modifient y) et d'autre part *z0*, *z1*, *z2* et *z3* (qui modifient z).

À la position 164, la police est changée et `dvitype` récapitule le texte qui a été affiché.

Enfin, bien plus loin, la page s'achève avec *eop* (*end of page*), juste après que la pile a été dépilée avec des commandes *pop*. Ces commandes ont comme on peut le voir restitué le contexte, c'est-à-dire les valeurs des variables du sextuplet avant la commande *push* correspondante.

3.3. Commandes `special`

Si des commandes `\special` sont exécutées par TEX, elles se traduisent par l'insertion de commandes *xxx1*, *xxx2*, *xxx3* ou *xxx4* dans le fichier `DVI`. Ces commandes diffèrent par la longueur du texte qui les suit. L'exemple de la figure 4 montre ce qu'insère `\includegraphics{org.eps}`. Les quatre paramètres `llx`, `lly`, `urx` et `ury` sont les dimensions (en points PostScript) de la *bounding box* du fichier PostScript inclus et le dernier paramètre, `rwi`, est la largeur du fichier PostScript en dixièmes de points PostScript.

```
42: beginning of page 1
87: down4 42908243 v:=0+42908243=42908243, vv:=2718
92: push
level 0:(h=0,v=42908243,w=0,x=0,y=0,z=0,hh=0,vv=2718)
93: down4 -36008935 v:=42908243-36008935=6899308, vv:=437
98: push
level 1:(h=0,v=6899308,w=0,x=0,y=0,z=0,hh=0,vv=437)
99: down3 -783519 v:=6899308-783519=6115789, vv:=387
103: push
level 2:(h=0,v=6115789,w=0,x=0,y=0,z=0,hh=0,vv=387)
104: right3 -372935 h:=0-372935=-372935, hh:=-24
[ ]
108: fntdef1 22: cmr8
128: fntnum22 current font is cmr8
129: setchar66 h:=-372935+394314=21379, hh:=1
130: setchar114 h:=21379+217091=238470, hh:=15
131: setchar111 h:=238470+278532=517002, hh:=33
132: setchar117 h:=517002+309480=826482, hh:=53
133: setchar105 h:=826482+154740=981222, hh:=63
134: setchar108 h:=981222+154740=1135962, hh:=73
135: setchar108 h:=1135962+154740=1290702, hh:=83
136: setchar111 h:=1290702+278532=1569234, hh:=101
137: setchar110 h:=1569234+309480=1878714, hh:=121
138: w3 185688 h:=1878714+185688=2064402, hh:=131
142: setchar100 h:=2064402+309480=2373882, hh:=151
...
150: setchar101 h:=3797945+247584=4045529, hh:=258
151: w0 185688 h:=4045529+185688=4231217, hh:=268
152: setchar112 h:=4231217+309480=4540697, hh:=288
153: right2 15474 h:=4540697+15474=4556171, hh:=289
156: setchar111 h:=4556171+278532=4834703, hh:=307
...
163: w0 185688 h:=6169017+185688=6354705, hh:=403
[Brouillon d'article pour les ]
164: fntdef1 32: cmti8
185: fntnum32 current font is cmti8
186: setchar67 h:=6354705+402683=6757388, hh:=429
...
3413: pop
level 2:(h=-372935,v=41043563,w=0,x=0,y=622592,z=737281,hh=-24,vv=2600)
3414: pop
level 1:(h=0,v=41043564,w=0,x=0,y=0,z=0,hh=0,vv=2600)
3415: pop
level 0:(h=0,v=42908243,w=0,x=0,y=0,z=0,hh=0,vv=2718)
3416: eop
```

FIGURE 3 – *Exemple de description de page de fichier* `DVI` *(sortie* `dvitype`*)*

```
3534: xxx 'PSfile="org.eps" llx=130 lly=568 urx=482 ury=710 rwi=3520 '
```

FIGURE 4 – *Commande «special» dans un fichier* `DVI` *(sortie* `dvitype`*)*

Une commande «special» n'est pas interprétée par tous les pilotes, mais uniquement par ceux qui savent les traiter. Ainsi, le programme `dvips` va interpréter cette instruction comme l'inclusion du fichier `org.eps` dans le fichier PostScript final.

3.4. Postambule

Après la dernière page d'un fichier `DVI` se trouve le postambule. Celui-ci comporte un pointeur vers le début de la dernière page et répète la fraction et le facteur d'agrandissement que l'on trouve dans le préambule. Le postambule comporte la hauteur plus la profondeur de la plus haute page et la largeur de la plus large page. En général, ces informations ne sont pas utilisées. On trouve ensuite la profondeur maximale d'empilement (avec *push*), le nombre de pages dans le fichier, la définition des polices (déjà présente dans le préambule). Le fichier s'achève normalement avec quatre à sept octets dont la valeur est 223, de sorte que la longueur totale d'un fichier `DVI` soit un multiple de quatre.

