# Tout savoir sur les fontes avec (LA)TEX

Thierry Bouche

Transparents du tutoriel donné le 18 mai 1999 à l'IPNL, dans le cadre des journées GUT'99.

# *Plan*

1. Notions de fontes :
   (a) codage,
   (b) métriques,
   (c) glyphes.
2. Moteur TEX : TFM et *codes*.
3. LATEX, les trois couches du NFSS :
   (a) utilisateur,
   (b) programmeur,
   (c) interfaçage avec virtex.
4. Installer de nouvelles fontes :
   (a) Métriques des fontes réelles ou virtuelles.
   (b) support NFSS,
5. Imprimer : configuration des pilotes.

# *Qu'est-ce qu'une fonte ?*

- Pour les programmes, selon ce qu'ils en font :
  - un ensemble de *métriques* :
    - AFM pour les fontes PS,
    - TFM pour TEX ;
  - une collection de *glyphes* :
    - PFB ou PFA pour les fontes PS,
    - PK, format *bitmap* par défaut dans le monde TEX ;
  - un codage :
    - « vecteur de codage » pour les fontes PS,
    - « codage de sortie » pour LATEX.
- Pour l'utilisateur : un attribut visuel, i.e. des *variantes* permettant de structurer le texte :
  - romain : la référence,
  - **gras**,
  - *italique*,
  - PETITES CAPITALES,
  - corps plus ou moins grand.

# *Codage*

Un codage, c'est une numérotation des caractères disponibles dans une fonte. Il existe des codages de *caractères* comme ASCII, ISO-latin-1, Unicode,...et des codages de *glyphes*.

- Les fontes PS on un vecteur de codage, qui peut être modifié (donne une correspondance entre numéro d'orde dans la police et nom de glyphe) ;
- LaTeX distingue le *codage d'entrée* (codage disponible au clavier de l'utilisateur, qui dépend de la plate-forme) et *codage de sortie* : la façon dont les glyphes sont disposés dans la fonte avec laquelle on imprime. Comme LaTeX n'est pas *Wysiwyg*, une fonte peut contenir 256 glyphes (pas de caractères de contrôle ou de caractères réservés).

# *Métriques*

Les *métriques* rassemblent toute l'information descriptive qui permet au programme de gérer les C&J.

Les métriques peuvent contenir

- des informations globales :
  - œil ou hauteur des lettres courtes ;
  - hauteur des capitales, hampes ou jambages, largeur de l'espace : uniquement AFM,
  - espace inter-mot en terme de *glue* TEX, « corps de référence » : uniquement TFM,
- des informations individuelles pour chaque caractère :
  - chasse,
  - correction italique : uniquement dans le TFM,
  - *Bounding Box* : uniquement AFM,

  mais aussi
- des crénages ;
- des ligatures ;
- des propriétés spéciales comme celles des fontes mathématiques de TEX.

## *Glyphes*

Ils servent finalement assez peu : ils n'interviennent qu'au dernier maillon de la chaîne de production. TEX les ignore, ce sont uniquement les pilotes, qui convertissent le DVI en un format imprimable ou visualisable, qui ont à s'en préoccuper. Par défaut, les installations de TEX sont configurées pour utiliser les *bitmaps* PK produits par *MetaFont*. Selon le medium d'impression, il est toutefois possible d'utiliser des formats vectoriels (PS type 1 ou 3, True Type).

# *Moteur TEX*

Au niveau du moteur TEX, une police est un fichier TFM, qui fournit les informations métriques nécessaires à la composition. Nous allons voir que le format TFM est un peu plus complet que les autres formats de métriques comme AFM : les décisions typographiques concernant une police trouvent leur place dans le TFM, TEX s'adaptant à toutes les situations.

Voici comment on utilise une fonte :

```
\font\mafonte=cmr12
\font\mafonte=cmr11 scaled 1100
\font\mafonte=cmr10 at 12pt
```

Ceci

- charge le TFM de cmr12/11/10 en mémoire ;
- définit la commande \mafonte :
  a) on utilise cmr12 tel quelle (attention, comme nous allons le voir, les fontes de TEX ont un « corps de référence » : ici douze points, cette fonte sera par conséquent utilisée en 12pt) ;

b) on utilise cmr11 (corps de référence : 11pt) rééchelonnée par un facteur 1 100 ‰, donc à 12,1pt.
c) on utilise cmr10 (corps de référence : 10pt) « zoomée » à 12pt,

*Du fait que les points de TEX sont plus petits que les points PS ou DTP, aucune de ces fontes ne donnera du CMR en « douze points » au sens usuel !*

On peut alors l'utiliser de cette façon :

```
{\mafonte xxxxxxx\par}
```

*Il est important d'inclure un changement de fonte dans un groupe, un certain nombre de paramètres typographiques ne changeront que si l'on compose un alinéa entier.*

Le TFM fournit les informations suivantes (polices de texte) :

☛ Les dimensions globales, ou *fontdimen* (je donne les valeurs pour cmti12, voir le fichier cmti12.pl) :

1) le taux d'inclinaison des italiques (ça n'est pas un angle, comme dans l'AFM, mais un *pourcentage* : 25 %),
2) l'espace intermot idéal (0,350 003 cadratin),
3) l'espace maximale que TEX a le droit d'ajouter à un intermot (0,150 002 cadratin),
4) l'espace maximale qu'il pourra retrancher (0,100 001 cadratin),
5) l'œil ou hauteur d'*x* (0,430 556 cadratin), c'est la valeur utilisée pour l'unité `ex`,
6) le « cadratin » (1,000 010 5 cadratin !), c'est la valeur utilisée pour l'unité `em`,
7) l'« espace supplémentaire » qui n'a a priori pas d'intérêt pour nous : il s'agit de l'espace que les anglais ajoutent à l'espace mot après une ponctuation. `\frenchspacing` supprime ce comportement (ici 0,100 001 cadratin).

Ces dimensions sont directement accessibles depuis TEX, et même modifiables, comme n'importe quelle dimension, pourvu qu'on n'ait pas encore utilisé ces paramètres :

```
\font\mafonte=cmr10
%  affiche l'intermot optimal
\message{\the\fontdimen2\mafonte}
% annule \fontdimen2
\fontdimen2\mafonte=0pt
%  affiche l'intermot optimal
\message{\the\fontdimen2\mafonte}
% vérifions :
\mafonte a b c \bye
```

En fait, il existe \xspaceskip et \spaceskip pour modifier directement l'intermot localement : voir la définition de \raggedright en *plain*.

Mais c'est le seul moyen de modifier d'autres dimensions, par exemple celles qui contrôlent le placement des indices en mode mathématique.

☛ Chaque caractère a les attributs suivants dans le TFM :

– dimensions de la boîte prise en compte par TEX : chasse, hauteur et « profondeur » (pas de notion de talus d'approche ou de *Bounding Box*),
– correction italique éventuelle (produite par `\/` : comparer *X*D et *x*D, *X*D et *x*D,
– instructions de crénage de paires,
– ligatures (substitution automatique qui n'inhibe pas les divisions).

