Formats de fichiers autour de TEX et outils de visualisation.

Denis ROEGEL

roegel@loria.fr

Journées GUTenberg

Mai 1997

Autour de TEX interviennent beaucoup de fichiers : `.tex`, `.log`, `.mf`, `.dvi`, `.fmt`, `.tfm`, `.300pk`, etc.

- Fichiers texte : `.tex`, `.log`, `.mf`, etc.
- Fichiers binaires : `.dvi`, `.fmt`, `.tfm`, `.300pk`, etc.

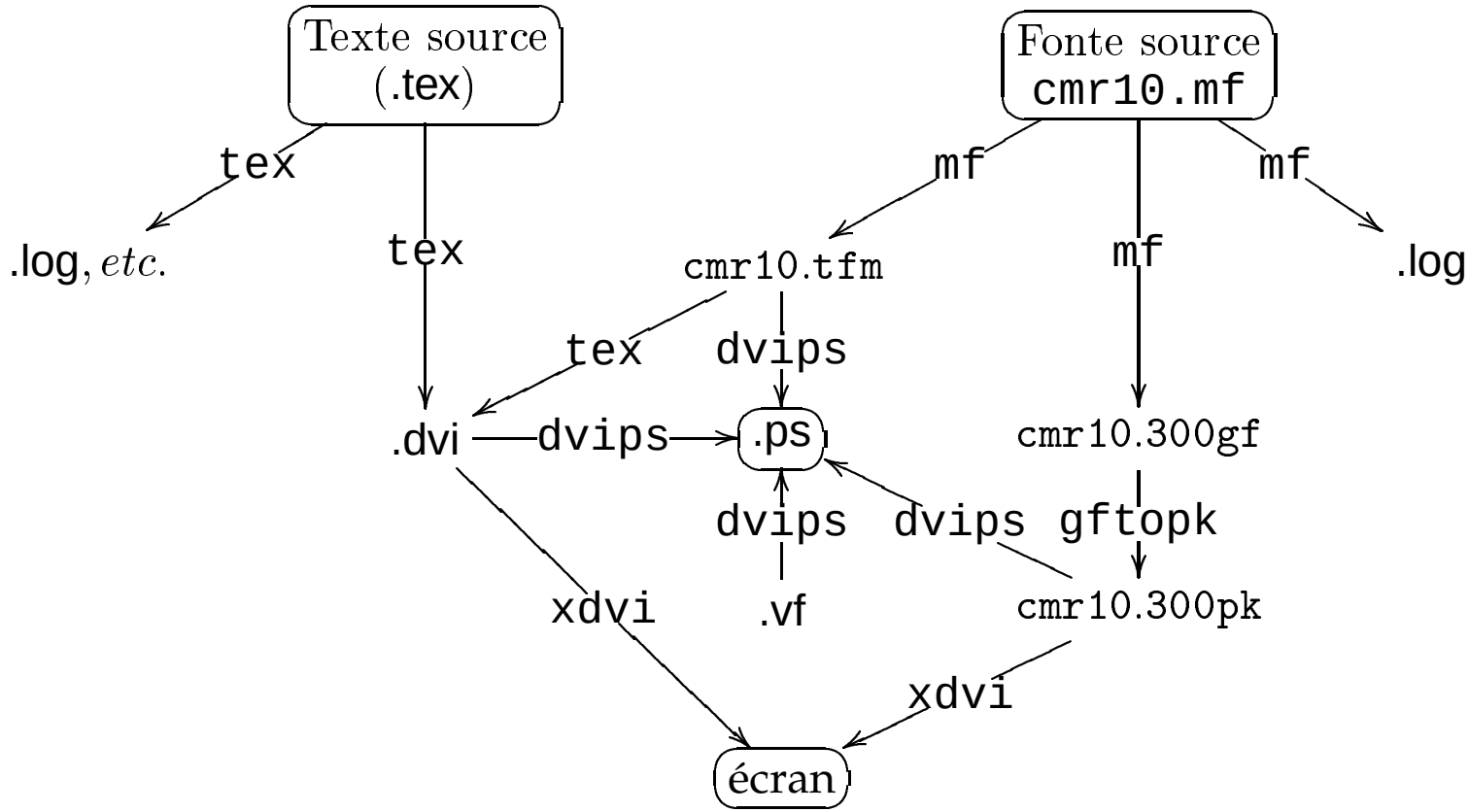
$\Rightarrow$ les formats binaires sont plus mystérieux...