Le postambule est tel qu'il est possible d'afficher un fichier `DVI` en commençant à le lire à partir de la fin.

4. Format GF

La principale sortie d'une exécution de METAFONT est un fichier `GF`. Ce fichier indique les emplacements des pixels d'un ensemble de caractères. Le format `GF` comporte beaucoup de points communs avec le format `DVI`, et cela se retrouve sur la figure 1. Un fichier `GF` est une suite d'octets pouvant être interprétés comme une suite de commandes.

Nous allons décrire le format de fichier `GF` en nous servant du programme `gftype` [5] qui permet de représenter de manière lisible un tel fichier. Une description complète du format se trouve dans [10]. Les figures 5, 6, 7 et 8 représentent des extraits de l'application de `gftype` au fichier `logo10.300gf`.

Un fichier `GF` consiste en un préambule, une suite de caractères et un postambule. Le préambule se limite à deux octets dont les valeurs sont 247 (commande *pre*, comme pour le format `DVI`) et 131, puis un commentaire d'au plus

255 octets, comportant par exemple la date de génération du fichier `GF` par METAFONT.

Ensuite (cf. figure 5) viennent les commandes des caractères. Chaque caractère commence par une commande *boc* (*beginning of character*) et s'achève par une commande *eoc* (*end of character*). Les caractères apparaissent dans l'ordre dans lequel METAFONT les a générés.

Tout comme pour le format `DVI`, le format `GF` laisse un certain nombre d'informations implicites, ce qui lui permet d'être relativement compact. Par exemple, les numéros de rangée et de colonnes où interviennent les pixels ne figurent pas explicitement dans le fichier `GF`. De plus, les valeurs autorisées dans le format `GF` sont bien plus générales que celles que produit un programme comme METAFONT, ce qui doit permettre au format `GF` d'être utilisé par de futures extensions de METAFONT.

4.1. Codage des caractères

Le point de référence d'un caractère (au sens de TEX) est le point situé en bas à gauche du pixel de la rangée 0 et de la colonne 0. Par conséquent, le caractère M représenté dans la figure 6 a la position inférieure gauche de son pixel inférieur gauche décalé de trois pixels vers la droite, par rapport au point de référence. Ceci correspond à l'*approche* du «M».

Dans une commande *boc*, le numéro de caractère est d'abord identifié. Dans la figure 5, il s'agit de 77 qui est le code ASCII de M. La commande *boc* comporte un pointeur vers le caractère précédent ainsi que des valeurs qui encadrent (mais pas nécessairement de manière exacte) les valeurs des numéros de colonnes (m) et de lignes (n). Le tracé commence à la plus petite valeur de m et à la plus grande valeur de n, c'est-à-dire en haut à gauche d'un caractère. `gftype` indique bien `initially n=24`. Suivent alors des commandes *paint*. Chaque commande *paint* noircit ou blanchit un nombre de pixels donnés en argument, et uniquement vers la droite. Bien que la première ligne nécessite quatre commandes *paint*, `gftype` les résume en `paint (0)3(21)3`, ce qui signifie qu'aucun octet ne doit être blanchi (ce qui a pour effet de changer la couleur et de passer au noir), que 3 pixels doivent être noircis, 21 pixels blanchis et à nouveau 3 pixels noircis.

Suit une commande *newrow 0* qui passe au début de la ligne suivante (c'est-à-dire décrémente n), en prenant pour couleur par défaut le *noir*. Le processus continue jusqu'à la fin du caractère (commande *eoc*).

```
This is GFtype, Version 3.1 (C version 6.1)
Options selected: Mnemonic output = true; pixel output = true.
’ METAFONT output 1997.04.03:0038’

35: beginning of char 77: 3<=m<=30 0<=n<=24
(initially n=24) paint (0)3(21)3
45: newrow 0 (n=23) paint 3(21)3
49: newrow 0 (n=22) paint 4(19)4
53: newrow 0 (n=21) paint 4(19)4
57: newrow 0 (n=20) paint 5(17)5
61: newrow 0 (n=19) paint 6(15)6
65: newrow 0 (n=18) paint 6(15)6
69: newrow 0 (n=17) paint 3(1)3(13)3(1)3
77: newrow 0 (n=16) paint 3(1)3(13)3(1)3
85: newrow 0 (n=15) paint 3(2)3(11)3(2)3
93: newrow 0 (n=14) paint 3(2)3(11)3(2)3
101: newrow 0 (n=13) paint 3(3)3(9)3(3)3
109: newrow 0 (n=12) paint 3(4)3(7)3(4)3
117: newrow 0 (n=11) paint 3(4)3(7)3(4)3
125: newrow 0 (n=10) paint 3(5)3(5)3(5)3
133: newrow 0 (n=9) paint 3(5)3(5)3(5)3
141: newrow 0 (n=8) paint 3(6)3(3)3(6)3
149: newrow 0 (n=7) paint 3(6)3(3)3(6)3
157: newrow 0 (n=6) paint 3(7)3(1)3(7)3
165: newrow 0 (n=5) paint 3(8)5(8)3
171: newrow 0 (n=4) paint 3(8)5(8)3
177: newrow 0 (n=3) paint 3(9)3(9)3
183: newrow 0 (n=2) paint 3(9)3(9)3
189: newrow 0 (n=1) paint 3(21)3
193: newrow 0 (n=0) paint 3(21)3
197: eoc
```

