Cahiers GUTenberg

MINION MM : INSTALLER UNE FAMILLE DE FONTES

Thierry Bouche

Cahiers GUTenberg, nº 26 (1997), p. 45-70.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1997___26_45_0>

Minion MM : installer une famille de fontes *multi-master* *

Thierry BOUCHE

Institut Fourier
Laboratoire de mathématiques pures
Université Joseph Fourier – Grenoble

Résumé. On décrit pas à pas les étapes nécessaires pour installer et utiliser pleinement une famille de fontes comme *Minion multi-master* avec LaTeX.

Abstract. *A rather precise description of the installation procedure of a multiple master text font family like* Minion MM *is given.*

1. Introduction

Au vu de publications récentes [4, 6, 7, 10], il est indéniable que la famille de caractères *Minion* dessinée par Robert SLIMBACH jouit d'une grande popularité dans le Landernau des LaTeXniciens. Il y a à cela deux raisons évidentes (comme toujours en typographie numérique, l'une est esthétique, l'autre technique) : il s'agit de la première fonte textuelle *multi-master* dotée d'un axe « corps de référence » ; de plus elle a de nombreuses caractéristiques qui en font un excellent choix pour les publications scientifiques désirant se distinguer de l'omniprésent *Times*. L'apparition de cette notion de *corps de référence*[1] a représenté une évolution importante dans le monde de la PAO bien qu'elle ait existé depuis de nombreuses années dans le monde TeX et ses méta-fontes. Je donne ici mes notes d'installation décrivant la totalité du travail à accomplir pour disposer d'un système (et d'un support) complet pour toutes les fontes de cette famille.

2. Description du produit

La technologie *multi-master* est une extension du format des fontes PostScript de « type 1 », qui permet en fait d'interpoler linéairement entre des dessins « extrê-

* Note présentée dans le cadre d'un tutoriel sur les fontes lors des Journées GUTenberg de Strasbourg, le 26 mai 1997. Nous avons choisi de composer cet article en *Minion*, de façon à illustrer son sujet, et à simplifier le recours aux illustrations (qui sont toutes codées en LaTeX dans le source).

1. Le corps pour lequel le dessin a été optimisé visuellement : les caractères utilisés en 5, 8 ou 10 points devraient avoir un dessin leur permettant de cohabiter en terme de « gris typographique » et de lisibilité.

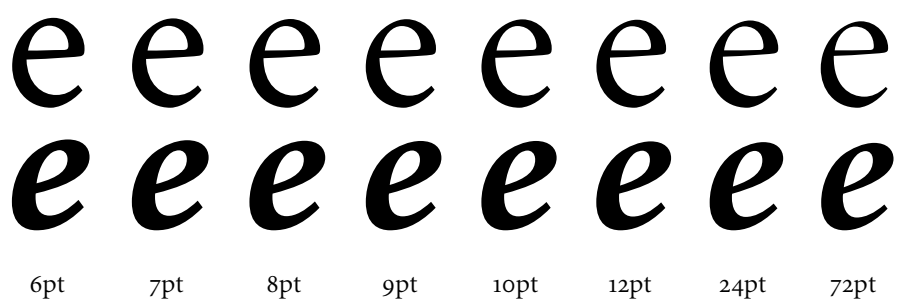

Fig. 1 – *Variations du corps de référence (maigre et demi-gras italique).*

maux » (les *masters*) : il suffit au créateur de caractères de dessiner (au plus) seize fontes PostScript aux caractéristiques semblables ; un opérateur makeblendedfont permettra de fabriquer une infinité de contours intermédiaires. *Minion multi-master* (mm) possède ainsi trois axes : graisse (de maigre à ultra-gras) ; chasse (d'étroit à normal) et corps de référence (de 6[2] à 72 points, cf. Fig. 1).

Minion multi-master (mm) est vendue par la firme Adobe sous la forme de huit fontes (toutes sont modulables selon les trois axes décrits plus haut) :

MinionMM La fonte de base : un romain proposant les 228 glyphes standard des fontes PostScript (Fig. 2) ;

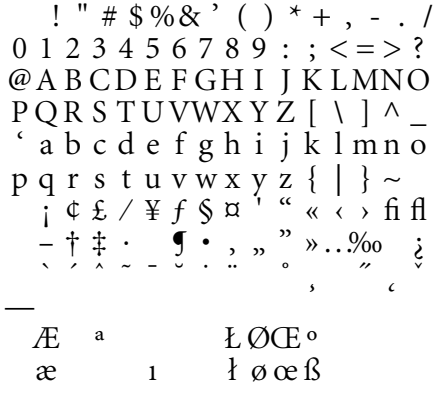

Fig. 2 – *Casse de* Minion *(codage d'origine).*

2. Notons tout de même que le format mm nécessitant des *masters* assez semblables, le dessin ne peut subir les distorsions importantes que nécessite la lisibilité des très petits corps comme 5 ou 6 points. De ce point de vue, *Minion* n'est toujours pas idéale pour la publication de mathématiques — il est à noter qu'Adobe est conscient de ce problème puisque de nouvelles fontes font appel à des *masters* intermédiaires, ce qui réduit le nombre d'axes possibles...

MinionMM-Ep Le complément expert, soit 165 caractères supplémentaires dont les petites capitales, les chiffres elzéviriens et les ligatures en ff, inexplicablement absents du « standard » ci-dessus (FIG. 3) ;

FIG. 3 – *Casse de* Minion *Expert.*

MinionMM-It, MinionMM-ItEp La même chose pour la fonte d'italiques associée (FIG. 4) ;

FIG. 4 – *Casse de* Minion *italique (codage 8r).*

MinionMM-Sw Une fonte d'usage strictement calligraphique d'un type assez inusité en TEX : les capitales ornées qui représentent en fait le dessin cursif cohérent avec les italiques, mais dont l'usage est limité aujourd'hui à des fonctions décoratives exceptionnelles — cette fonte est enrichie de deux ligatures courantes au temps du plomb : *ct, st* (Fig. 5) ;

A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
ct
st

Fig. 5 – *Casse de* Minion *Swash.*

MinionMM-Or Une fonte de fleurons, culs-de-lampes (Fig. 6) ;

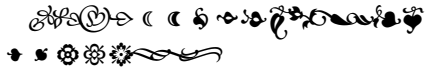

Fig. 6 – *Casse de* Minion *Ornaments.*

MinionMM-SC Une fonte n'apportant pas de glyphe nouveau, dont l'utilité vient peut-être du fait qu'aucun codage de caractères ne supporte les petites capitales indépendamment des bas de casse (Fig. 7) ;

! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [\ ] ^ _
‘ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z { | } ~
¡ ¢ £ / ¥ ƒ § ¤ ' “ « ‹ › FIFL
– † ‡ · ¶ • ‚ „ ” » … ‰ ¿
` ´ ˆ ˜ ¯ ˘ ˙ ¨ ˚ ˝ ˇ
’ ‘
—
Æ ª Ł Ø Œ º
Æ I Ł Ø ŒSS

Fig. 7 – *Casse de* Minion *petites capitales (codage par défaut).*

MinionMM-ItSC Idem pour la version italiques (*Minion* est l'une des rares fontes proposant les petites capitales italiques, cf. Fig. 8).

FIG. 8 – *Casse de* Minion *petites capitales italiques (codage T1).*

Dans la pratique, on utilise des « avatars » (*instances*) qui sont définis par un choix de paramètres (coordonnées relatives aux axes de modulation). Un avatar est matérialisé par un petit fichier « d'amorce » (*PostScript stub*, d'extension PSS). Il faut charger la fonte MM, puis l'amorce pour définir un dessin vectoriel équivalent d'une fonte de type 1 *mono-master* (SM). Dans un fichier PostScript, on ne chargera qu'une fois la fonte MM, quelle que soit la quantité d'avatars requise : de ce point de vue, la technologie MM est économique.

Comme j'ai décidé dès le départ de compléter *Minion* par les glyphes mathématiques de la famille *Computer Modern*, j'ai utilisé la flexibilité de la technologie MM pour obtenir la variante de *Minion* qui se rapproche le plus (autrement dit : jure le moins) avec *CM*. J'ai donc sélectionné le dessin le moins étroit et le plus maigre possible, en conservant un certain nombre de variations du corps de référence (6, 7, 8, 9, 10, 11, 12, 24 et 72 points). J'ai également sélectionné une variante grasse (qui en fait correspondrait plutôt à un demi-gras dans l'échelle de *Minion*, tant *CM* est peu grasse). La famille que je me propose d'installer est donc constituée de l'équivalent de 90 fontes PostScript *mono-master*, soit neuf corps et deux graisses pour chacune des cinq fontes textuelles ci-dessus : je n'ai installé que quatre variantes des fleurons.