On ne peut pas modifier ces attributs. Si on veut accéder à ces valeurs, il suffit de mettre le caractère dans une boîte, et de la mesurer. Pour le crénage ou la correction italique, on peut comparer les chasses de deux boîtes.

Étudier quelques fichiers PL, qui sont une forme lisible des fichiers TFM. Déterminer tous les espaces insérés par TEX, et le pourquoi des substitutions entre

```
``!` Avec  --- ou {\it sans\/} -- effort~!''
```

« ¡ Avec — ou *sans* – effort !»

# *Fontes mathématiques*

Les formules mathématiques sont composées en mode mathématique, très différent du mode texte. Les TFM des fontes mathématiques sont donc assez spéciaux. En général, l'espace mot est nul, la chasse est erronée, car c'est elle qui sert à placer les indices, la chasse prise en compte par TEX (en particulier pour placer les exposants) étant la somme de la chasse et de la correction italique. Pour placer les accents, il faut utiliser la commande \mathaccent car la commande \accent les placerait trop à gauche ! En fait, la commande \mathaccent centre l'accent, non pas sur la chasse déclarée du caractère, mais en le déplaçant vers la droite de la valeur donnée comme « approche de paire » avec un caractère spécial : le \skewchar qu'il faut définir pour chaque fonte d'italiques mathématiques : ce crénage n'en est donc pas un ! La valeur de l'ex de la fonte en cours est utilisée pour le placement *vertical* des accents.

Il existe trois grandes familles de fontes mathématiques : les italiques, les symboles et les grands symboles. Les deux dernières ont une vingtaine de \fontdimen, et des instructions spéciales dans le TFM (`nextlarger`). (Regarder le PL et les tables de cmmi10, cmsy10 et cmex10.)

```
(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)
   )
```

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

# *Codes...*

En mode texte, TEX se contente de recopier le code (8 bits) du caractère saisi dans le DVI : on imprime donc le glyphe qui se trouve à cette place dans la fonte, ce qui est sans surprise avec les lettres non accentuées, mais plus étonnant si on imprime un « é » PC ou UNIX avec une fonte de Mac, ou un « \ » en cmr10 ! Pour cela, un système de commandes avec noms symboliques, ou le recours à l'extension LATEX inputenc résout le problème. En mode texte, les seuls caractères « bizarres » sont ceux qui sont un \catcode différent de 11 (lettre). inputenc rend *actifs* tous les caractères au-delà de 128.

En mode mathématique, les choses sont très différentes : il faut imaginer que tout ce que l'on saisit est une commande, que ce soit une macro ou une lettre. Les caractères ont un \mathcode composé de 4 chiffres hexadécimaux, qui détermine entièrement la façon dont ils seront traités :

– Il existe 16 familles de fontes mathématiques au plus, qu'on introduit avec ce genre de choses (cf. plain.tex) :
```
\textfont0=\tenrm\scriptfont0=\sevenrm
\scriptscriptfont0=\fiverm
```

Il est nécessaire de disposer de familles pour que la gestion des indices et exposants soit cohérente dans une formule.

– chaque signe entré en mode mathématique est membre d'une « classe » parmi 8, qui contrôle son espacement. Le \mathcode est simplement le nombre formé de l'identifiant de la classe, puis de la famille, puis du code hexa du caractère à imprimer dans cette famille.

Tous les membres d'une famille doivent donc être codés de la même façon, et on a souvent des surprises...

Regarder plain.tex, compiler ceci et comprendre pourquoi ça donne un *moins* à la place d'un *trait-d'union* (et un prime en prime !) :

```
$$\rm trait-d'union$$
```

$$\rm trait-d'union$$

# *LaTeX & NFSS*

LaTeX permet d'interfacer le « désir » du typographe en convertissant une série d'attributs « orthogonaux » en une requête précise d'un TFM pour le moteur TeX.
NFSS a deux faces :

☛ l'interface utilisateur :
`\fontsize{`*corps*`}{`*interligne*`}`
`\usefont{`*codage TeX*`}{`*famille*`}{`*série*`}{`*forme*`}`

**codages de texte** OT1 (CM, défaut), T1 (EC), OT2 (cyrillique), LGR (grec) ;

**codages mathématiques** OML (lettres italiques + grec), OMS (symboles), OMX (symboles extensibles), U (*unknown* : non défini) ;

**famille** 20 000 à 30 000 possibilités aux formats *TrueType*, PS *Type 1*, *Metafont*...

**« série »** correspond simultanément aux variations de graisse et de chasse. Valeurs courantes : *m* (medium, défaut), *b* (gras), *bx* (gras étendu) ;

**« forme »** romain (*n*, défaut), italique (*it*), petites capitales (*sc*).

On dispose pour chaque variante, de trois moyens (au moins !) de modifier l'attribut correspondant :

```
{\itshape italiques}
{\fontshape{it}\selectfont italiques}
\textit{italiques}
\begin{textit}
  italiques
\end{textit}
{\it italiques}
```

La dernière version est équivalente à \normalfont\itshape si on n'a pas chargé l'extension newlfont.

Voici les variantes disponibles en standard (plus d'infos dans le fichier ltfntcmd.dtx de la distribution LaTeX) :

```
\DeclareTextFontCommand{\textrm}{\rmfamily}
\DeclareTextFontCommand{\textsf}{\sffamily}
\DeclareTextFontCommand{\texttt}{\ttfamily}
\DeclareTextFontCommand{\textnormal}
                                {\normalfont}
\DeclareTextFontCommand{\textbf}{\bfseries}
\DeclareTextFontCommand{\textmd}{\mdseries}
\DeclareTextFontCommand{\textit} {\itshape}
\DeclareTextFontCommand{\textsl} {\slshape}
\DeclareTextFontCommand{\textsc} {\scshape}
\DeclareTextFontCommand{\textup} {\upshape}
\DeclareTextFontCommand{\emph}   {\em}
```

On manipule les attributs, soit individuellement à l'aide des commandes déjà décrites, soit tous à la fois grâce à \usefont, soit enfin en spécifiant exactement la valeur à changer comme ceci :

```
\fontsize{10}{20}
\fontencoding{T1}
\fontfamily{pmn}
\fontseries{b}
\fontshape{scit}
```

*Attention ! Aucune de ces modifications n'est prise en compte tant que LaTeX n'a pas rencontré \selectfont.*

En fait, \selectfont convertit l'ensemble d'attributs en un nom de fonte, ici : \T1/pmn/b/scit/10 et le mécanisme de bas niveau du NFSS va sélectionner le TFM chargé par \selectfont.

# *NFFS & Maths*

Il n'y a quasiment pas d'interface utilisateur pour les fontes en mode maths. La configuration est faite ailleurs, dans fontmath.ltx ou dans une extension. On dispose des mêmes alphabets, mais sous le nom \mathrm,bf,sf,tt,cal. Pour saisir du texte en maths, utiliser la commande \text d'AMSLaTeX, qui fournit aussi \mathbb,frak.