1. Formats et programmes de visualisation

- `DVI` : description des pages (lisible par `dvitype`)
- `TFM` : métriques des polices (lisibles par `tftopl`)
- `VF` : polices virtuelles (lisibles par `vftovp`)
- `GF` : *bitmaps* METAFONT (lisibles par `gftype`)
- `PK` : *bitmaps* compactés (lisibles par `pktype`)

À notre connaissance, il n'y a pas de programme (à part TEX) pour visualiser le contenu d'un fichier `FMT`.

2. Organisation des fichiers



3. Le format DVI

– défini par David R. Fuchs en 1979 ;

– sorte de langage machine pour décrire ce qui se trouve sur une page ;

– dimensions exprimées en unités DVI, égales à $1/65536$ pt si le fichier est produit par TEX ;

– référence : point supérieur gauche de la page décalé d'un pouce vers la droite et le bas ;

– contenu du fichier lisible par `dvitype`.

3.1. Préambule de fichier DVI

```
Options selected:
  Starting page = *
  Maximum number of pages = 1000000
  Output level = 4 (the works)
  Resolution = 300.00000000 pixels per inch
numerator/denominator=25400000/473628672
magnification=1000;       0.00006334 pixels per DVI unit
' TeX output 1997.04.02:2101'
Postamble starts at byte 8410.
maxv=42908243, maxh=30207812, maxstackdepth=6, totalpages=4
```

$$f = \frac{1}{65536} \cdot \frac{2{,}54}{72{,}27} \cdot \frac{1}{100} \cdot 10^7 = \frac{25400000}{473628672}$$

3.2. Préambule de fichier DVI (suite)

```
Font 40: logosl10---loaded at size 655360 DVI units
Font 39: cmtt10 scaled 1095---loaded at size 717619 DVI units
 (this font is magnified 109%)
...
Font 7: cmr10---loaded at size 655360 DVI units
Font 6: cmr7---loaded at size 458752 DVI units
```

3.3. Positions sur la page

- h : coordonnée horizontale ;
- v : coordonnée verticale ;
- w : espacement horizontal ;
- x : espacement horizontal ;
- y : espacement vertical ;
- z : espacement vertical.

w, x, y, et z permettent de mémoriser des espacements qui se reproduisent (inter-mots, crénages, inter-lignes).

- hh : valeur de h en pixels ;
- vv : valeur de v en pixels.

3.4. Description de page DVI

```
42: beginning of page 1
87: down4 42908243 v:=0+42908243=42908243, vv:=2718
92: push
level 0:(h=0,v=42908243,w=0,x=0,y=0,z=0,hh=0,vv=2718)
93: down4 -36008935 v:=42908243-36008935=6899308, vv:=437
98: push
level 1:(h=0,v=6899308,w=0,x=0,y=0,z=0,hh=0,vv=437)
99: down3 -783519 v:=6899308-783519=6115789, vv:=387
103: push
level 2:(h=0,v=6115789,w=0,x=0,y=0,z=0,hh=0,vv=387)
104: right3 -372935 h:=0-372935=-372935, hh:=-24
[ ]
108: fntdef1 22: cmr8
128: fntnum22 current font is cmr8
129: setchar66 h:=-372935+394314=21379, hh:=1
130: setchar114 h:=21379+217091=238470, hh:=15
...
138: w3 185688 h:=1878714+185688=2064402, hh:=131
142: setchar100 h:=2064402+309480=2373882, hh:=151
...
```