FIGURE 5 – *Commandes d’un caractère dans un fichier* `GF` *(sortie* `gftype`*)*

```
.<-This pixel's lower left corner is at (3,25) in METAFONT coordinates
***                       ***
***                       ***
****                     ****
****                     ****
*****                   *****
******                 ******
******                 ******
*** ***               *** ***
*** ***               *** ***
***  ***             ***  ***
***  ***             ***  ***
***   ***           ***   ***
***    ***         ***    ***
***    ***         ***    ***
***     ***       ***     ***
***     ***       ***     ***
***      ***     ***      ***
***      ***     ***      ***
***       *** *** ***       ***
***         *****         ***
***         *****         ***
***          ***          ***
***          ***          ***
***                       ***
***                       ***
.<-This pixel's upper left corner is at (3,0) in METAFONT coordinates
```

FIGURE 6 – *Bitmap d'un caractère dans un fichier* `GF` *(sortie* `gftype`*)*

```
836: xxx 'fontid=MFLOGO'
851: xxx 'codingscheme=AEFMNOPST only'
880: xxx 'fontfacebyte'
894: yyy 15335424 (234)
899: xxx 'jobname=logo10'
915: xxx 'mag=1'
922: xxx 'mode=cx'
931: xxx 'pixels_per_inch=300'
952: xxx 'blacker=0'
963: xxx 'fillin=0.2'
975: xxx 'o_correction=0.6'
```

FIGURE 7 – *Commandes* ***special*** *d'un fichier* `GF` *(sortie* `gftype`*)*

4.2. Commandes special

Des commandes « *special* » peuvent apparaître entre les caractères, après le préambule ou avant le postambule. Dans le fichier `logo10.300gf`, les commandes «*special*» sont regroupées avant le postambule (cf. figure 7).

Les commandes **special** de METAFONT sont traduites en commandes *xxx* dans le fichier `GF`. Il s'agit ici simplement de commentaires qui donnent les paramètres de la compilation METAFONT, en particulier le nom de la police et les paramètres du *mode*, comme la résolution.

La commande *yyy* est produite par une commande METAFONT **numspecial** et sert de paramètre à la commande *xxx* qui précède.

4.3. Postambule d'un fichier `GF`

Le dernier caractère d'un fichier `GF` est suivi d'un postambule (cf. figure 8). Celui-ci commence par indiquer le corps de référence (*design size*). Suit la somme de contrôle, identique à celle se trouvant dans le fichier `TFM` ayant été produit en même temps par METAFONT (voir plus loin). Viennent ensuite *hppp* et *vppp*, les nombres de pixels par point, horizontalement et verticalement. Ici $hppp = vppp = 300/72.27 = 4.1511$. Ces valeurs sont en fait indiquées en *scaled integers*, c'est-à-dire multipliées par 2^{16}.

Les valeurs minimales et maximales des numéros de lignes et de colonnes (pour l'ensemble des caractères) sont ensuite données.

Enfin, le postambule comporte la localisation de tous les caractères. Chaque caractère est indiqué par son numéro. Puis vient le déplacement horizontal *dx*. Il est indiqué par `gftype` à la fois en *scaled integers* et en pixels. Dans des polices plus sophistiquées, il pourrait aussi y avoir un déplacement vertical

```
Postamble starts at byte 993, after special info at byte 836.
design size = 10485760 (10pt)
check sum = -124489922
hppp = 272046 (4.1511)
vppp = 272046 (4.1511)
min m = 0, max m = 30
min n = 0, max n = 24
Character 65: dx 1835008 (28), width 699048 (27.67384), loc 312
Character 69: dx 1703936 (26), width 652445 (25.82893), loc 198
Character 70: dx 1703936 (26), width 652445 (25.82893), loc 411
Character 77: dx 2162688 (33), width 838858 (33.20863), loc 35
Character 78: dx 1835008 (28), width 699048 (27.67384), loc 703
Character 79: dx 1835008 (28), width 699048 (27.67384), loc 604
Character 80: dx 1703936 (26), width 652445 (25.82893), loc 468
Character 83: dx 1703936 (26), width 652445 (25.82893), loc 547
Character 84: dx 1507328 (23), width 605842 (23.98401), loc 255
The file had 9 characters altogether.
```