Comme NFSS charge les fontes à la demande, il est souvent difficile de savoir quelle est la valeur courante de \textfont1 par exemple. Nous verrons que ces familles sont manipulées sous des noms symboliques (symbols, letters,...), leur utilisation est néanmoins essentiellement équivalente à ce qui se passe en *plain*.

☛ Lorsque LaTeX rencontre la commande \selectfont, un TFM est chargé. Pour ce faire, LaTeX doit convertir la requête « qualitative » de l'utilisateur en nom de fichier TFM.

Le mécanisme est le suivant : si la fonte est \T1/pmn/m/n/10, LaTeX va chercher le fichier t1pmn.fd (dont le nom est donc la concaténation du nom du codage, et de celui de la famille). Ce fichier doit contenir des « déclarations » qui déterminent le TFM correspondant aux spécifications. Le cas échéant, une variante existante sera substituée (le corps le plus proche si seulement certains corps sont disponibles, la forme *n* si la forme demandée n'existe pas dans la graisse donnée (exemple fréquent des petites capitales grasses), etc.).

## *Variables internes*

NFSS conserve une représentation interne de chaque attribut en cours d'utilisation : il s'agit, respectivement, de

```
\f@size           (10)
\f@baselineskip   (20)
\f@encoding       (T1)
\f@family        (pmn)
\f@series          (b)
\f@shape        (scit)
```

*Attention ! les deux premières ne sont pas des dimensions, simplement le résultat d'un* `\def\f@size{10}`.

# *Programmation*

LaTeX définit un certain nombre de défauts : il suffit de les redéfinir pour modifier toute les fontes utilisées dans le document. S'il n'y a pas eu de déclaration spécifique dans le fichier, ce sont les défauts qui sont utilisés : ils sont définis par LaTeX ou dans une extension adéquate. Voici comment ils sont mis en place (pour le texte) :

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}
\newcommand\rmdefault{cmr}
\newcommand\sfdefault{cmss}
\newcommand\ttdefault{cmtt}
\newcommand\bfdefault{bx}
\newcommand\mddefault{m}
\newcommand\itdefault{it}
\newcommand\sldefault{sl}
\newcommand\scdefault{sc}
\newcommand\updefault{n}
\newcommand\encodingdefault{OT1}
\newcommand\familydefault{\rmdefault}
\newcommand\seriesdefault{\mddefault}
\newcommand\shapedefault{\updefault}
```

## *Extensions NFSS*

Pour modifier la famille ou le codage utilisés dans tout le document, il suffit de redéfinir (avec \renewcommand) ces défauts. Exemple :

```
\ProvidesPackage{times}[\filedate\space
        \fileversion\space
        Times PSNFSS2e package]
\renewcommand{\sfdefault}{phv}
\renewcommand{\rmdefault}{ptm}
\renewcommand{\ttdefault}{pcr}
\endinput
```

LaTeX émet un \selectfont au \begin{document}, où il sélectionne la fonte par défaut : c'est souvent à ce moment-là qu'un système mal configuré est révélé.

Pour les maths, il y a par défaut 4 polices de symboles qui sont introduites par les déclarations suivantes :

```
\DeclareSymbolFont{operators}{OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}{OML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}{OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
\SetSymbolFont{operators}{bold}{OT1}{cmr}{bx}{n}
\SetSymbolFont{letters}{bold}{OML}{cmm}{b}{it}
\SetSymbolFont{symbols}{bold}{OMS}{cmsy}{b}{n}
\DeclareSymbolFontAlphabet{\mathrm}{operators}
\DeclareSymbolFontAlphabet{\mathnormal}{letters}
\DeclareSymbolFontAlphabet{\mathcal}{symbols}.
```

Par défaut, LaTeX utilise la police operators (cmr) pour

- les chiffres (0–9) ;
- les petits délimiteurs (parenthèses, crochets, etc.) ;
- quelques signes de ponctuation comme ; : ;
- les lettres capitales grecques ;
- la plupart des accents ;
- les signes + = .

Par défaut, LaTeX utilise la police letters (cmmi) pour

- les lettres (non accentuées) « normales » ;
- quelques rares signes de ponctuation comme , . ;
- les lettres grecques italiques ;
- quelques symboles de type lettre très utiles comme \imath ($\imath$), \jmath ($\jmath$), \ell ($\ell$), \partial ($\partial$), quelques « harpons » comme \leftharpoonup ($\leftharpoonup$) ;
- quelques symboles relativement inutiles comme \smile ($\smile$) ;
- l'unique « accent » qui ne se trouve pas dans une fonte de texte \vec ($\vec{}$).

Les deux familles symbols et largesymbols correspondent aux grand opérateurs et aux symboles de taille variable, comme \prod ($\prod$), \int ($\int$), etc.

# *Étude de fontmath*

```
\DeclareMathAlphabet      {\mathbf}{OT1}{cmr}{bx}{n}
\DeclareMathAlphabet      {\mathsf}{OT1}{cmss}{m}{n}
\DeclareMathAlphabet      {\mathit}{OT1}{cmr}{m}{it}
\DeclareMathAlphabet      {\mathtt}{OT1}{cmtt}{m}{n}
\SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
\SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}

 \DeclareMathSizes{5}{5}{5}{5}
 \DeclareMathSizes{6}{6}{5}{5}
 \DeclareMathSizes{7}{7}{5}{5}
 \DeclareMathSizes{8}{8}{6}{5}
 \DeclareMathSizes{9}{9}{6}{5}
 \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
 \DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
 \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
 \DeclareMathSizes{\@xviipt}{\@xviipt}{\@xiipt}{\@xpt}
 \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
 \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xviipt}

\DeclareMathSymbol{a}{\mathalpha}{letters}{‘a}

\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
\DeclareMathSymbol{)}{\mathclose}{operators}{"29}

\DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast

\DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}

\DeclareMathSymbol{.}{\mathord}{letters}{"3A}
```

```
\DeclareMathSymbol{=}{\mathrel}{operators}{"3D}

\DeclareMathDelimiter{(}{operators}{"28}
                                {largesymbols}{"00}
\DeclareMathDelimiter{)}{operators}{"29}
                                {largesymbols}{"01}

\DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
    \def\int{\intop\nolimits}

\DeclareMathAccent{\hat}{\mathalpha}{operators}{"5E}
\DeclareMathAccent{\vec}{\mathord}{letters}{"7E}

\DeclareMathRadical{\sqrtsign}{symbols}{"70}{largesymbols}{"70}

\def\operator@font{\mathgroup\symoperators}
\thinmuskip=3mu
\medmuskip=4mu plus 2mu minus 4mu
\thickmuskip=5mu plus 5mu
```

Autres exemples très utiles : mtime.sty et lucidabr.sty, qui gèrent des variantes supplémentaires, et des codages 8 bits.

# *Passerelle NFSS – TFM*

☛ Voici un exemple de fichier FD (*font definition*) :

```
\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
}{}
```

## « *Size functions* »

La commande \DeclareFontFamily a trois arguments : le codage, le nom de la famille. Le troisième argument contient du code qui sera exécuté après la définition de chaque fonte de cette famille, comme \skewchar\font=-1 voire \fontdimen2\font=1.6\fontdimen2\font ou tout autre code farfelu.

La commande \DeclareFontShape a six arguments : les quatre premiers sont exactement les attributs demandés par \usefont, l'objet de cette commande est donc de donner un tableau de correspondance entre corps et TFM, ce qui est précisément le cinquième argument. Le sixième permet d'exécuter un code spécifique au chargement de *ce* TFM.

Nous allons regarder des exemples de ce cinquième argument, qui fait intervenir les « *size functions* ».