3. Qu'est-ce qu'une fonte ?

Avant de nous lancer dans la description nécessairement laborieuse de l'installation proprement dite, essayons de cerner son utilité, et sa portée. Ce qui se ramène à cette bête question : au bout du compte, qu'est-ce qu'une fonte ?

Je ne vais pas essayer ici de répondre de façon catégorique à cette question plus vicieuse qu'il n'y paraît au premier abord ; mais je vais au moins décrire les réponses possibles d'un point de vue *fonctionnel*, le seul qui nous importe réellement ici.

Il faut bien voir que chaque programme de la « chaîne graphique » utilise de façon cruciale une notion de fonte, celle-ci n'étant jamais vraiment la même. Je viens d'égrener ce que l'on trouve dans les deux coffrets « *Minion* MM & *Minion* Expert MM » vendus par Adobe. Pour chaque « fonte » MM énumérée, j'ai de fait reçu deux fichiers : le premier décrit les contours de chaque caractère (c'est en termes récents une collection de *glyphes*) tandis que le second apporte une description uniquement métrique de la fonte et de ses caractères (table des chasses, mais aussi hauteur des capitales, corrections des approches de paires, etc.). Un programme comme *Page Maker* utiliserait directement ces deux fichiers (la description des contours pour afficher le texte saisi, la description des métriques pour composer la page en fonction des réglages de C&J (césures et justifications). L'installation se résumerait en général à copier les fontes dans un répertoire prévu à cet effet, et les déclarer auprès d'*ATM*. On y accéderait alors par un menu déroulant, le présent article paraît un peu long en comparaison...

Un système TEX repose sur une conception bien différente, dans laquelle chaque étape menant de la saisie à la production de l'imprimé final est prise en charge par un programme spécifique. Chacun de ces programmes gère donc une notion de *fonte* qui lui est propre, et dont il est utile de bien préciser la spécificité.

TEX Le seul travail du « moteur » TEX est de produire un fichier DVI [9] décrivant les page à imprimer de façon précise. Pour être en mesure d'effectuer les calculs nécessaires, il lui suffit de connaître les métriques attachées à chaque police utilisée, ce qui se résume en un fichier TFM. En faisant usage de la distinction prônée par [2] on peut dire que le moteur TEX ne se préoccupe pas de fonte, mais de police : ici un TFM à une certaine échelle. Les glyphes ne le concernent pas. Pour TEX, une fonte est donc exactement ce que l'on trouve dans un fichier TFM, c'est donc 256 caractères imprimables (aucune place perdue car n'étant pas Wysiwyg, TEX n'a pas besoin de caractères spéciaux « de contrôle » pour interagir avec le système d'exploitation). Les fonctions essentielles pour une typographie de qualité que sont les césures (coupures de mot en fin de ligne pour faciliter la justification), le crénage (corrections d'approches de paires, pour maintenir un espacement consistant à l'intérieur des mots) et les ligatures auto-

matiques s'effectuent à l'intérieur d'une même fonte[3]. Ceci signifie que TEX coupe un mot seulement s'il est composé d'une suite de caractères provenant de la même police (on pourrait dire que ceci est la définition du terme *mot* pour un programme de ce type). Par exemple, lorsque l'on utilise le codage OT1 des fontes *Computer Modern* comme cmr10, la lettre é, n'est pas un caractère de cette police, mais une boite dans laquelle le glyphe accentué a été produit par TEX à l'aide de la primitive \accent qui place l'accent aigu puis imprime la lettre e dessous. Par conséquent, TEX ne peut couper un mot comportant de telles lettres[4]. Le codage OT1 est un codage sur 7 bits qui n'autorise pas plus de 128 caractères, le recours à un codage 8 bits permet d'inclure beaucoup plus de caractères diacrités, c'est la motivation du codage T1 utilisé implicitement dans la suite de cet article. D'un point de vue théorique, l'adoption d'Unicode, un codage sur 16 bits, devrait rejeter aux oubliettes de l'histoire les débats douloureux sur l'omission de tel ou tel caractère (donc l'exclusion du support pour les langues l'utilisant). Pour fixer les idées, on peut comparer les figures 8 et 9[5].

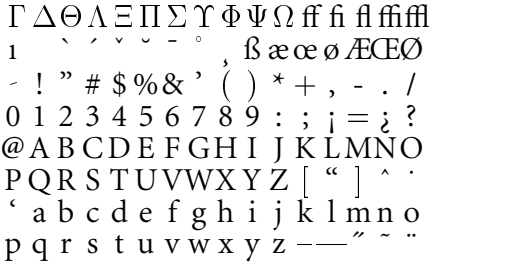

Fig. 9 – *Casse de* Minion *(codage OT1).*

De même, TEX est capable d'appliquer un crénage ou une ligature automatiquement entre deux caractères d'une même police dès que ces informations se trouvent dans le fichier TFM correspondant. Par exemple, au vu de la liste qui précède, pour composer le mot *Taſtes*, le programme devrait être capable de sélectionner le glyphe « Tswash » de la fonte MinionMM-Sw, puis de sélectionner la fonte MinionMM-It pour imprimer le *a*. Il est probable qu'il sera nécessaire de corriger l'approche de paire entre ces deux glyphes (i.e. rentrer le *a* sous la barre du *T* pour équilibrer le blanc laissé entre les deux lettres). Ensuite, il faudrait imprimer la ligature *ſt* en revenant à MinionMM-Sw pour finir avec MinionMM-It. En fin de ligne, il faudrait peut-être couper ce mot en *Tas-tes*. Toutes ces opérations doivent être faites à la main si elles font appel à deux

3. Ceci est une limitation partagée par tous les programmes de PAO dont j'ai entendu parler. Les fonctions Quick Draw GX du Macintosh, ou les fontes virtuelles de TEX sont à ce jour les seules technologies permettant de contourner ce problème.

4. Du moins après une telle lettre. Il existe une version modifiée de TEX : MlTEX qui permet tout de même de couper les mots accentués *via* la primitive \accent.

5. Certains glyphes (capitales grecques en particulier) proviennent de *CM*.

fontes distinctes, mais TEX permettant de créer des fontes virtuelles composées de plusieurs fontes, il suffit que les glyphes utilisés ci-dessus soient inclus dans une même fonte virtuelle, dans laquelle on aura spécifié les crénages et ligatures désirés pour que le mot *Taſtes* s'obtienne simplement par la saisie `Tastes`, la césure automatique restant alors possible.

LATEX Le (plus très) « nouveau schéma de sélection des polices » ou NFSS, est l'interface utilisée désormais par LATEX, laissant l'utilisateur sélectionner les fontes par attributs « orthogonaux » (la modification d'un attribut n'altérant pas la valeur des autres, tant que la fonte correspondante existe). Pour l'utilisateur, une fonte est donc la donnée d'un corps, d'un codage, d'une famille, d'une « série » (concept fourre-tout très mal défini) et d'une « forme ». À l'autre bout, LATEX doit convertir ces attributs de haut niveau en noms de fichiers TFM à passer au moteur TEX. C'est l'objet des fichiers FD (cf. § 4.2.4).

Pilotes Lorsque le pilote interprète le fichier DVI, son but est de donner forme aux instructions métriques qu'il contient sans trop les dénaturer. Il a toujours besoin du TFM pour connaître la chasse de chaque caractère, donc où placer le suivant, mais il est le premier à avoir pour finalité d'afficher graphiquement la page, il est donc le premier à s'intéresser au dessin des caractères. S'il s'agit d'un pilote PostScript (*dvips, dvipsone*, ...) il recopiera telle quelle la définition des contours dans le fichier PostScript généré. S'il s'agit d'un utilitaire de visualisation sur écran (*xdvi, dviwin, dviwindo*, ...) il aura besoin de ce dessin sous forme de *bitmap* de basse définition, qui peuvent être calculés à la volée par un programme résidant comme *ATM* pour les systèmes qui en disposent, sinon par des utilitaires séparés comme *gsftopk*.

4. L'installation

L'installation comporte cinq étapes :

- associer un nom unique et reconnaissable par des programmes à chaque fonte ;
- créer les métriques dans le format utilisé par TEX, en faisant appel ou non à l'artifice de *fontes virtuelles* ;
- créer l'interface permettant à LATEX d'y recourir pleinement ;
- instruire les pilotes utilisés de la façon de gérer ces nouvelles fontes ;
- enfin, ranger tous les fichiers de telle sorte qu'ils soient accessibles aux programmes qui les utilisent.