FIGURE 8 – *Fin de fichier* `GF` *(sortie* `gftype`*)*

dy, mais il est ici nul. Suit la largeur réelle du caractère telle qu'elle est donnée dans le fichier `TFM`. *dx* au contraire n'est qu'une approximation tenant compte de la résolution, c'est pourquoi *dx* se traduit par un nombre entier de pixels. Enfin, on trouve la localisation du début du caractère. Par exemple, le caractère 77 commence à la position 35.

Un fichier `GF` s'achève par un octet valant 131 suivi d'une série d'octets de valeur 223. METAFONT place de quatre à sept tels octets afin de rendre la taille du fichier `GF` un multiple de quatre.

Le postambule des fichiers `GF` rend possible la lecture d'un tel fichier à partir de la fin, tout comme pour le format `DVI`.

5. Format `TFM`

Un fichier `TFM` est utilisé par TEX pour obtenir la taille des caractères qui sont manipulés. Un fichier `TFM` peut être associé à un fichier `GF` qui aura été généré par une même exécution de METAFONT.

Un fichier `TFM` est une suite d'octets. Une première partie constitue l'en-tête du fichier.

5.1. Représentation d'un fichier TFM avec tftopl

Le programme tftopl [6] permet de produire une version intelligible d'un fichier TFM, appelée le format PL (*Property List*). Alors que la représentation d'un fichier DVI ou GF par dvitype ou gftype est très proche du fichier binaire, tftopl introduit au contraire de nombreux remaniements qui permettent de mieux comprendre le contenu d'un fichier TFM. De plus, le format PL diffère des formats de sortie de dvitype et de gftype par le fait que ce format peut être édité et un nouveau fichier binaire peut être produit par pltotf [7]. Nous nous limiterons ici à l'étude des sorties produites par tftopl.

5.2. En-tête d'un fichier TFM

L'un des fichiers TFM les plus simples est celui de la police logo, celle qui permet d'afficher « METAFONT » ou « METAPOST ». Le début de la sortie de tftopl est indiqué figure 9. Comme on le voit, le format PL consiste en une suite de commandes ou déclarations, chacunes entre parenthèses. Certaines déclarations sont simples, telles (FAMILY MFLOGO), d'autres comportent des composantes plus complexes, par exemple la déclaration LIGTABLE. Décrivons les différentes parties de la sortie de tftopl :

- FAMILY : ce mot-clé est suivi d'une chaîne d'au plus vingt caractères identifiant la police ; TEX ne tient pas compte de cette information ;
- FACE : ce mot-clé est suivi d'un octet précisant l'information donnée par FAMILY ; TEX ne tient pas non plus compte de cette information ;
- CODINGSCHEME : ce mot-clé indique le codage de la police et est ignoré par TEX ;
- DESIGNSIZE : ce mot-clé introduit le corps de référence de la police ; elle est indiquée en points ;
- COMMENT : ce mot-clé introduit un commentaire ajouté par tftopl ; au moment où un fichier PL est transformé en fichier TFM via pltotf, ces parties sont ignorées ;
- CHECKSUM : ceci introduit la somme de contrôle du fichier ; tftopl l'indique en octal (0) ;

5.3. Paramètres de dimension

L'en-tête (figure 9) se poursuit par l'indication des paramètres de la police dans la section `FONTDIMEN`. Chaque police comporte un certain nombre de paramètres qui figurent ici ; ces paramètres sont accessibles en TEX via la commande `\fontdimen` ; dans l'exemple de la figure 9, le premier paramètre est l'inclinaison (`SLANT`), utilisée pour positionner correctement les accents, le second l'espace normal (`SPACE`) obtenu dans TEX avec `\␣`, les troisième et quatrième sont l'extensibilité (`STRETCH`) et la contractabilité de l'espace (`SHRINK`), le cinquième paramètre est la hauteur d'un «x» minuscule (`XHEIGHT`) et le sixième est la largeur d'un cadratin (`QUAD`); toutes ces dimensions, à l'exception de `SLANT`, sont données en fractions du corps de référence `DESIGNSIZE`.