```
\DeclareFontShape{OT1}{cmr}{m}{n}%
     {<5><6><7><8><9><10><12>gen*cmr%
      <10.95>cmr10%
      <14.4>cmr12%
      <17.28><20.74><24.88>cmr17}{}
\providecommand{\EC@family}[5]{%
  \DeclareFontShape{#1}{#2}{#3}{#4}%
  {<5><6><7><8><9><10><10.95><12><14.4>%
   <17.28><20.74><24.88>genb*#5}{}}
\EC@family{T1}{cmr}{m}{n}{ecrm}
\DeclareFontFamily{OML}{cmm}{\skewchar\font127 }
\DeclareFontShape{OML}{cmm}{m}{it}
    { <5> <6> <7> <8> <9> gen * cmmi
      <10><10.95>cmmi10
      <12><14.4><17.28><20.74><24.88>cmmi12
      }{}
\DeclareFontFamily{OML}{cmr}{\skewchar\font127 }
\DeclareFontShape{OML}{cmr}{m}{n}%
   {<->ssub*cmm/m/it}{}
\DeclareFontShape{T1}{ptm}{b}{n}{
   <-> ptmb8t
}{}
\DeclareFontShape{T1}{punut}{b}{n}{
   <-> s * [.88]punb8t
}{}
```

# *Exemples de codages*

Voici une collection de glyphes (Times romain, codage AdobeStandardEncoding) :

! " # $ % & ’ ( ) * + , - . /

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 § ¤ ' “ « ‹ › fi fl

– † ‡ · ¶ • ‚ „ ” » … ‰ ¿

` ´ ˆ ˜ ¯ ˘ ˙ ¨ ˚ ˝ ˛ ˇ

—

Æ ª Ł Ø Œ º

æ ı ł ø œ ß

- ↘ TEX utilise un codage interne pour ses algorithmes de césures : il doit être cohérent avec celui du TFM.
- → TEX n'opère les césures qu'à l'intérieur d'une même fonte : le choix des glyphes codés ou non est donc crucial en fonction des lettres utilisées pour la langue considérée.
- ↗ De même les crénages et ligatures automatiques se font à l'intérieur d'une même fonte.
- → Une fonte virtuelle peut être employée pour changer de codage entre celui vu par TEX et celui que verra le pilote d'impression.
- → On peut ainsi utiliser tous les glyphes d'une fonte PS en la codant en 8r, et la recoder en OT1 ou T1 pour que (LA)TEX sache s'en servir.
- → Il existe des codages nécessairement non standardisables (dingbats, « casseaux » de variantes, jeux de caractères complémentaires...).

Times romain, codage 8r (a priori pas utilisée par TEX) :

˙ fi fl / ˝ Ł ł ¸ ° ˇ – Ž ž

ˇ ı ` '

! " # $ % & ’ ( ) * + , - . /

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 „ … † ‡ ˆ ‰ Š ‹ Œ

“ ” • – — ˜ ™ š › œ Ÿ

¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯

° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿

À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï

Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß

à á â ã ä å æ ç è é ê ë ì í î ï

ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

Times romain, codage T1 (virtuelle, avec les glyphes « réels » de 8r) :

` ´ ^ ~ ¨ ˝ ° ˇ ˘ ¯ ·

¸ ˛ ‚ ‹ ›

“ ” „ « » –— ■ ı ▮ ff fi fl ffiffl

␣ ! " # $ % & ’ ( ) * + , - . /

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 { | } ~ -

Ă Ą Ć Č Ď Ě Ę Ğ Ĺ Ľ Ł Ń Ň ■ Ő Ŕ

Ř Ś Š Ş Ť Ţ Ű Ů Ÿ Ź Ž Ż IJ İ đ § 

ă ą ć č ď ě ę ğ ĺ ľ ł ń ň ■ ő ŕ

ř ś š ş ť ţ ű ů ÿ ź ž ż ij ¡ ¿ £

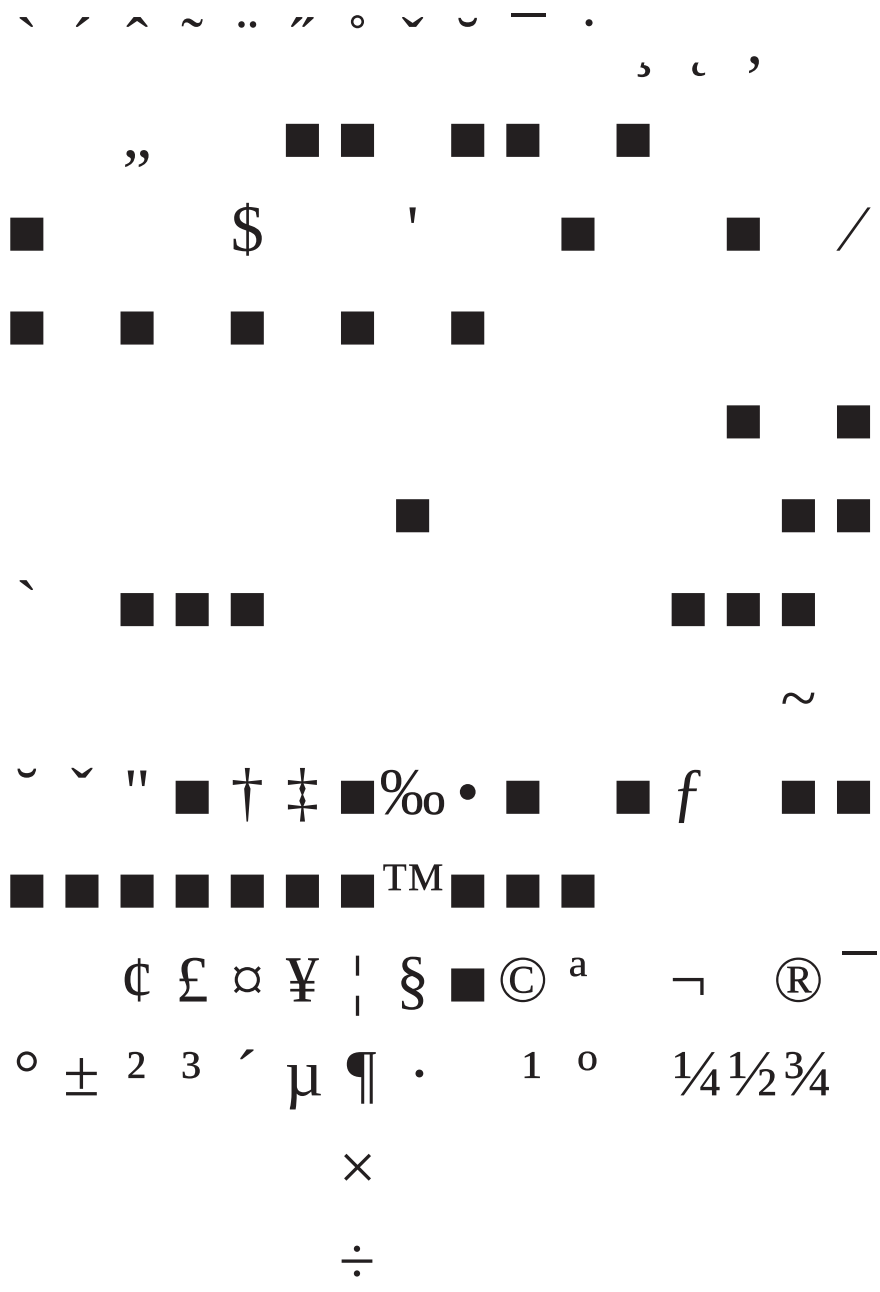
À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï

Ð Ñ Ò Ó Ô Õ Ö Œ Ø Ù Ú Û Ü Ý Þ SS

à á â ã ä å æ ç è é ê ë ì í î ï

ð ñ ò ó ô õ ö œ ø ù ú û ü ý þ ß

Times romain, codage TS1 (virtuelle, avec les glyphes « réels » de 8r) :



Times romain, codage OT1 (virtuelle, glyphes manquants : Symbol) :

Γ Δ Θ Λ Ξ Π Σ ϒ Φ Ψ Ω ff fi fl ffiffl

ı ▮ ` ´ ˇ ˘ ¯ ° ¸ ß æ œ ø Æ Œ Ø

■ ! ” # $ % & ’ ( ) * + , - . /

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 [ “ ] ˆ ˙

‘ 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 – — ˝ ˜ ¨

Utopia romain, codage 8r :

˙ fi fl / ″ Ł ł ¸ ° ˘ – Ž ž

ˇ ı ` '

! " # $ % & ’ ( ) * + , - . /

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 „ … † ‡ ˆ ‰ Š ‹ Œ

“ ” • – — ˜ ™ š › œ Ÿ

¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯

° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿

À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï

Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß

à á â ã ä å æ ç è é ê ë ì í î ï

ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

Utopia romain, codage 8x (complément expert) :

Utopia romain, codage T1 (virtuelle, avec les glyphes « réels » de 8r et 8x) :

` ´ ^ ~ ¨ ˝ ° ˇ ˘ – ·

‚ ¿ ¸ ‹ ›

“ ” „ « » – — ■ ı ▮ ff fi fl ffi ffl

␣ ! " # $ % & ’ ( ) * + , - . /

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 { | } ~ -

Ă Ą Ć Č Ď Ě Ę Ğ Ĺ Ľ Ł Ń Ň ■ Ő Ŕ