4.1. Quels noms donner aux fichiers ?

Nous allons avoir à manipuler environ 90 fontes. Pour chaque fonte, nous créerons au minimum un fichier définissant le contour des caractères PSS, un AFM, deux

TFM, un VF [9] : il est par conséquent essentiel d'automatiser au maximum ces tâches, et de pouvoir appliquer à chaque fichier le traitement convenable sans recours à une intervention manuelle. Dans cette optique, il est très utile de disposer d'une convention permettant à des programmes d'identifier sans ambigüité la fonte et le type de fichier à l'aide du nom de chaque fichier. Les fontes elle-mêmes sont en général vendues avec un nom « théorique » qui ne sert à rien (il est parfois affiché dans les menus déroulants « polices » de certaines applications peu recommandables) comme *Minion MM Italic*, d'un nom interne PostScript identifiant le « dictionnaire » qui contient cette fonte (MinionMM-It), et un nom de fichier qui varie beaucoup selon le système d'exploitation, le fondeur, et les aléas de la vie... Par exemple, si je crée les avatars que j'ai mentionnés avec *ATM*, les fichiers PSS auront un préfixe bizarre (zgx__) suivi du numéro d'ordre de création en hexadécimal, donc ce nom n'identifie en rien ce fichier en dehors de mon installation.

Pour les systèmes qui le permettent (noms de fichiers arbitrairement longs, et distinction capitales/bas de casse), il est possible de donner à ces fichiers le nom PostScript de la fonte. Malheureusement, de tels noms sont attribués au petit bonheur, les variations de dessin étant spécifiées de façon très variable, voire très étrange... Pour caractériser l'italique, on trouve *Italic, It, Ital*, mais aussi *Kursiv* voire *Italique* ! la version la plus grasse d'Antique Olive est *AntiqueOlive-Nord*...

Karl Berry [3] a introduit un schéma visant à nommer toute fonte de façon systématique. Ce schéma rend donc de grands services, il avait également pour but de fournir des noms courts compatibles avec le format 8+3 de MS-DOS ou des cédéroms. Étant donnée la diversité des fontes dans le monde, ceci est malheureusement impossible, mais ne doit pas être perdu de vue lorsque l'on prévoit de diffuser le type d'installation que je décris ici. Ce schéma se résume comme suit :

$$S \quad TT \quad W \quad [V_1 V_2 \ldots] \quad [N] \quad [E] \quad [DD]$$

ce qui signifie qu'on attribue à une fonte donnée un identifiant constitué des caractères (les crochets signalent comme d'habitude des identifiants optionnels)

- S vendeur (*supplier*) indiquant la provenance de la fonte (Adobe est identifié par la lettre *p*) ;
- TT fonte (*typeface*) : deux lettres pour identifier la fonte (ici, *Minion* équivaut à *mn*) ;
- W graisse (*weight*) à choisir parmi *l* (*light* : maigre), *r* (*regular* : normal), *d* (*demibold* : demi-gras), *b* (*bold* : gras), etc. Comme les fontes MM n'ont pas à proprement parler de graisse, j'ai inventé l'identifiant *MM* qui est légèrement en contradiction avec les usages, une autre solution est de considérer ces fontes comme bizarres, et de leur attribuer un nom quelconque débutant par la lettre *z* ;
- V_i variantes : ici on précise la ou les variante(s) concernée(s), concept un peu flou dans lequel entrent le romain (*r*, en général implicite donc omis), les italiques (*i*), les petites capitales (*c*), les chiffres elzéviriens (*j*) ;

N codage (*encoding*), ceux qui nous intéressent ici sont 8*a* Adobe Standard Encoding, le codage par défaut des fontes textuelles, qui laisse un nombre important de glyphes « standards » non codés, 8*r* inventé par la communauté TEX, ce codage prend en compte tous les glyphes standards évoqués plus haut, ainsi que les glyphes standards pour TEX et présents dans les rares fontes conçues (aussi) pour ce logiciel comme Lucida (dont), 8*x* le codage des fontes « expert », 9*e* la partie du codage T1 (TEX Text ou codage de Cork pour les langues latines européennes) que l'on peut obtenir à partir d'une fonte expert (8*r* + 8*x*), 9*d* la même chose mais les chiffres étant elzéviriens (un raccourci donc pour *j*9*e*), 9*c* la partie du codage TS1 (Text Companion Symbols) que l'on peut obtenir à partir d'une fonte expert, *w* concerne les fontes « *swash* » comme notre Minion-Sw ;

E chasse (*width*) qui là encore n'a pas de sens pour la fonte MM mais en aurait pour ses avatars à choisir parmi *q* pour compressé, *c* pour condensée, ou *x* pour étendue ;

DD le corps de référence en points. Il semble que dans la pratique il ne soit pas nécessaire de préciser la notion de « point », bien que TEX et PostScript par exemple n'utilisent pas la même...

Je conclue par un exemple : la fonte MM *Minion* sera donc utilisée après avoir été copiée dans le fichier pmnMM8a.pfb, son complément expert dans pmnMM8x.pfb. L'avatar de base (romain maigre non condensé, corps de référence 10 points) sera donc défini par les amorces pmnl8a10.pss [6] et pmnl8x10.pss. La fonte virtuelle réalisée à l'étape suivante sera codée en T1 pour supporter correctement le français, elle donnera donc lieu aux fichiers pmnl8r10.tfm, pmnl8x10.tfm, pmnl9e10.tfm et pmnl9e10.vf.

4.2. Générer un ensemble de métriques au format TEX

Pour TEX, une police est une description exhaustive de quelques dimensions globales, et des dimensions de chaque caractère nécessaires pour réaliser tous les calculs aboutissant à la mise en page, cette description se trouve dans les fichiers TFM [9]. Le dessin des caractères ne concerne pas TEX proprement dit, mais ses pilotes.

4.2.1. Obtention des fichiers AFM

Les métriques des fontes PostScript étant diffusées sous forme AFM [1], il existe différents outils pour les convertir au format TFM. La première difficulté que nous rencontrons ici est d'obtenir les fichiers AFM correspondant aux avatars que nous avons décidé d'installer. En effet, Adobe ne peut diffuser la totalité des fichiers AFM possibles, mais seulement ceux correspondant aux *masters* et des fichiers décrivant l'interpolation en

6. Certains préfèrent 8*r* à 8*a* ici, ce qui relève de considérations plus politiques que pratiques.

fonction des paramètres que l'on donne à makeblendedfont pour construire les avatars. Je connais trois façons d'obtenir le fichier AFM pour un avatar donné : la première, décrite par [6], repose sur *ghostscript* et l'interprétation en PostScript de l'opérateur suscité, la seconde — plus satisfaisante, mais plus sensible à l'évolution du format MM — repose sur le script *mmblend* écrit en *perl* par Richard Walker [11], qui implémente l'algorithme d'interpolation tel que publié par Adobe. Enfin, ma solution a été d'utiliser *ATM* sous Windows qui génère des fichiers PFM[7] : j'ai ensuite utilisé le script *getmetric*[8] pour reconstituer un AFM complet à l'aide de *ghostscript* et de ce fichier PFM. L'avantage de « ma » méthode est que l'on peut sélectionner l'avatar désiré en affichant un texte d'essai dans la fenêtre d'*ATM*, et en faisant varier les différents paramètres. Cela est également possible avec l'utilitaire *Type Installer* d'Adobe qui fonctionne sous quelques dialectes d'Unix.

4.2.2. La conversion AFM vers TFM

Une fois obtenus ces fichiers, la tâche de l'installateur est de les convertir au format TFM. On pourrait très bien utiliser des fichiers TFM créés directement à partir du fichier AFM sans aucun artifice : c'est ce que font certains outils du monde TEX, soit parce qu'ils ne supportent pas les fontes virtuelles, soit parce qu'ils préfèrent reléguer les problèmes de codage au pilote PostScript. Cela pose certains problèmes car TEX ne distingue pas vraiment le codage interne dans lequel il se représente la fonte utilisée du codage de sortie, il est donc préférable de recoder les fontes utilisées selon un codage standard (T1 ou OT1 pour le texte, OML, OMS et OMX pour les maths, en attendant les nouveaux codages mathématiques sur 8 bits). On a donc recours au mécanisme des fontes virtuelles. Il existe à ce jour un seul outil (à part un éditeur Ascii...) permettant de fabriquer des fontes virtuelles composites (combinant les glyphes provenant de fontes PostScript différentes, nous en verrons quelques applications plus loin), c'est *fontinst*. C'est donc l'outil de choix pour le travail qui nous occupe.