Normalement, toute police doit avoir au moins sept paramètres, à savoir les six que nous venons de voir et un septième représentant l'espace supplémentaire ajouté à la fin des phrases si l'option `\nonfrenchspacing` de TEX est activée.

Les polices EC [8] sont munies de neuf paramètres supplémentaires comme le montre l'extrait de la figure 10. Ces paramètres sont accessibles depuis TEX par `\fontdimen8` à `\fontdimen16`. Leur signification est la suivante :

fontdimen 8 est la hauteur des lettres capitales (non accentuées) ;

fontdimen 9 est la hauteur des lettres minuscules avec partie ascendante ;

fontdimen 10 est la hauteur des lettres capitales accentuées ;

fontdimen 11 est la profondeur des lettres minuscules avec partie descendante ;

fontdimen 12 est la hauteur maximale de tous les glyphes de la police ;

fontdimen 13 est la profondeur maximale de tous les glyphes de la police ;

fontdimen 14 est la largeur maximale des chiffres de la police ;

fontdimen 15 est la largeur de la barre verticale du «I» majuscule ;

fontdimen 16 est la valeur suggérée pour l'intervalle entre deux lignes de base (`\baselineskip`) avec cette police.

`tftopl` connaît les noms des sept premiers paramètres et utilise au-delà la notation conventionnelle `PARAMETER` suivi du numéro de paramètre en décimal (`D`).

```
(FAMILY MFLOGO)
(FACE O 352)
(CODINGSCHEME AEFMNOPST ONLY)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 37045067476)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.266665)
   (STRETCH R 0.133333)
   (SHRINK R 0.088888)
   (XHEIGHT R 0.0)
   (QUAD R 0.799997)
   )
```

FIGURE 9 – *Préambule du fichier* `PL` *de la police* `logo10` *(sortie* `tftopl`*)*

Les polices de symboles mathématiques doivent avoir au moins 22 paramètres (cf. figure 11) et les polices de symboles d'extension (comme les parenthèses) doivent en avoir au moins 13 (cf. figure 12). La signification de tous ces paramètres est donnée dans l'appendice G de [11] et nous n'en dirons pas plus ici.

5.4. Table de ligatures

Le mot-clé `LIGTABLE` est suivi d'une liste de commandes concernant les ligatures et crénages. La figure 13 donne un aperçu d'une telle liste. Celle-ci spécifie qu'un T suivi d'un A donnera lieu à un crénage (`KRN`) de -0.222225 points. De même, un F suivi d'un O donnera lieu à un crénage de -0.44444 points. Les commandes `LABEL` représentent des points d'entrée dans la liste des commandes, et les commandes `STOP` évitent que les ligatures ou crénages ne se fassent en cascade. Les ligatures et crénages sont repris en commentaires pour chaque caractère comme on peut le voir à la figure 14. Des explications plus détaillées sur le format de la table de ligatures se trouvent dans [7].

5.5. Dimensions des caractères

Après l'en-tête du fichier `TFM`, la liste des paramètres et la table des ligatures et crénages, on trouve le détail de chaque caractère (cf. figure 14). Pour chacun d'eux sont indiqués le cas échéant la largeur (`CHARWD`), la hauteur (`CHARHT`), la profondeur (`CHARDP`), la correction italique (`CHARIC`) et éventuellement des

```
(FAMILY ECRM)
(FACE O 352)
(CODINGSCHEME EXTENDED TEX FONT ENCODING - LATIN)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 1414365261)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.333252)
   (STRETCH R 0.166626)
   (SHRINK R 0.111084)
   (XHEIGHT R 0.43045)
   (QUAD R 0.999756)
   (EXTRASPACE R 0.111084)
   (PARAMETER D 8 R 0.6831665)
   (PARAMETER D 9 R 0.694275)
   (PARAMETER D 10 R 0.891449)
   (PARAMETER D 11 R 0.194397)
   (PARAMETER D 12 R 0.891449)
   (PARAMETER D 13 R 0.249939)
   (PARAMETER D 14 R 0.499878)
   (PARAMETER D 15 R 0.088867)
   (PARAMETER D 16 R 1.199997)
   )
```