Ř Ś Š Ş Ť Ţ Ű Ů Ÿ Ź Ž Ż IJ İ đ §

ă ą ć č ď ě ę ğ Í ľ ł ń ň ■ ő ŕ

ř ś š ş ť ţ ű ů ÿ ź ž ż ij ¡ ¿ £

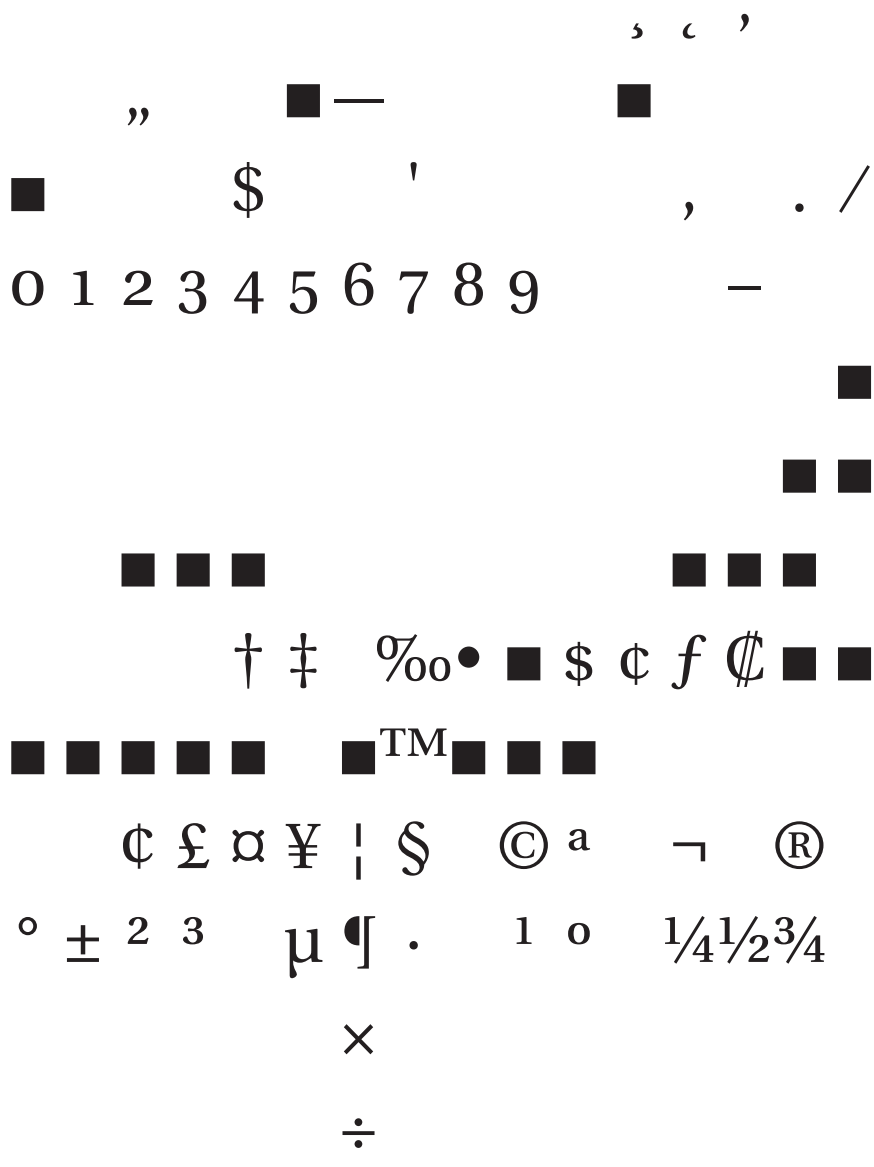
À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï

Ð Ñ Ò Ó Ô Õ Ö Œ Ø Ù Ú Û Ü Ý Þ SS

à á â ã ä å æ ç è é ê ë ì í î ï

ð ñ ò ó ô õ ö œ ø ù ú û ü ý þ ß

Utopia romain, codage TS1 (virtuelle, avec les glyphes « réels » de 8r et 8x) :



## *Installer de nouvelles fontes*

L'installation se décompose comme suit :

- 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.

# *Les noms selon Karl Berry*

Une fonte est identifiée par une suite de caractères :

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

**S** vendeur (*supplier*, Adobe = p) ;

**TT** fonte (*typeface*, Minion = mn) ;

**W** graisse (*weight*), maigre = l, normal = r semi-gras = s ou d, gras = b, MM = ( ?) MM ;

**V**$_i$ variantes, concept flou. Romain = r, italique = i, petites capitales = c, chiffres elzéviriens = j ;

**N** codage (*encoding*), Adobe Standard Encoding = 8a, Adobe Expert 8x, 8r, T1 = 8t, OT1 = 7t, TS1 = 8c, 9e = 8t + 8x, 9d = 8t + j, 9c = 8c + 8x, *swash* = w ;

**E** chasse (*width*) compressé = q, condensé = c, étendu = x ;

**DD** le corps de référence en points.

## *Donc*

**putr8a** est le nom de Karl BERRY de la fonte Utopia-Regular dans son codage d'origine ;

**putr8r** la version recodée pour disposer de tous les glyphes ;

**putr8t** la version recodée en T1, prête pour une utilisation avec TEX.

**putr8x** le complément expert, dans son codage d'origine ;

**putr9e** la version recodée en T1, utilisant les glyphes de putr8r et putr8x ;

**pmnl9d24** l'avatar maigre, chiffres elzéviriens, chasse moyenne utilisé ici : il s'agit d'une police virtuelle codée en T1 qui fait appel aux glyphes définis par **pmnl8r24.pss** et **pmnl8r24.pss**.

## *Étapes pratiques de l'installation*

- Fichiers de métriques PS (AFM) ;
- AFM $\mapsto$ TFM : *fontinst* (il existe d'autres possibilités : *afm2tfm*, *AfmToTfm*...)
  - commandes standard,
  - adaptation au cas MM,
  - cas particuliers (initiales ornées, ligatures inhabituelles),
  - fontes mathématiques,
  - interaction avec LaTeX ;
- configurer LaTeX ;
- configurer les pilotes ;
- un peu de rangement (TDS).

- ➶ Les métriques de fontes PS sont au format AFM. Le fichier des glyphes (PFB) permet de retrouver les métriques des caractères, pas les décisions « esthétiques » (ligatures, crénages, composites).
- → Si on n'a pas l'AFM voulu, on peut donc en recréer un incomplet à partir du PFA ou PFB. Cela nécessite un interpréteur PS. (imprimante, DPS ou GS).
- ➜ Sous windows, un PFM contient ces informations (généré par ATM par exemple), il existe des programmes :
  - *pfm2afm* (pas de BB),
  - *getmetric*.

  Dans tous les cas, l'AFM n'est pas aussi bon que l'original.
- → Dans le cas des fontes MM, on a un AMFM, et les AFM des *masters*. Il faut fabriquer un AFM pour chaque avatar.
  - *ATM* génère un PFM (cf. ci-dessus...) ;
  - *mminstance* : deux programmes, l'un calcule l'AFM, l'autre crée un avatar (*snapshot*).

† *Fontinst* est un programme de création de fontes virtuelles spécialement conçu pour les fontes PS.

✞ Ses points forts sont (utilisateur λ) :

- commandes de haut niveau permettant d'automatiser l'installation d'une famille « standard » (expert ou non), pourvu qu'elle se conforme aux noms de K. BERRY,
- capacité de simuler des glyphes absents (glyphes diacrités, mais aussi symboles mathématiques). Très largement configurable.

✝ Ses points forts sont (utilisateur exigeant) :

- gère les codages par noms de glyphes (comme les fontes PS),
- utilise à fond les macros de TEX, donc très flexible, en particulier
- définition de glyphes conditionnelles,
- peut utiliser de nombreuses fontes pour fabriquer un glyphe : on peut aisément créer à partir d'une famille des fontes répondant à des besoins variés.

✠ Deux points faibles :

- « *(I assume fontinst users are TeXperts)* » (commentaire qui se trouve dans les sources...),
- marche à peu près 100 fois moins vite que *afm2tfm*.

## *Installation standard*

Pour installer une famille de fontes SM comme `put`, il suffit de créer le fichier suivant, et de le compiler avec TEX ou LATEX