4.2.3. Fontinst

Si nous nous contentions d'installer une famille de fonte PostScript sm, les commandes \latinfamily et \textcompfamily de *fontinst* suffiraient à générer un ensemble de polices virtuelles (OT1, T1 et TS1) couvrant l'essentiel des besoins. Dans le cas qui nous occupe, en attendant le super-gourou capable d'adapter la commande \latinfamily à toutes les situations, il faut se débrouiller avec des commandes de plus bas niveaux, mais qui sont essentiellement équivalentes, tout au moins pour la partie « standard » de l'installation. Je vais simplement montrer comment je fabriquerais les fontes T1 (pmnl9e10), T1 « old style » (pmnl9d10) et TS1 (pmnl9c10). Il est à noter que *fontinst* est capable de créer l'interface entre les fontes virtuelles qu'il assemble et la couche NFSS de LATEX, ce

7. Un format propriétaire de Windows qui ne sauvegarde que les approches de paires.
8. Disponible sur Ctan (archive t1tools).

qui allège d'autant le travail manuel. Voici les commandes qui permettent de fabriquer les fontes virtuelles en question :

```
\transformfont{pmnl8r10}{\reencodefont{8r}{\fromafm{pmnl8a10}}}
\installfont{pmnl9e10}{pmnl8r10,pmnl8x10,latin}
                                {T1}{T1}{pmnx}{m}{n}{<9.5-10.5>}
\installfont{pmnl9d10}{pmnl8r10,pmnl8x10,latin}
                                {T1j}{T1}{pmnj}{m}{n}{<9.5-10.5>}
\installfont{pmnl9c10}{pmnl8r10,pmnl8x10,textcomp}
                                {TS1}{TS1}{pmnx}{m}{n}{<9.5-10.5>}.
\installfamily{T1} {pmnx}{}
\installfamily{T1} {pmnj}{}
\installfamily{TS1}{pmnx}{}
```

Comme je l'ai déjà expliqué dans [4], la première commande est juste le recodage du fichier AFM obtenu à l'étape précédente vers un fichier MTX (métriques sauvegardées en un format optimisé pour *fontinst*) codé en 8*r*. Les commandes \installfont fabriquent un fichier VPL (permettant d'obtenir une fonte virtuelle, c.f. [9]) dont le nom est donné par le premier argument, qui est constitué en assemblant les informations métriques contenues dans les fichiers MTX listés dans le deuxième argument, et en les rangeant selon le codage (fichier ETX) spécifié par le troisième argument. Les arguments restant étant les identifiants en termes NFSS de la fonte ainsi installée. Les commandes \installfamily sont celles qui provoquent la création des fichiers NFSS de définitions de fontes FD.

La première fonte sera donc nommée pmnl9e10, elle sera codée en T1, et constituée à l'aide des glyphes des deux fontes pmnl8r10 et pmnl8x10 (le passage par le codage 8*r* rendant accessible un certain nombre de glyphes utiles en T1 mais inaccessibles en 8*a*, la fonte expert ne fournira guère que les ligatures en ff dans ce cas) selon les macros du fichier latin.mtx. Le recours à des macros fait toute la puissance de *fontinst* : il est possible de définir un glyphe de plusieurs façons en fonction des caractéristiques de la fonte ; il est possible de simuler un glyphe s'il est absent tout en l'utilisant s'il est présent. Par exemple, latin.mtx définit le caractère ž en plaçant l'accent ˇ au-dessus de la lettre z, on peut aussi définir des caractères extrêmement complexes sans difficulté.

J'ai par exemple ajouté quelques lignes dans latin.mtx qui corrigent l'espacement insuffisant par défaut entre les caractères f et è en *Minion* : fè n'est pas fè !

4.2.4. Fichiers FD

Étant donné que *fontinst* prend en charge une partie de l'installation LaTeX des nouvelles fontes, en voici tout de suite une description. LaTeX 2_ε accède aux fontes à travers une couche nommée NFSS (New Font Selection Scheme, version 2_ε). Pour fonctionner, cette couche — qui s'appuie sur une description « fonctionnelle » des fontes plutôt que sur un appel direct aux fichiers — a besoin d'un fichier définissant la correspondance entre la description des fontes en termes NFSS et les fontes (TFM) « réelles ». Pour

NFSS, une fonte est identifiée par la combinaison des paramètres suivants[9] : le codage, la famille, la « série », la forme et le corps ; la conversion entre ces paramètres (qui correspondent à l'interface utilisateur) et les fontes sera décrite dans un fichier d'extension .fd dont le nom est la concaténation du codage et de la famille.

Le code *fontinst* ci-dessus va produire les fichiers t1pmnx.fd, t1pmnj.fd et ts1pmnx.fd. Étant donné qu'on utilisera dans le monde réel les fontes dans des corps pouvant varier arbitrairement, en ajoutant les commandes relatives aux 9 corps de références, on obtient la définition suivante dans t1pmnx.fd :

```
\DeclareFontFamily{T1}{pmnx}{}
\DeclareFontShape{T1}{pmnx}{m}{n}{
   <-6.5>       pmnl9e6
   <6.5-7.5>    pmnl9e7
   <7.5-8.5>    pmnl9e8
   <8.5-9.5>    pmnl9e9
   <9.5-10.5>   pmnl9e10
   <10.5-11.5>  pmnl9e11
   <11.5-15>    pmnl9e12
   <15-36>      pmnl9e24
   <36->        pmnl9e72
}{}
```

La déclaration \DeclareFontFamily « enregistre » la famille *Minion* Expert (pmnx) auprès de LaTeX, tandis que la suivante détermine le fichier exact qui sera utilisé lorsque l'utilisateur fera appel à la police *Minion* expert codée en T1 dans la « série » *m* (médium, i.e. graisse normale, chasse normale) et la forme *n* (normale, i.e. romain) en fonction du corps requis. Par exemple, la commande

```
\fontsize{18}{22}\usefont{T1}{pmnx}{m}{n}
```

chargera la fonte pmnl9e24 rééchelonnée à un corps de 18 points (c'est-à-dire pas exactement au trois-quart de son corps optimal, car les points pica de TeX sont légèrement plus petits que les points PostScript).

4.2.5. Raffinements typographiques

Pour illustrer la souplesse et la puissance de *fontinst*, je vais décrire comment faire usage des fontes d'initiales ornées dont j'ai parlé. Ces fontes ne sont en fait conformes à aucun codage standard : elles ne contiennent en tout et pour tout qu'un alphabet complet de capitales ornées (non accentuées) $\mathcal{A}$–$\mathcal{Z}$ et les deux ligatures $\mathcal{CT}$, $\mathcal{ST}$. On pourrait évidemment installer cette fonte telle quelle, par exemple à l'aide du programme *afm2tfm* ou des commandes \afmtomtx, \mtxtopl de *fontinst*, mais le recours aux ligatures serait alors malaisé, et rendrait impossible l'utilisation de tout outil de traitement de texte

9. On pourra noter la similitude *et* la différence de cette description avec celle sous-tendant le schéma de Karl Berry. En particulier, la notion de « *series* » recouvre simultanément la graisse et la chasse, ce qui est un non sens pour toute autre fonte que *Computer Modern* telle qu'utilisée par D. Knuth (le seul changement de graisse usuel s'accompagnant d'un changement de chasse...).

usuel (correction orthographique, mise en capitales, etc.) ; de plus TEX ne coupe pas les mots qui ne sont pas composés uniquement de lettres d'une même fonte. Il est en fait beaucoup plus commode de créer une fonte virtuelle dont les capitales sont les initiales ornées, les bas de casse des italiques, et pour laquelle nous rendons automatiques les ligatures *c+t* ↦ *ct*, *s+t* ↦ *st* de la même façon que les autres ligatures déjà présentes dans toute fonte T1 comme <+< ↦ « ou f+i ↦ fi. Pour ce faire, il suffit de définir un nouveau codage, appelons-le T1aj[10] dans lequel on aura affecté une place aux deux nouveaux caractères que sont les ligatures. Ceci ne pose en fait aucun problème car il existe toujours au moins deux caractères (ŋ et Ŋ) dont le dessin ne peut être reconstitué à partir des glyphes de *Minion*. Voici la définition originale de ces caractères dans le fichier t1.etx original :

```
\setslot{\uc{Ng}{ng}}
   \comment{The Sami letter 'Ng', which looks like an uncial 'N' (or a
      large 'n') with a right tail taken from a 'j'.  It is unavailable
      in \plain\ \TeX.}
\endsetslot

\setslot{\lc{Ng}{ng}}
   \comment{The Sami letter 'ng', which looks like an 'n'
       with a right tail taken from a 'j'.  It is unavailable
      in \plain\ \TeX.}
\endsetslot
```

qu'il est évidemment trivial de supprimer au profit de

```
\setslot{ct}
 \comment{The 'old style' ligature ct.}
\endsetslot

\setslot{st}
 \comment{The 'old style' ligature st.}
\endsetslot
```