3.5. Description de page DVI (suite)

```
150: setchar101 h:=3797945+247584=4045529, hh:=258
151: w0 185688 h:=4045529+185688=4231217, hh:=268
152: setchar112 h:=4231217+309480=4540697, hh:=288
153: right2 15474 h:=4540697+15474=4556171, hh:=289
156: setchar111 h:=4556171+278532=4834703, hh:=307
...
163: w0 185688 h:=6169017+185688=6354705, hh:=403
[Brouillon d'article pour les ]
164: fntdef1 32: cmti8
185: fntnum32 current font is cmti8
186: setchar67 h:=6354705+402683=6757388, hh:=429
...
3413: pop
level 2:(h=-372935,v=41043563,w=0,x=0,y=622592,z=737281,hh=-24,vv=2600)
3414: pop
level 1:(h=0,v=41043564,w=0,x=0,y=0,z=0,hh=0,vv=2600)
3415: pop
level 0:(h=0,v=42908243,w=0,x=0,y=0,z=0,hh=0,vv=2718)
3416: eop
```

3.6. Commandes « special » dans un fichier DVI

Ces commandes permettent d'inclure des commandes qui seront interprétées par le pilote (`xdvi`, `dvips`, ...).

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

(produit par `\includegraphics{org.ps}`)

4. Le format TFM

– Format utilisé par TEX pour obtenir les dimensions des caractères des polices ;

– TEX n'utilise aucun autre format de fichier pour les polices ;

– Fichiers produits par METAFONT ou d'autres programmes comme `afm2tfm`, etc. ;

– Contenu du fichier lisible par `tftopl`, transformant le format TFM en PL (*Property List*).

4.1. Préambule de fichier PL (logo10)

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

4.2. Préambule de fichier PL (ecrm1000)

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

4.3. Nouveaux paramètres des polices EC

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

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

fontdimen 10 est la hauteur des lettres capitales accentuées ;

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

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

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

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

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

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

4.4. Préambule de fichier PL (cmsy10)

```
(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)
...
   (AXISHEIGHT R 0.25)
   )
```

4.5. Préambule de fichier PL (cmex10)

```
(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)
   )
```

4.6. Ligatures et crénages

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

4.7. Caractères

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

4.8. Liens entre caractères

Codage de suites de caractères croissant en taille.

Exemple : (, (, (, (, (.

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

4.9. Caractères décomposés

Le caractère octal 62 est un élément d'un symbole extensible dont le *haut* (TOP) est le symbole octal 62 (⎾), le *bas* (BOT) est le symbole octal 64 (⎿) et la partie répétée (REP) est le symbole octal 66 (⏐).

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

5. Le format VF

– Format introduit par Knuth en 1990, et inspiré d'un système analogue réalisé par David R. Fuchs en 1984 ;

– Abstraction de police permettant de créer une « nouvelle » police à partir d'une ou plusieurs polices déjà existantes ;

– Utilisation typique : recodage ;

– Version lisible par `vftovp` produisant une sortie VPL (*Virtual Property List*) ;

– Les fichiers VPL peuvent être créés par le programme `fontinst` ;

– Programme `vptovf` permettant de passer d'un fichier VPL à un fichier VF.

5.1. Début de fichier VPL

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

5.2. Caractères dans un fichier VPL

```
(CHARACTER O 0
   (CHARWD R 0.5)
   (CHARHT R 0.502997)
   (MAP
      (SETRULE R 0.5 R 0.5)
      (SPECIAL Warning: missing glyph ‘Gamma’)
      )
   )
...
(CHARACTER C A
   (CHARWD R 0.721997)
   (CHARHT R 0.667999)
   (COMMENT
      (KRN C y R -0.091992)
      (KRN C w R -0.091992)
      (KRN C v R -0.073999)
      (KRN O 47 R -0.110999)
      (KRN C Y R -0.104993)
      (KRN C W R -0.08999)
      (KRN C V R -0.134998)
...
      )
   (MAP
      (SETCHAR C A)
      )
   )
...
```

5.3. Déplacements et empilements dans un fichier VPL

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

6. Le format GF

– Format *bitmap* produit par METAFONT ;