```
\input fontinst.sty
\latinfamily{put}{}
\bye
```

Pour installer une famille de fontes SM expert comme `putx`, on fait comme ça :

```
\input fontinst.sty
\latinfamily{putx}{}
\bye
```

*fontinst* génère les PL et VPL, plus les FD : il ne reste plus qu'à mouliner les TFM et VF, et à trouver une fonte mathématique convenable.

Pour une fonte MM, ou pour toute adaptation plus fine, il faut redescendre aux commandes de plus bas niveau, par exemple pour prendre en compte le corps. Voici les commandes qui permettent de fabriquer les fontes virtuelles utilisées ici :

```
\input fontinst.sty
\needsfontinstversion{1.8}
\setint{capspacing}{0}
\setint{smallcapspacing}{0}
\setint{minimumkern}{5}
\installfonts
\installfamily{T1} {pmnx}{}
\installfamily{T1} {pmnj}{}
\installfamily{TS1}{pmnx}{}
\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>}
```

La commande de base de *fontinst* est

`\installfont{`*nom du VF*`}{`*liste des AFM/MTX source*`}{`codage`}...`

(le reste des arguments paramètre le FD)

`\transformfont` permet d'obtenir tous les effets spéciaux autorisés en PS

– incliner les caractères (`\slantfont`);
– recoder la fonte (`\reencodefont`);
– rééchelonner (`\scalefont`, `\xscalefont`, `\yscalefont`).

## *Contrôle typographique fin*

On veut créer une fonte permettant d'obtenir *Pastes* en composant `\swashit Pastes`. Pour cela, il faut

– modifier le *codage* pour y introduire les glyphes de **pmnMMw** et définir les ligatures automatiques $s+t \mapsto \textit{st}$ ;

– introduire quelques crénages entre les initiales ornées et les bas de casses si c'est nécessaire (cf. *Pa* vs *Pa*), créner *st* à gauche comme *s* et à droite comme *t* : ceci se fait dans un fichier MTX de *métriques*.

Un codage est un fichier ETX qui contient des *slots*, voici la définition de *f* en T1

```
\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
   \comment{The letter ‘{f}’.}
\endsetslot
```

on peut s’en inspirer pour :

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

**latin.mtx** est un gros fichier qui gère tout ce qui est automatisable dans la configuration des métriques créées par *fontinst*. On peut rajouter ces lignes :