Maintenant, voici comment sont définies les ligatures usuelles ci-dessus :

```
\setslot{less}
   \ligature{LIG}{less}{guillemotleft}
   \comment{The less-than sign '$<$'.}
\endsetslot

\setslot{\lc{F}{f}}
\ifisint{monowidth}\then\else
   \ligature{LIG}{\lc{I}{i}}{\lclig{FI}{fi}}
   \ligature{LIG}{\lc{F}{f}}{\lclig{FF}{ff}}
   \ligature{LIG}{\lc{L}{l}}{\lclig{FL}{fl}}
\fi
```

10. Là comme ailleurs, le j signifie que les chiffres seront elzéviriens et non grandes capitales — ce qui serait à coup sûr une faute de goût dans une telle fonte. C'est donc de fait le codage T1j que j'altère en T1aj.

```
   \comment{The letter `{f}'.}
\endsetslot
```

Il existe plusieurs types de ligatures possibles, nous avons ici le cas le plus simple, la commande \ligature prend pour premier argument le type de ligature (en fait littéralement la commande à insérer dans le fichier VPL), pour deuxième le caractère qui va provoquer l'appel de la ligature lorsqu'il se trouve après le caractère qui est en cours de codage, et pour dernier le caractère vers lequel cette ligature pointe. Pour rendre automatique la ligature $c{+}t \mapsto$ *ct*, il suffit par conséquent de modifier de la façon suivante le codage des lettres *c* et *s* :

```
\setslot{\lc{C}{c}}
 \ligature{LIG}{\lc{T}{t}}{ct}
   \comment{The letter `{c}'.}
\endsetslot

\setslot{\lc{S}{s}}
 \ligature{LIG}{\lc{T}{t}}{st}
   \comment{The letter `{s}'.}
\endsetslot
```

et de sauver ce nouveau codage dans le fichier t1a.etx. Maintenant, la ligne suivante suffit pour installer la fonte vieux-jeu désirée en 10 points :

```
\installfont{pmnli9w10}{pmnliw10,pmnli8r10,pmnli8x10,latina}
                                  {T1aj}{T1a}{pmnw}{m}{it}{<9.5-10.5>}.
```

J'ai décidé que 9*w* représenterait mon nouveau codage dans le système de Karl Berry, le fait de charger pmnliw10 en premier assure que les capitales seront prises dans cette fonte et non dans la suivante (pmnli8r10) dans laquelle elles sont codées sous le même nom. Le fichier latina.mtx qui pointe le bout de son nez n'est autre que latin.mtx dans lequel j'ai simplement rajouté des instructions pour que la ligature *ct* hérite du crénage des lettres *c* sur la gauche, et *t* sur la droite, idem pour *st*. Il faudrait également créer un fichier t1aenc.def pour LaTeX qui se résumerait à recopier t1enc.def en remplaçant le nom de codage par T1a, la définition de \ng et \Ng par des messages d'erreur, et qui peut-être définirait des commandes pour accéder directement aux nouvelles ligatures.

On voit qu'une fonte non standard est facilement mise en place par la modification de deux ou trois fichiers. Cette fonte ne posera aucun problème particulier avec TeX (sauf évidemment pour les langues qui nécessitent les caractères ŋ et Ŋ...) car l'automatisation des ligatures rend leur utilisation parfaitement transparente. En revanche, le fait qu'elle ne comprenne que des capitales non accentuées est fâcheux (quoique historiquement justifié) : un texte saisi non spécifiquement pour l'utilisation de cette fonte et comportant des capitales accentuées aura la particularité de produire des capitales ornées pour toutes les lettres en majuscules si *et seulement si* elles sont dénuées d'accent ! De plus, les mots saisis entièrement en capitales seraient difficilement lisibles (cas où

toutes les lettres sont ornées) ou carrément anormaux (mélange des deux genres). Il est donc nécessaire[11] d'envisager des

4.2.6. Raffinements suprêmes

La version 3.0 de TEX a un introduit un mécanisme de « ligatures intelligentes » qui permet en particulier de disposer de ligatures contextuelles pour les lettres qui apparaissent en début ou en fin de mot. Le traitement des capitales dans l'exemple précédent n'étant pas idéal, il est donc possible d'envisager l'amélioration suivante : conserver le codage normal pour la fonte d'italiques, capitales et bas de casse ; ajouter les ligatures *Ct*, *St* comme précédemment ; trouver une place dans le codage où rajouter toutes les capitales ornées ; créer un nouveau codage avec ligatures automatiques en début de mot substituant pour chaque capitale (accentuée ou non) la capitale ornée correspondante (sans accent). De cette façon, les mots entièrement composés en capitales ou comportant une capitale initiale accentuée seront corrects. Le problème évident est qu'il n'y a plus de place libre dans le codage T1 ! Il faudrait donc libérer 26 emplacements, ce qui se fera forcément au détriment de quelque chose. En étant franco-centriste, on peut observer qu'une cinquantaine des glyphes accentués générés par *fontinst* sont en fait dessinables de la même façon par des macros (parce que les langues qui les utilisent sont ignorées par le standard d'Adobe) : il est possible de les retirer du codage de façon à faire de la place ; on n'aura plus ensuite qu'à créer un fichier LATEX de définition de codage qui remplace ces glyphes accentués en T1 par les macros adéquates. Évidemment, ce faisant on a un peu perdu en route l'esprit même de T1 puisque il sera dès lors impossible d'optimiser les crénages et les césures pour les langues ayant recours aux glyphes sacrifiés. Cela dit, la seule alternative serait d'utiliser un programme différent de TEX acceptant des fontes codées sur deux octets comme Ω...

Voici comment on pourrait faire en pratique :

- changer le nom des glyphes de MinionMM-Sw (par exemple remplacer *A* par *Aswash*, ce qui peut aussi bien se faire en éditant directement l'AFM, ou en intercalant un fichier MTX entre pmnliw10 et pmnli8r10, la conversion se faisant à coups de `\setglyph{Aswash}\glyph{A}{1000}\endsetglyph`) ;
- créer un nouveau codage avec tout l'alphabet à sa position normale, et les capitales ornées à une position non standard. Il faudra aussi définir un « boundarychar » qui est n'importe quel caractère qui n'apparaît jamais seul au début d'un mot. On peut choisir visiblespace (le caractère occupant la position 32) ou percent ou encore grave selon l'humeur. Une ligne comme ceci fait l'affaire : `\setint{boundarychar}{32}`.

11. Une « solution » simple est aussi de remplacer toutes les capitales accentuées par les versions ornées sans accent ; le seul inconvénient qui subsiste est alors l'illisibilité des mots entièrement composés en capitales.

- les ligatures boundarychar+A ↦ Aswash se codent simplement en les attribuant au caractère qui incarne le boundarychar (visiblespace dans mon exemple), d'où l'importance de la remarque précédente : si ce caractère devait apparaître comme première lettre d'un mot en capitales, la ligature s'opérerait ! Il est d'ailleurs probablement préférable d'utiliser la ligature de type /LIG qui effectuera une substitution : boundarychar+A ↦ boundarychar+Aswash.
- là encore, rajouter des petites choses au fichier latin.mtx peut être utile. Par exemple certaines fontes PostScript ont des approches de paires corrigées avec le caractère space. Mais l'espace — étant entièrement géré par TEX comme de la « *glue* » — n'est jamais un caractère présent dans une fonte de TEX : il peut être utile de transmettre ce crénage[12] au boundarychar pour bénéficier de ces corrections de façon équivalente, comme me l'a suggéré [10] ;
- définir le codage *ad hoc* pour LATEX.

En suivant les mêmes lignes, il est simple de remettre au goût du jour le « s long » d'avant la révolution française avec recours automatique au « s » court en fin de mot. On peut aussi automatiser le recours à d'autres lettres ornées comme certaines initiales ou finales qui sont parfois vendues dans des compléments typographiques.

Pour la portée, la diversité et la maîtrise de ces ligatures, nous reportons le lecteur à l'article de Yannis Haralambous [8]. L'objet de cet aparté était de donner une idée de la simplicité de mise en œuvre de ces mécanismes à travers les commandes mises à notre disposition par *fontinst*.

4.2.7. Raffinements extrêmes

Au point où nous en sommes, nous disposons donc de la famille[13] *Minion* aux codages TS1, T1 et T1j, plus quelques italiques d'apparat dans un codage bizarre. Il ne manque plus que des fontes mathématiques. Comme je l'ai précisé en débutant, j'ai décidé de compléter *Minion* par des glyphes d'origine *Computer Modern* pour les symboles mathématiques absents. Ceci est douteux car, si mon choix d'avatars permet de faire cohabiter ces deux familles pour ce qui est de la couleur, il n'en est pas de même des proportions, du style, ni de la répartition des graisses. Je renvoie sur ces questions aux articles [4, 5] qui donnent une idée (dans le cas SM, mais les généralisations sont évidentes) du travail qui reste à accomplir pour disposer d'un système d'édition scientifique complet. Il est aussi probable que je chercherai des substituts pour les fontes de type calligraphique proposées par *Computer Modern* et qui ne sont pas franchement l'idée qu'on se fait des lettres anglaises utilisées fréquemment en physique ou en maths

12. Ce qui est aussi simple qu'une ligne : `\setleftrightkerning{boundarychar}{space}{1000}`.

13. Par une généralisation immédiate, nous avons évidemment obtenu tous les corps de référence, les deux graisses et les « formes » standard de LATEX, soit : romain, italique, petites capitales, oblique, ainsi que petites capitales italiques pour un total d'environ 220 fontes par codage.