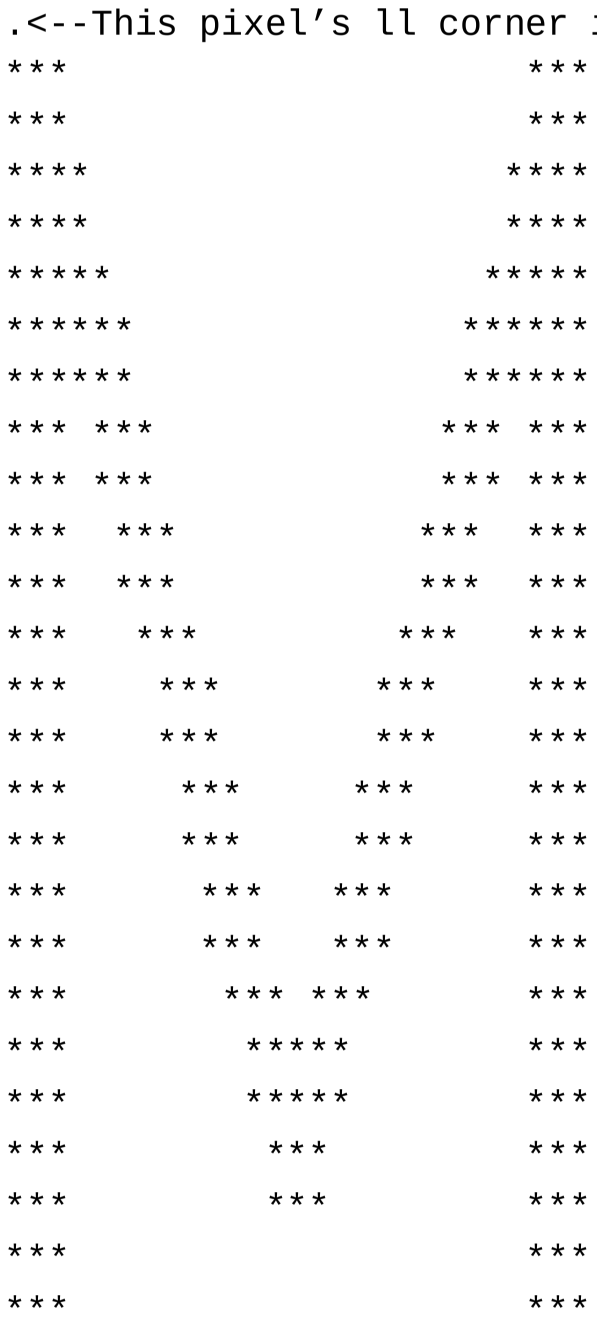
– Lisible par le programme `gftype`.