```
\setleftkerning{st}{s}{1000}
\setleftkerning{ct}{c}{1000}
\setrightkerning{st}{t}{1000}
\setrightkerning{ct}{t}{1000}
```

On installera alors notre fonte par

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

# *Fontes mathématiques*

- « *Don’t mix faces haphazardly when specialized sorts are required* » Robert BRINGHURST.
- En d’autres termes : il existe très peu de fontes mathématiques. Il n’y a rien de tel pour casser une typographie que de mélanger au petit bonheur une fonte de texte et l’une des rares fontes mathématiques.
- Dans l’exemple que je décris, j’ai *décidé* d’utiliser CM pour compléter Minion, j’ai utilisé la technologie MM pour me rapprocher au maximum de CM.
- J’aurais aussi bien pu prendre un Minion plus condensé et plus gras, pour le compléter avec MathTimes.
- Dans les deux cas, il y a incompatibilité de style.
- C’est tout de même moins pire que d’utiliser Minion avec des italiques mathématiques venant d’un autre monde.

# *Création de fontes mathématiques*

```
\installfamily{OML}{pmn}{\skewchar\font=127}

\installfont{zpmnlcmm}
 {kernoff,cmmi10,kernon,unsetalmf,unsetos,%
 pmnli8r10,pmnli8x10,bmathit,pmnli8a10-sbhax}
  {OML}{OML}{pmn}{m}{it}{<9.5-10.5>}
\installfont{zpmndcmm}
 {kernoff,cmmib10,kernon,unsetalmf,unsetos,%
 pmndi8r10,pmndi8x10,bmathit,pmndi8a10-sbhax}
  {OML}{OML}{pmn}{m}{it}{<9.5-10.5>}

\installfamily{OT1}{pmn}{}

\installfont{zpmnlcm7t}
 {pmnl8r10,pmnl8x10,cmr10,blatin,zrhax,%
  kernoff,cmr10}
        {OT1}{OT1}{pmn}{m}{n}{<9.5-10.5>}
\installfont{zpmndcm7t}
 {pmnd8r10,pmnd8x10,cmbx10,blatin,zrhax,%
  kernoff,cmbx10}
        {OT1}{OT1}{pmn}{b}{n}{<9.5-10.5>}
```

Italiques mathématiques de MathPTM (virtuelle, basée sur Times et Symbol, codage OML) :

$\Gamma\ \Delta\ \Theta\ \Lambda\ \Xi\ \Pi\ \Sigma\ \Upsilon\ \Phi\ \Psi\ \Omega\ \alpha\ \beta\ \gamma\ \delta\ \varepsilon$

$\zeta\ \eta\ \theta\ \iota\ \kappa\ \lambda\ \mu\ \nu\ \xi\ \pi\ \rho\ \sigma\ \tau\ \upsilon\ \phi\ \chi$

$\psi\ \omega\ \varepsilon\ \vartheta\ \varpi\ \varrho\ \varsigma\ \varphi\ \leftharpoonup\ \rightharpoondown\ \flat\ \natural\ \sharp\ \smile\ \frown$

$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ .\ ,\ <\ /\ >\ \star$

$\partial\ A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O$

$P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z\ \flat\ \natural\ \sharp\ \smile\ \frown$

$\ell\ a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o$

$p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z\ \imath\ \blacksquare\ \wp\ \vec{}\ \frown$

## *Fichiers FD*

- *Fontinst* est assez malin pour prendre en charge l'interface obscur de NFSS : les fichiers FD.
- Les cinq derniers arguments de `\installfont` déterminent l'entrée correspondant à la fonte créée.
- C'est la commande `\installfamily` qui provoque la création de ces fichiers.
- Par exemple, les commandes qu'on vient de voir ont créé deux FD.

☛ omlpmn.fd :

```
%Filename: omlpmn.fd
%Created by: tex build_math_pmn
%Created using fontinst v1.600

\ProvidesFile{omlpmn.fd}
   [1997/12/31 Fontinst v1.600
  font definitions for OML/pmn.]

\DeclareFontFamily{OML}{pmn}
         {\skewchar \font =127}

\DeclareFontShape{OML}{pmn}{b}{it}{
   <-6.5> zpmndcmm6
   <6.5-7.5> zpmndcmm7
   <7.5-8.5> zpmndcmm8
   <8.5-9.5> zpmndcmm9
   <9.5-10.5> zpmndcmm10
   <10.5-11.5> zpmndcmm11
   <11.5-15> zpmndcmm12
   <15-36> zpmndcmm24
   <36-> zpmndcmm72
}{}
```

☛ ot1pmn.fd :

```
%Filename: ot1pmn-cm.fd
%Created by: tex build_math_pmn
%Created using fontinst v1.600

\ProvidesFile{ot1pmn.fd}
   [1997/12/31 Fontinst v1.600
   font definitions for OT1/pmn.]

\DeclareFontFamily{OT1}{pmn}{}

\DeclareFontShape{OT1}{pmn}{m}{n}{
   <-6.5> zpmnlcm7t6
   <6.5-7.5> zpmnlcm7t7
   <7.5-8.5> zpmnlcm7t8
   <8.5-9.5> zpmnlcm7t9
   <9.5-10.5> zpmnlcm7t10
   <10.5-11.5> zpmnlcm7t11
   <11.5-15> zpmnlcm7t12
   <15-36> zpmnlcm7t24
   <36-> zpmnlcm7t72
}{}
```

# *Configurer LaTeX*

Trois « couches » de LaTeX interagissent avec les fontes :

- NFSS de bas niveaux, à travers les fichiers FD ;
- 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...).

Je propose une extension dont le but est double :

– remplacer toutes les fontes utilisées dans le document par un système alternatif et cohérent (ici : Minion/CM). 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.

Dans le cas d'une famille suffisamment standard, ceci change tous les appels à la famille par défaut en mode texte :