(un *Commercial Script* quelconque ferait l'affaire). De même, le « gras de tableau noir » de l'Ams est trop lié en terme de dessin à *CM* pour convenir, il est probable que j'essaierais celui de *Mathematical Pi*, à moins que celui de *MathTime* ne convienne mieux. Enfin, pour être complet, il nous reste à effectuer une nouvelle fois le même travail pour une famille compatible de linéales et d'un caractère de type machine à écrire (ces derniers n'étant jamais utilisés dans l'édition scientifique à proprement parler mais, je le crains, beaucoup par les informaticiens). Vu que j'ai choisi un *Minion* assez spécial, je vais avoir quelques difficultés à trouver une graisse de *Syntax* (l'une des plus belles linéales modernes, réalisant l'équilibre impossible entre la finesse, la chaleur des garaldes et la rigueur des linéales) compatible avec mes avatars, je suis donc plus ou moins condamné à utiliser une fonte MM comme *Myriad* (comme, je dois dire, le reste du suscité Landernau...). Pour ce qui est du caractère machine à écrire, je ne vois aucune raison de dédaigner celui de D. Knuth qui convient parfaitement en termes de graisse et de proportions.

4.3. Configurer LaTeX

L'installation de nouvelles fontes se fait à différents niveaux correspondants aux trois « couches » de LaTeX qui interagissent avec les fontes :

- NFSS de bas niveaux, à travers les fichiers FD, qui décrit la conversion entre la description fonctionnelle gérée par NFSS et les fichiers TFM qui devront être chargés par le « moteur » TeX ;
- prise en compte des codages à travers une définition (fichier .def) qui permettra à l'extension standard fontenc de convertir les macros Ascii vers le ou les caractères convenables ;
- mise au point d'une extension sélectionnant — éventuellement de façon configurable par options — les nouvelles fontes en remplacement des défauts, mais aussi définissant des commandes permettant d'accéder à des variantes non prévues par LaTeX (voire ingérables par NFSS, comme les petites capitales italiques...).

Bien que j'aie fait le choix de ne pas aborder de questions esthétiques ici, il ne faudrait pas perdre de vue qu'une fonte impose des contraintes sur la mise en page. Sans trop exagérer, on peut dire que la lettre *m* détermine par le rythme de ses jambages, par la forme de ses rondeurs, par l'ampleur de ses approches, l'espacement horizontal qui convient : ce dernier est fixé à la création des métriques[14], il impose un certain rapport entre « noir » et « lumières » sur la page. Par exemple, une fonte comme *Computer Modern* est très maigre, très largement espacée, le dessin des caractères faisant très largement place à la lumière : elle ne supporterait pas des marges réduites ou un interligne

14. On peut en fait le modifier depuis le fichier FD, mais ces modifications étant globales, elles sont équivalentes à la modification du fichier TFM.

insuffisant. En revanche, *Minion* (ou *Times*) est idéale pour constituer des blocs de texte denses : un interligne trop important donne une page « hachée » au gris chaotique, mais les lignes trop longues sont à proscrire avec plus de vigueur encore. Il y a donc réellement un travail de maquette qui doit intervenir après les trois étapes que je viens de citer. Ceci est un travail énorme qui dépasse de beaucoup les limites de cet article, et les compétences de son auteur.

Je pense avoir donné aux § 4.2.4, 4.2.5 des détails suffisants sur les deux premiers sujets, qui se traitent en parallèle avec la création des métriques par *fontinst*.

Il reste donc à décrire une extension dont le but est double :

- remplacer toutes les fontes utilisées dans le document par un système alternatif et cohérent. Ceci doit être transparent pour l'utilisateur, et doit préserver le balisage standard des fichiers LaTeX, de façon à rester compatible au niveau des sources, c'est-à-dire que nous devons éviter à tout prix d'introduire de nouvelles macros, mais au contraire modifier le comportement des macros standard ;
- définir de nouvelles macros pour ce qui est hors d'atteinte du système standard.

La première requête est devenue extrêmement simple grâce à LaTeX 2_ε qui ne fait jamais appel[15] à des fontes ni même à des codages explicites, elle est largement illustrée par la distribution psnfss : il suffit essentiellement d'une ligne comme `\renewcommand {\rmdefault}{pmnx}` pour que tout appel à la famille de caractères avec empattement aboutisse à *Minion* plutôt qu'à *Computer Modern tant que cette famille a été définie pour le codage requis*. Il me semble que certaines options esthétiques comme le choix de petits chiffres pour le texte doivent être globales et cohérentes entre toutes les fontes utilisées (dans la mesure du possible) : j'ai donc un peu perfectionné les extensions spartiates de psnfss comme suit.

```
%% This is file 'minion.sty', not generated
%%  with the docstrip utility (1).
\def\fileversion{1a}
\def\filedate{97/06/27}
%% File: minion.sty -thb-
\ProvidesPackage{minion}[\filedate\space\fileversion\space
 atypical  PSNFSS2e LaTeX2e package]
%% options globales
\DeclareOption{oldstyle}{\def\TheSpecialBit{j}}
\DeclareOption{lining}  {\def\TheSpecialBit{x}}
\DeclareOption{swashit} {\def\TheSpecialBit{w}%
            \renewcommand\encodingdefault{T1a}}
\DeclareOption{mathfont}{%
  \DeclareSymbolFont{letters}      {OML}{pmn}{m}{it}%
  \SetSymbolFont{letters}{normal}  {OML}{pmn}{m}{it}%
  \SetSymbolFont{letters}{bold}    {OML}{pmn}{b}{it}%
```

15. Ce qui contraste très fortement avec LaTeX-2.09 ou plain...

```
  \DeclareSymbolFont{operators}    {OT1}{pmn}{m}{n}%
  \SetSymbolFont{operators}{normal}{OT1}{pmn}{m}{n}%
  \SetSymbolFont{operators}{bold}  {OT1}{pmn}{b}{n}%
  \DeclareMathAlphabet{\mathcal}   {T1} {biO}{m}{n}%
  \SetMathAlphabet{\mathcal}{normal}{T1}{biO}{m}{n}%
    \let\vec@@ori\vec
    \renewcommand\vec[1]{\skewO\vec@@ori{#1}}%
}
\ExecuteOptions{lining}
\ProcessOptions
\renewcommand{\rmdefault}{pmn\TheSpecialBit}
\renewcommand{\sfdefault}{pmy}
\def\bfdefault{b}
%% fin des options globales
```

Il y a donc quatre options (idéalement sélectionnées au niveau \documentclass) : *lining* (défaut), *oldstyle*, *mathfont* et *swashit*. La première sélectionne *Minion* Expert (chiffres capitales : pmnx) comme fonte textuelle par défaut, et ne fait que cela ; le deuxième choisit au contraire la version dotée de chiffres elzéviriens (pmnj) ; la troisième est indépendante des deux précédentes : elle remplace les deux fontes mathématiques de type operators et letters par des fontes dans lesquelles tout ce qui pouvait être recomposé à partir de *Minion* l'a été, le reste étant en *Computer Modern*[16] (c.f. [4, 5]). On voit aussi comment remplacer les « anglaises » de *CM* par un *Commercial Script* (ici celui de Bitstream), et corriger un positionnement d'accent placé trop à droite (en attendant bien sûr de corriger la valeur du crénage correspondant dans les paramètres *fontinst*). La quatrième est beaucoup moins générale car elle va utiliser par défaut notre fonte d'italiques à capitales ornées. Enfin on sélectionne dans tous les cas la famille *Myriad* (pmy) comme complément sans empattement.

Pour fixer les idées, voici l'en-tête du source du présent article :

```
\documentclass[mathfont,oldstyle]{cah-gut}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{amsfonts,french,minion}
```

si je transmets cet article à une personne qui ne dispose pas de *Minion*, elle n'a qu'à retirer l'appel à l'extension minion, l'article sera compilé avec deux « *warnings* » mais donnera un résultat correct.