FIGURE 10 – *Préambule du fichier* `PL` *de la police* `ecrm1000` *(sortie* `tftopl`*)*

```
(FAMILY CMSY)
(FACE O 352)
(CODINGSCHEME TEX MATH SYMBOLS)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 4110426232)
(FONTDIMEN
   (SLANT R 0.25)
   (SPACE R 0.0)
   (STRETCH R 0.0)
   (SHRINK R 0.0)
   (XHEIGHT R 0.430555)
   (QUAD R 1.000003)
   (EXTRASPACE R 0.0)
   (NUM1 R 0.676508)
   (NUM2 R 0.393732)
   (NUM3 R 0.443731)
   (DENOM1 R 0.685951)
   (DENOM2 R 0.344841)
   (SUP1 R 0.412892)
   (SUP2 R 0.362892)
   (SUP3 R 0.288889)
   (SUB1 R 0.15)
   (SUB2 R 0.247217)
   (SUPDROP R 0.386108)
   (SUBDROP R 0.05)
   (DELIM1 R 2.389999)
   (DELIM2 R 1.01)
   (AXISHEIGHT R 0.25)
   )
```

FIGURE 11 – *Préambule du fichier* `PL` *de la police* `cmsy10` *(sortie* `tftopl`*)*

```
(FAMILY CMEX)
(FACE O 352)
(CODINGSCHEME TEX MATH EXTENSION)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 37254272422)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.0)
   (STRETCH R 0.0)
   (SHRINK R 0.0)
   (XHEIGHT R 0.430555)
   (QUAD R 1.000003)
   (EXTRASPACE R 0.0)
   (DEFAULTRULETHICKNESS R 0.039999)
   (BIGOPSPACING1 R 0.111112)
   (BIGOPSPACING2 R 0.166667)
   (BIGOPSPACING3 R 0.2)
   (BIGOPSPACING4 R 0.6)
   (BIGOPSPACING5 R 0.1)
   )
```

FIGURE 12 – *Préambule du fichier* `PL` *de la police* `cmex10` *(sortie* `tftopl`*)*

```
(LIGTABLE
   (LABEL C T)
   (KRN C A R -0.0222225)
   (STOP)
   (LABEL C F)
   (KRN C O R -0.044444)
   (STOP)
   (LABEL C P)
   (KRN C O R 0.044444)
   (STOP)
   )
```

FIGURE 13 – *Table de ligatures du fichier* `PL` *de la police* `logo10` *(sortie* `tftopl`*)*

```
(CHARACTER C A
   (CHARWD R 0.666664)
   (CHARHT R 0.6)
   )
(CHARACTER C E
   (CHARWD R 0.62222)
   (CHARHT R 0.6)
   )
(CHARACTER C F
   (CHARWD R 0.62222)
   (CHARHT R 0.6)
   (COMMENT
      (KRN C O R -0.044444)
      )
   )
...
```

FIGURE 14 – *Caractères d'un fichier* `PL` *(sortie* `tftopl`*)*

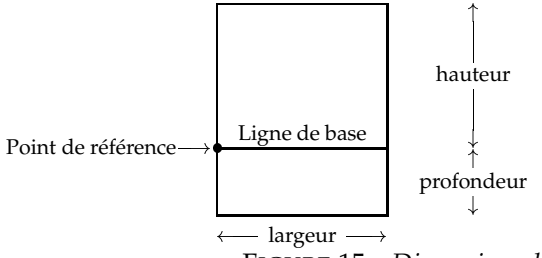

FIGURE 15 – *Dimensions d'un caractère*

ligatures ou crénages en commentaires. La figure 15 indique la signification de ces paramètres.

Dans les polices de symboles extensibles, telle `cmex10`, un caractère peut être lié à un caractère qui le suit en taille. Ceci est indiqué via le mot-clé `NEXTLARGER`. Ainsi, l'exemple de la figure 16 signifie que le caractère de code octal 0 ( ) [8] est suivi du caractère de code octal 20 ( ). De telles listes permettent à TEX de choisir un symbole de taille convenable en parcourant simplement une liste. La liste complète de la série de parenthèses est : (, (, (, (, (.

8. La hauteur de ce symbole et des suivants est pratiquement nulle ce qui explique les effets décentrés.

```
(CHARACTER O 0
   (CHARWD R 0.458336)
   (CHARHT R 0.039999)
   (CHARDP R 1.160013)
   (NEXTLARGER O 20)
   )
```

FIGURE 16 – *Liens entre caractères dans* `cmex10` *(sortie* `tftopl`*)*

```
(CHARACTER O 62
   (CHARWD R 0.666669)
   (CHARHT R 0.039999)
   (CHARDP R 1.760019)
   (VARCHAR
      (TOP O 62)
      (BOT O 64)
      (REP O 66)
      )
   )
```

FIGURE 17 – *Caractères décomposés dans* `cmex10` *(sortie* `tftopl`*)*

Des caractères extensibles composés d'éléments simples peuvent aussi être spécifiés. Ainsi, l'exemple de la figure 17, toujours extrait du fichier `cmex10`, indique que le caractère octal 62 est un élément d'un symbole extensible dont le *haut* (`TOP`) est le symbole octal 62 ( ), le *bas* (`BOT`) est le symbole octal 64 ( ) et la partie répétée (`REP`) est le symbole octal 66 ( ).