```
\renewcommand{\rmdefault}{pmnx}
```

```
%% This is file ‘minion.sty’, not generated
%%  with the docstrip utility
\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}%
 \DeclareSymbolFont{operators}    {OT1}{pmn}{m}{n}%
 \SetSymbolFont{operators}{normal}{OT1}{pmn}{m}{n}%
 \SetSymbolFont{operators}{bold}  {OT1}{pmn}{b}{n}%
 \DeclareMathAlphabet{\mathcal}   {T1} {bi0}{m}{n}%
 \SetMathAlphabet{\mathcal}{normal}{T1}{bi0}{m}{n}%
  \let\vec@@ori\vec
  \renewcommand\vec[1]{\skew0\vec@@ori{#1}}%
}
\ExecuteOptions{lining}
\ProcessOptions
\renewcommand{\rmdefault}{pmn\TheSpecialBit}
\renewcommand{\sfdefault}{pmy}
\def\bfdefault{b}
%% fin des options globales
%% macros spéciales
\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’.
```

## *Configurer les pilotes*

- Si j'utilise la fonte virtuelle **pmnl9d10**, LaTeX et TeX n'ont vu que le TFM.
- J'obtiens un DVI qui fait référence à ce TFM (mais ne contient pas le dessin des caractères).
- Un pilote va convertir ce DVI en un format utilisé dans le monde réel : *bitmap* X11, PS ou PCL.
- Le pilote doit d'abord voir qu'il s'agit d'une fonte virtuelle, et lire le VF, qui le redirige vers des fontes « réelles » (**pmnl8r10.tfm**, **pmnl8x10.tfm**).
- À ces fontes correspondent les glyphes définis en PS par les fichiers **pmnl8r10.pss**, **pmnl8x10.pss**, qui reposent sur **pmnMM8a.pfb**, et **pmnMM8x.pfb**.

- De deux choses l'unes :
  - c'est un pilote PS : il doit savoir quelles fontes charger, et où les trouver,
  - sinon il faut convertir ces fontes au format utilisé en sortie.
- Il existe des utilitaires qui convertissent à la volée des fontes PS en *bitmap* d'écran, certaines visioneuses savent en tirer partie, d'autres non (*dviwindo/ATM* vs. *xdvi/Xfs*).
- Comme la plupart des pilotes TEX gèrent le format PK des *bitmaps Metafont*, on peut utiliser les fontes PS en les convertissant à ce format (*ps2pk* ou *gsftopk*).

# *Fontes virtuelles*

Une fonte de texte normale comme putr8t est une fonte virtuelle. C'est-à-dire qu'il existe un fichier putr8t.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 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, puis à traiter le DVI résultant. Cela signifie que le pilote a besoin des fichiers putr8t.tfm et putr8t.vf dans un premier temps, puis des fichiers TFM correspondant aux fontes (« réelles ») utilisées par la fonte virtuelle (ici putr8r.tfm uniquement). À partir de ce moment, la fonte virtuelle est inutile : le pilote ne doit pas chercher à imprimer la fonte putr8t qui n'existe pas. Il cherchera les fichiers lui permettant de dessiner les caractères utilisés : dans un cas simple comme Utopia, il s'agira de la description PS-type 1 du contour de ces caractères par le fichier

putr8a.pfb. 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 put8r.tfm (DVI) mais le contour est défini par putr8a.pfb au niveau PS : cela signifie que le codage est différent. Le pilote convertissant DVI vers PS devra donc recoder putr8a.pfb avant de l'utiliser.

Voici ce qu'il faut ajouter au fichier `psfonts.map` de *dvips* pour qu'il reconnaisse putr8r :

```
putr8r Utopia-Regular <putr.pfb \
 "TeXBase1Encoding ReEncodeFont" <8r.enc
```

Et voici un cas un peu plus difficile : **pmnl9d10** (*Minion* MM)

```
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
```

*gsftopk* utilise le même fichier de configuration, ce qui est bien agréable.

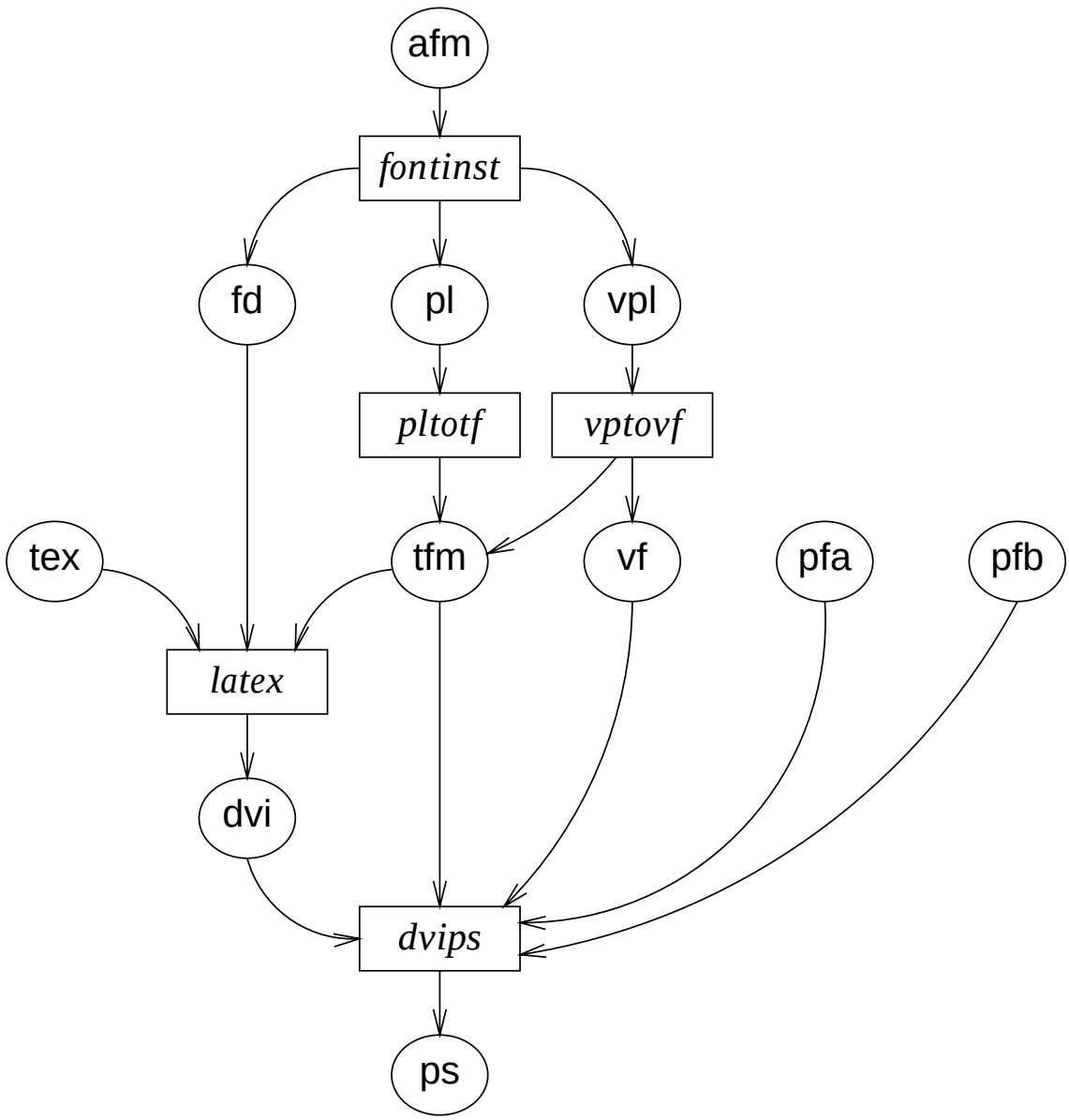
# *TDS*

On vient de créer 1 354 fichiers pour un total de 15 méga octets... Ça mérite un peu de rangement ! Sinon pas un *dvips* n'y retrouverait ses VF !

```
$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
```


afm
fontinst
fd
pl
vpl
pltotf
vptovf
tex
tfm
vf
pfa
pfb
latex
dvi
dvips
ps