Maintenant que nous savons modifier globalement l'aspect visuel d'un document sans modifier son source, voyons comment — au prix de la compatibilité absolue — exploiter pleinement les variantes de *Minion* non prévues par LaTeX. Je recopie la seconde moitié de l'extension minion.sty :

```
%% macros spéciales
```

16. Notez que je ne touche pas aux fontes de symbols ni largesymbols qui resteront donc en *CM*.

```
\def\lining  {\fontfamily{pmnx}\selectfont}
 \let\MU@ORI\MakeUppercase
 \renewcommand\MakeUppercase[1]{{\lining\MU@ORI{#1}}}
\def\oldstyle{\fontfamily{pmnj}\selectfont}
 \let\ML@ORI\MakeLowercase
 \renewcommand\MakeLowercase[1]{{\oldstyle\ML@ORI{#1}}}
\def\itsc    {\fontshape{scit}\selectfont}
\def\scit    {\fontshape{scit}\selectfont}
\def\swashit {\usefont{T1a}{pmnw}{\f@series}{it}}
\def\ornaments{\usefont{U}{pmn}{m}{n}}
\endinput
%% End of file `minion.sty'.
```

qui permet de passer d'une fonte à l'autre localement : \lining me permettrait d'écrire certains logos correctement, je le rajoute aussi à la définition de \MakeUppercase pour pallier la déficience des codages qui ne savent pas distinguer les chiffres bas de casse des chiffres capitales... De même, \swashit permettrait d'imprimer le mot *Section* un peu plus joliment que d'habitude, \ornaments me donne un moyen d'imprimer un fleuron. Enfin, les petites capitales italiques *N'EXISTANT PAS !* dans le schéma NFSS[17], on voit que je les ai installées en les identifiant à l'aide de la « forme » hypothétique *scit* que je suis obligé d'appeler explicitement par l'une ou l'autre des commandes \scit ou \itsc. L'inconvénient majeur de ceci est qu'un texte faisant appel à ces variantes utilisé avec une autre fonte perdrait les deux attributs car NFSS substituerait à la forme inexistante *scit* la forme par défaut : généralement *n*.

À ce stade, nous avons réuni tous les ingrédients nécessaires à l'obtention d'un fichier DVI depuis un source LaTeX adapté. Il ne reste plus qu'à paramétrer les pilotes de visualisation et d'impression pour qu'ils soient capables d'afficher les glyphes magnifiques que nous achetâmes à prix d'or.

4.4. Voir, imprimer

L'UTILISATION de fontes de type 1 ne devrait pas poser de problèmes particuliers tant qu'on imprime en POSTSCRIPT[18], mais comme les pilotes sont en général paramétrés par défaut pour utiliser les *bitmaps* de *Metafont*, il faut les instruire de la démarche à suivre dès qu'on utilise une fonte d'un type différent. Cela demande de comprendre un peu comment va fonctionner le mécanisme des fontes virtuelles mis en œuvre ici. Une fonte de texte normale comme pmnl9e10 est, nous l'avons vu, une fonte virtuelle. C'est-à-dire qu'il existe un fichier pmnl9e10.tfm valide qui a permis à TeX de l'utiliser. TeX ne fait aucune différence entre ce TFM et par exemple celui de cmr10. Maintenant l'astuce du format VF est qu'un pilote sait nécessairement interpréter un code DVI, et qu'un fichier VF est à peu de choses près une collection de fragments de

17. Car les attributs « petites capitales » et « italiques » sont deux « formes », donc exclusifs l'un de l'autre.
18. Il serait judicieux de dire que c'est l'utilisation de fontes *Metafont* qui pose des problèmes dans ce cas !

code DVI — chaque « caractère » de cette « fonte » étant une suite d'instructions DVI : déplacements, choix de fontes, filets, special, etc. — si bien que la tâche du pilote se résume grossièrement à remplacer dans le DVI fourni par TEX les caractères de la fonte virtuelle par le code DVI correspondant[19], puis à traiter le DVI résultant. Cela signifie que le pilote a besoin des fichiers pmnl9e10.tfm et pmnl9e10.vf dans un premier temps, puis des fichiers TFM correspondant aux fontes (« réelles ») utilisées par la fonte virtuelle (ici pmnl8r10.tfm et pmnl8x10.tfm). À partir de ce moment, la fonte virtuelle est inutile : le pilote ne doit pas chercher à imprimer la fonte pmnl9e10 qui n'existe pas. Il cherchera les fichiers lui permettant de dessiner les caractères utilisés : dans notre cas il s'agira de la description du contour de ces caractères par la combinaison d'une fonte MM et de son amorce (pmnMM8a.pfb et pmnl8a10.pss, pmnMM8x.pfb et pmnl8x10.pss). Ce lien (entre le contour réel et les métriques correspondantes) n'étant pas inscrit dans les fichiers TFM ou VF, il faut le donner au pilote de façon explicite. De plus, la fonte de base utilisée ici est pmnl8r10.tfm (DVI) mais le contour est défini par pmnl8a10.pss au niveau PostScript : cela signifie que le codage est a priori différent. Le pilote convertissant DVI vers PostScript devra donc recoder pmnl8a10.pss avant de l'utiliser.

Les bons pilotes sont tous capables de faire ces opérations, et il y a toujours un moyen de les paramétrer pour ce faire. Dans le cas de *dvips* qui est probablement le convertisseur DVI vers PostScript le plus répandu, chaque fonte PostScript est définie par une ligne dans un fichier psfonts.map. Voici les lignes permettant d'imprimer notre *Minion* :

```
pmnl8r10 MinionMM_345_wt_600_wd_10_op "AddDotlessj \
                                    TeXBase1Encoding ReEncodeFont" \
                    <8r.enc  <pmnMM8a.pfb <dotlessj.pro  <pmnl8a10.pss
pmnl8x10 MinionMM-Ep_345_wt_600_wd_10_op <pmnMM8x.pfb <pmnl8x10.pss
```

Le premier terme est le nom de la fonte utilisée par le DVI : *dvips* cherche toujours cette fonte dans son fichier psfonts.map en premier ; s'il ne l'y trouve pas, il cherchera un *bitmap* à imprimer. Le deuxième terme est le nom PostScript de cette fonte (ici généré par *ATM* en fonction des paramètres définissant l'avatar). On dispose entre *quotes* des fragments de code PostScript qui seront utilisés au moment où *dvips* définira la fonte en question dans le fichier PostScript : ici la fonte est recodée en 8*r* comme nous l'avions noté. Pour cela la définition de ce codage (contenue dans le fichier 8r.enc) sera incluse dans le fichier PostScript : c'est ce que signifie le symbole <. On peut faire subir d'autres transformations à la fonte à ce moment là du moment qu'elles sont compatibles avec les métriques utilisées, on se contente ici de créer un caractère utile pour les fontes mathématiques [5], ce qui nécessite encore l'inclusion de l'en-tête approprié. Il ne reste plus qu'à inclure la fonte MM et son amorce (dans cet ordre) pour achever la définition complète de la fonte désirée dans le fichier PostScript. La définition de la fonte expert est évidemment plus simple car elle sera utilisée sans malice dans son codage d'origine (tous les glyphes sont codés).

19. L'utilitaire *dvicopy* remplit précisément cette tâche : il permet en particulier de produire un fichier DVI utilisable par les pilotes incapables de gérer les fontes virtuelles.

Les fontes PostScript étant essentiellement conçues pour être rendues par un interpréteur PostScript, on peut considérer que cette étape est suffisante. De fait, les visualisateurs qui savent exploiter le programme *ATM* pour les systèmes qui le supportent peuvent afficher sans difficulté les fontes PostScript en laissant *ATM* dessiner les caractères sur l'écran. Il est toutefois possible d'utiliser un utilitaire pour convertir les fontes PostScript au format PK géré de fait par quasiment tous les pilotes TEX puisque c'est le format d'utilisation des fontes *Metafont*. Il existe pour ce faire *ps2pk* aux fonctionnalités plus ou moins équivalentes à une vieille version d'*ATM*, mais qui ne gère pas les fontes MM ni les bricolages PostScript, et *gsftopk* écrit par l'excellent mathématicien Paul Vojta qui s'appuie sur *ghostscript* (donc dispose de toute la puissance d'un interpréteur PostScript récent) pour obtenir les *bitmaps*. Étant donné que *gsftopk* sait interpréter le fichier psfonts.map de *dvips*, il n'y a rien de plus[20] à faire pour pouvoir utiliser nos fontes MM dans un environnement ne parlant pas du tout PostScript ! On peut en effet imprimer avec ces *bitmaps*, ou les utiliser pour prévisualiser le DVI à l'écran avec une application comme *xdvi*[21].