6. Format VF

Les polices virtuelles ont été introduites par Knuth en 1990 et sont inspirées d'un programme similaire réalisé par David R. Fuchs en 1984. Une police virtuelle représente une *abstraction* d'un fichier `TFM`, en ce sens qu'un fichier `VF` va faire le lien entre une police vue par TEX (au travers d'un fichier `TFM`) et une ou plusieurs polices réelles. On peut voir le format `VF` comme une extension du format `TFM`. Très souvent, une police virtuelle permettra simplement de réencoder une police déjà existante, et c'est ce qui est fait par exemple pour `ptmr7t` (*Times Roman* au codage OT1) ou `ptmr8t` (*Times Roman* au codage T1) qui sont des versions recodées de `ptmr8r`. Nous allons sommairement illustrer le contenu d'un fichier `VF` au travers du programme `vftovp` [13] qui permet de passer d'un fichier `.vf` et de son fichier `.tfm` associé à un fichier `.vpl`, c'est-à-dire une version lisible de son contenu.

```
(VTITLE virtual font ptmr7t created by fontinst v1.335)
(FAMILY UNSPECIFIED)
(FACE F MRR)
(CODINGSCHEME TEX TEXT)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 614675731)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.25)
   (STRETCH R 0.15)
   (SHRINK R 0.0599985)
   (XHEIGHT R 0.45)
   (QUAD R 1.0)
   (EXTRASPACE R 0.0599985)
   )
(MAPFONT D 0
   (FONTNAME ptmr8r)
   (FONTCHECKSUM O 4767720433)
   (FONTAT R 1.0)
   (FONTDSIZE R 10.0)
   )
```

FIGURE 18 – *Début de fichier* `VF` *(sortie* `vftovp`*)*

La figure 18 montre le préambule du fichier `VPL` (*Virtual Property List*) correspondant à la police `ptmr7t`. Par rapport au format `PL`, outre l'identification par `VTITLE` (qui indique que le fichier `VPL` a été créé avec le programme `fontinst` d'Alan Jeffrey), la seule nouveauté est la partie `MAPFONT`. Celle-ci indique les polices de base. Ici, il n'y en a qu'une, à savoir `ptmr8r`. Le numéro interne attribué à cette police est le paramètre de `MAPFONT`, soit 0. `FONTCHECKSUM` est la somme de contrôle de la police à laquelle il est fait référence. Le paramètre `FONTAT` indique le facteur permettant de passer de la taille de la police à laquelle il est fait référence (`ptmr8r`) à la taille de la police virtuelle courante. Enfin, `FONTDSIZE` est le corps de référence absolu de la police `ptmr8r`.

Une fois la ou les polices de base indiquées, chaque définition de caractère peut faire référence à une police de base ou éventuellement à d'autres constructions. Dans la figure 19, on voit la définition de deux caractères. Le premier correspond à l'affichage d'un petit rectangle noir au moyen de la commande `SETRULE`. Cette commande est suivie des dimensions du rectangle. L'affichage de ce rectangle donne '■' et donne lieu au message `Warning: missing glyph 'Gamma'` figurant dans la commande `SPECIAL`.

```
(CHARACTER O 0
   (CHARWD R 0.5)
   (CHARHT R 0.502997)
   (MAP
      (SETRULE R 0.5 R 0.5)
      (SPECIAL Warning: missing glyph ‘Gamma’)
      )
   )
...
(CHARACTER C A
   (CHARWD R 0.721997)
   (CHARHT R 0.667999)
   (COMMENT
      (KRN C y R -0.091992)
      (KRN C w R -0.091992)
      (KRN C v R -0.073999)
      (KRN O 47 R -0.110999)
      (KRN C Y R -0.104993)
      (KRN C W R -0.08999)
      (KRN C V R -0.134998)
      (KRN C U R -0.054993)
      (KRN C T R -0.110999)
      (KRN C Q R -0.054993)
      (KRN C O R -0.054993)
      (KRN C G R -0.03999)
      (KRN C C R -0.03999)
      (KRN O 37 R -0.054993)
      )
   (MAP
      (SETCHAR C A)
      )
   )
...
```

FIGURE 19 – *Caractères dans un fichier* `VF` *(sortie* `vftovp`*)*

Le second caractère correspond simplement au «A» de la police de base. Ceci signifie donc que lorsque le caractère «A» de la police `ptmr7t` sera demandé, il s'agira en fait du caractère «A» de la police `ptmr8r`. TEX lui-même ne fera pas la différence, mais les pilotes feront le lien entre ces deux représentations.

L'extrait de la figure 20 montre le caractère octal 202 de `ptmr8t`. Celui-ci est composé de deux caractères de base de la police `ptmr8r` : l'accent aigu (caractère octal 264) et le « C ». L'accent est d'abord placé légèrement à droite (`MOVERIGHT`) et vers le haut (valeur négative pour `MOVEDOWN`) par rapport à sa position normale. Ces déplacements sont effectués entre `PUSH` et `POP` de sorte que lorsque le «C» est placé, on se trouve là où l'on se trouvait avant de placer l'accent.

```
(CHARACTER O 202
   (CHARWD R 0.666992)
   (CHARHT R 0.8984995)
   (CHARDP R 0.013995)
   (MAP
      (PUSH)
      (MOVEDOWN R -0.225989)
      (MOVERIGHT R 0.166992)
      (SETCHAR O 264)
      (POP)
      (SETCHAR C C)
      )
   )
...
```

FIGURE 20 – *Déplacements et empilements dans un fichier* `VF` *(sortie* `vftovp`*)*

Il est possible de spécifier des opérations bien plus complexes dans une police virtuelle ; en particulier, il est possible d'insérer des fragments de PostScript dans les commandes `SPECIAL`, pour modifier certains caractères. Il est même possible de mettre des pages `DVI` entières dans des fichiers `VF` et placer des caractères revient alors à placer des images `DVI` au sein d'un document ! Une description détaillée du format des polices virtuelles est donnée dans [14] ou dans [15].

7. Conclusion

Nous avons voulu dans cet article donner un panorama de quelques formats de fichiers fréquemment rencontrés dans le monde TEX. Grâce à l'emploi de programmes tels que `dvitype`, `gftype`, `tftopl`, etc., il est possible d'analyser et de comprendre le contenu de ces fichiers et éventuellement de détecter des anomalies. Nous espérons avoir ainsi contribué à lever un voile sur des aspects quelquefois mystérieux de TEX.

Bibliographie

[1] André (Jacques). – Caractères numériques : introduction. *Cahiers GUTenberg*, 26, 1997, pp. 5–44.

[2] Bouche (Thierry). – Minion MM : installer une famille de fontes multi-master. *Cahiers GUTenberg*, vol. 26, 1997, pp. 45–70.

[3] Cousquer (Alain) et Picheral (Éric). – Polices, TEX & Cie. *Cahiers GUTenberg*, vol. 9, 1991, pp. 3–31.

[4] Fuchs (David R.) et Knuth (Donald E.). – `dvitype`, version 3.4, 1990. Distribution TEX standard.

[5] Fuchs (David R.) et Knuth (Donald E.). – `gftype`, version 3.1, 1991. Distribution TEX standard.

[6] Guibas (Leo), Knuth (Donald E.), Liang (Frank), Ramshaw (Lyle) et Wyatt (Doug). – `tftopl`, version 3.1, 1989. Distribution TEX standard.

[7] Guibas (Leo), Knuth (Donald E.), Liang (Frank), Ramshaw (Lyle) et Wyatt (Doug). – `pltotf`, version 3.4, 1991. Distribution TEX standard.

[8] Knappen (Jörg). – The dc fonts 1.3: Move towards stability and completeness. *TUGboat*, vol. 17, n2, 1996, pp. 99–101.

[9] Knuth (Donald E.). – *TEX: The Program*. – Reading, MA, USA, Addison-Wesley, 1986, *Computers and Typesetting*, volume B, xv + 594p.

[10] Knuth (Donald E.). – METAFONT*: The Program*. – Reading, MA, USA, Addison-Wesley, 1986, *Computers and Typesetting*, volume D, xv + 560p.

[11] Knuth (Donald E.). – *The TEXbook*. – Reading, MA, USA, Addison-Wesley, 1986, *Computers and Typesetting*, volume A, ix + 483p.

[12] Knuth (Donald E.). – *The* METAFONT*book*. – Reading, MA, USA, Addison-Wesley, 1986, *Computers and Typesetting*, volume C, xi + 361p.

[13] Knuth (Donald E.). – `vftovp`, version 1.2, 1990. Distribution TEX standard.

[14] Knuth (Donald E.). – Virtual Fonts: More Fun for Grand Wizards. *TUGboat*, vol. 11, n1, avril 1990, pp. 13–23.

[15] Knuth (Donald E.). – `vptovf`, version 1.3, 1991. Distribution TEX standard.

[16] The TUG `DVI` Driver Standards Committee. – The `DVI` Driver Standard, Level 0, 1991. Fichier `CTAN: dviware/driv-standard/level-0/dvistd0.tex`.