4.5. Que faire de tous ces fichiers ?

L'interface que je viens de discuter se résume au bout du compte à la création et la modification d'une pléthore de fichiers. Pour qu'il soit possible de maintenir une telle installation, il est nécessaire de la structurer. D'autre part, ces fichiers sont faits pour être « vus » par les applications supposées les utiliser, il est typique qu'un même fichier doive être accessible par plusieurs applications différentes (les fichiers TFM sont a priori nécessaires à toutes les applications de la famille TEX). Là encore un minimum de standardisation ne fait pas de mal. Le standard TDS (*TEX Directory Standard*) spécifie l'organisation de tous les fichiers liés à (LA)TEX. Associé à un système performant de recherche de fichiers suffisamment général pour être partagé par tous les logiciels concernés (comme la bibliothèque *kpathsea* de Karl Berry) il conjugue universalité et efficacité.

J'estime que la version actuelle est entachée d'une erreur de conception qui est dûe au refus de certains installateurs d'implémenter le recours à une base de donnée pour la recherche des fichiers dans une vaste arborescence. En effet, les fichiers relatifs à une fonte sont dispersés à travers différentes hiérarchies relatives à leurs *types* (TFM, VF, PFB, PK, etc.), ce qui rend la maintenance et la mise à jour problématique.

Pour mesurer l'importance de ce problème, je signale que l'installation complète (y compris les *bitmaps* correspondants aux corps standards de LATEX) représente 1 354 fichiers pour un total de 15 méga octets... Voici un listing des répertoires et de ce qu'ils

20. En fait, si : il faut installer les fontes MM auprès de *ghostscript*. Ceci se résume à ajouter une ligne comme ceci dans le fichier Fontmap concerné : `/MinionMM (pmnMM8a.pfb) ;`

21. Le terminal *X* gère maintenant les fontes de type 1 SM : peut-être un jour *xdvi* sera-t-il capable d'afficher de telles fontes sans avoir recours au format PK ?

contiennent (en me limitant à pmnl9e10 pour ne pas trop encombrer...) selon TDS. J'ai noté `$TEXMF` la « racine » de l'arborescence.

```
$TEXMF/fonts/afm/adobe/minion
$TEXMF/fonts/afm/adobe/minion/pmnl8a10.afm
$TEXMF/fonts/afm/adobe/minion/pmnl8x10.afm

$TEXMF/fonts/pk/adobe/minion
$TEXMF/fonts/pk/modeless/adobe/minion
$TEXMF/fonts/pk/modeless/adobe/minion/pmnl8r10.600pk
$TEXMF/fonts/pk/modeless/adobe/minion/pmnl8x10.600pk

$TEXMF/fonts/tfm/adobe/minion
$TEXMF/fonts/tfm/adobe/minion/pmnl8r10.tfm
$TEXMF/fonts/tfm/adobe/minion/pmnl8x10.tfm
$TEXMF/fonts/tfm/adobe/minion/pmnl9c10.tfm
$TEXMF/fonts/tfm/adobe/minion/pmnl9d10.tfm
$TEXMF/fonts/tfm/adobe/minion/pmnl9e10.tfm

$TEXMF/fonts/type1/adobe/minion
$TEXMF/fonts/type1/adobe/minion/pmnMM8a.pfb
$TEXMF/fonts/type1/adobe/minion/pmnl8a10.pss
$TEXMF/fonts/type1/adobe/minion/pmnl8x10.pss

$TEXMF/fonts/vf/adobe/minion
$TEXMF/fonts/vf/adobe/minion/pmnl9c10.vf
$TEXMF/fonts/vf/adobe/minion/pmnl9d10.vf
$TEXMF/fonts/vf/adobe/minion/pmnl9e10.vf

$TEXMF/tex/latex/psfonts/adobe/minion
$TEXMF/tex/latex/psfonts/adobe/minion/8rpmnj.fd
$TEXMF/tex/latex/psfonts/adobe/minion/8rpmnx.fd
$TEXMF/tex/latex/psfonts/adobe/minion/minion.sty
$TEXMF/tex/latex/psfonts/adobe/minion/omlpmn.fd
$TEXMF/tex/latex/psfonts/adobe/minion/omspmn.fd
$TEXMF/tex/latex/psfonts/adobe/minion/ot1pmn.fd
$TEXMF/tex/latex/psfonts/adobe/minion/t1pmnj.fd
$TEXMF/tex/latex/psfonts/adobe/minion/t1pmnw.fd
$TEXMF/tex/latex/psfonts/adobe/minion/t1pmnx.fd
$TEXMF/tex/latex/psfonts/adobe/minion/ts1pmnj.fd
$TEXMF/tex/latex/psfonts/adobe/minion/ts1pmnw.fd
$TEXMF/tex/latex/psfonts/adobe/minion/ts1pmnx.fd
$TEXMF/tex/latex/psfonts/adobe/minion/upmn.fd

$TEXMF/tex/fontinst/build/adobe/minion
$TEXMF/tex/fontinst/build/adobe/minion/build_math_pmn.tex
```

```
$TEXMF/tex/fontinst/build/adobe/minion/build_pmnc.tex
$TEXMF/tex/fontinst/build/adobe/minion/build_pmnj.tex
$TEXMF/tex/fontinst/build/adobe/minion/build_pmnw.tex
$TEXMF/tex/fontinst/build/adobe/minion/build_pmnx.tex
$TEXMF/tex/fontinst/build/adobe/minion/t1a.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1aj.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1ci.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1cij.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1cj.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1i.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1iaj.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1ij.etx
$TEXMF/tex/fontinst/build/adobe/minion/t1j.etx

$TEXMF/dvips/adobe/minion
$TEXMF/dvips/adobe/minion/config.pmn
$TEXMF/dvips/adobe/minion/dotlessj.pro
$TEXMF/dvips/adobe/minion/pmn.map
```

5. Conclusion

En conclusion, j'ai tenté de donner un aperçu relativement complet[22] de l'installation et de l'utilisation de fontes PostScript avec LaTeX dans le cas à ce jour le plus difficile. J'espère ne pas avoir découragé d'enthousiasme juvénile par l'accumulation de détails techniques rébarbatifs. J'espère en revanche avoir donné suffisamment de détails pratiques pour qu'il soit possible de se lancer dans une installation comparable en suivant ma description. De nombreuses interfaces obtenues de façon similaire, ainsi que les scripts utilisés pour leur gestion se trouvent sur Ctan dans le répertoire fonts/psfonts sous la responsabilité de S. Rahtz. L'ensemble des fichiers concernant *Minion* dont j'ai parlé ici (y compris les fichiers de paramètres pour *fontinst* ou LaTeX) devrait également trouver à terme son chemin vers Ctan[23].

22. Je n'ai pas traité de plate-formes très générales par méconnaissance, mais une très large part de ce que j'ai décrit est indépendant de la plate-forme. Certains produits commerciaux comme Y&YTeX disposent d'outils d'installation et de manipulation de fontes dont l'ergonomie est sans commune mesure avec ce que j'ai décrit. En particulier le support pour les fontes MM est assez transparent, mais ces outils ne gèrent pas les fontes virtuelles composites comme *fontinst* : leur portée est plus limitée.

23. On peut déjà consulter une version préliminaire à l'Url ftp://fourier.ujf-grenoble.fr/pub/contrib-tex/psfonts/adobe. Ces fichiers sont soumis à une licence qui en interdit l'usage professionnel ou commercial, *y compris sans but lucratif*.

Références

[1] Jacques André & Justin Bur « Métrique des fontes PostScript », *Cahiers GUTenberg* n° 8 (29–49) mars 1991.

[2] Jacques André « Caractères numériques : introduction », *Cahiers GUTenberg* n° 26 (3–44) mai 1997.

[3] Karl Berry « Filename for Fonts », document disponible sur Ctan.

[4] Thierry Bouche « Sur la diversité des fontes mathématiques », *Cahiers GUTenberg* n° 25 (1–24) novembre 1996.

[5] Thierry Bouche & Bernard Desruisseaux « Comment ne pas mettre les points sur les », à paraitre aux *Cahiers GUTenberg*.

[6] Robin Fairbairns, Michel Goossens, Sebastian Rahtz, « Using Adobe Type 1 Multiple Master Fonts with TEX », *TUGboat* vol. 16 n° 3 (253–258) juin 1995.

[7] Michel Goosens, Sebastian Rahtz, Frank Mittelbach « The LATEX Graphic Companion », Addison-Wesley, Reading, USA, 1997.

[8] Yannis Haralambous « Tour du monde des ligatures », *Cahiers GUTenberg* n° 22 (87–99) septembre 1995.

[9] Denis Roegel « Formats des fichiers associés à TEX », *Cahiers GUTenberg* n° 26 (71–95) mai 1997.

[10] Hilmar Schlegel *Communications personnelles*, 1996–1997.

[11] Richard Walker *Communications personnelles*, 1996.