6.1. Commandes d'un caractère dans un fichier GF

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

35: beginning of char 77: 3<=m<=30 0<=n<=24
(initially n=24) paint (0)3(21)3
45: newrow 0 (n=23) paint 3(21)3
49: newrow 0 (n=22) paint 4(19)4
53: newrow 0 (n=21) paint 4(19)4
57: newrow 0 (n=20) paint 5(17)5
61: newrow 0 (n=19) paint 6(15)6
65: newrow 0 (n=18) paint 6(15)6
69: newrow 0 (n=17) paint 3(1)3(13)3(1)3
77: newrow 0 (n=16) paint 3(1)3(13)3(1)3
...
183: newrow 0 (n=2) paint 3(9)3(9)3
189: newrow 0 (n=1) paint 3(21)3
193: newrow 0 (n=0) paint 3(21)3
197: eoc
```

6.2. *Bitmap* d’un caractère dans un fichier GF



6.3. Commandes « special » d'un fichier GF

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

6.4. Postambule d'un fichier GF

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

- Taille du corps exprimée en multiples de $\frac{1}{2^{20}}$ pt, ici $2^{20} \cdot 10$.
- Nombre de pixels par point (en *scaled integers*) en 300dpi : $300/72.27 \cdot 2^{16} \approx 272046$
- TFM width du caractère 77 (M) (en fraction du corps et multiplié par 2^{20}) : $0.799997 \cdot 2^{20} \approx 838858$
- dx : $2162688 = 33 \cdot 2^{16}$, donc 33 pixels.

$$\frac{\text{dx}}{\text{TFM}} \approx \frac{10pt \cdot 300}{72.27 \cdot 16}$$

7. Le format PK

- Format introduit par Tomas Rokicki.
- Produit à partir du format GF avec `gftopk`, ou de `.gsf` avec `gsftopk`, etc.
- Représentation compacte (*packed*) des données d'un fichier GF ;
- Plus facile à transformer en *bitmap*, car peu d'instructions à interpréter ;
- Les *bounding box* minimales sont données explicitement et ne doivent pas être calculées comme dans le format GF ;
- Le nombre d'octets pris par chaque caractère est indiqué, ce qui permet de charger les données rapidement, sans interpréter chaque commande ; si un caractère n'est pas utilisé, le pilote peut le sauter rapidement.

7.1. Deux sortes d'octets

Les octets sont divisés en deux catégories :

- 0 à 239 : valeurs introduisant une définition de caractère ;
- 240 à 255 : valeurs n'introduisant pas une définition de caractère ; ce sont des commandes pouvant avoir zéro ou plus paramètres. Ces valeurs ne peuvent pas apparaître autrement que sous forme de commandes.

7.2. Structure d'un fichier PK

– préambule (commençant par la *commande* préambule) ;

– définition d'un ou plusieurs caractère(s) ;

– postambule (finissant par la *commande* postambule).

7.3. Commandes

– *pk_post* (code 245) : début du postambule. Cette commande est suivie de suffisamment de *pk_no_op* pour que le fichier ait une longueur multiple de 4.

– *pk_no_op* (code 246) : pas d'opération.

– *pk_pre* (code 247) : suivi du code 89 identifiant le format PK, d'un commentaire identifiant en général l'origine du fichier, de la taille du corps *ds* (en $1/2^{20}$ points), de la somme de contrôle *cs* et du nombre de pixels par points, horizontalement et verticalement, mais multiplié par 2^{16}.

7.4. Compactage

Deux buts antagonistes du format PK :

– compacter des *bitmaps* ;

– conserver la facilité de traduction en *bitmaps*.

Principe :

– concaténation des bits de toutes les lignes et codage des nombres de pixels noirs ou blancs consécutifs ;

– utilisation de la largeur du caractère pour le décodage (pas besoin de commandes spéciales pour le changement de ligne) ;

– utilisation de la *bounding box* minimale pour trouver la fin d'un caractère.

7.5. Distribution des segments de pixels

– Expérimentalement, en observant beaucoup de *bitmaps* à 300dpi, il a été constaté un maximum à 4 pixels. La fréquence baisse lentement jusqu'à 10 pixels, puis plus rapidement jusqu'à 20 pixels, s'approchant ensuite assymptotiquement de 0.

  $\Rightarrow$ La majorité des segments peuvent être codés sur 4 bits. Le codage des *bitmaps* dans un fichier PK est basé sur des quartets.

– 37% des lignes sont identiques à la ligne précédente ;

  $\Rightarrow$ le format PK permet de spécifier combien de fois une ligne horizontale est répétée.

– Aucun segment n'est nul car cela correspondrait à un caractère n'ayant aucun pixel, ce qui peut être détecté avec les dimensions de la *bounding box*.

En résumé :

– longueurs de segments (*run counts*)

– nombre de répétitions (*repeat counts*)

Note : les répétitions de lignes entièrement blanches ou entièrement noires sont codées sous forme de longueurs de segments plutôt que de nombres de répétitions.

7.6. Codage des quartets

- 16 valeurs possibles : 0 à 15 ;
- les longueurs de segments sont plus fréquentes que les nombres de répétitions ;
-
  - valeurs 0 à 13 pour coder les longueurs de segments ;
  - valeur 14 suivi d'un *nombre compacté* pour représenter un nombre de répétitions ;
  - valeur 15 pour représenter une seule répétition.

7.7. Codage des nombres compactés

- On veut coder les entiers positifs jusqu'à $2^{31} - 1$;
- on souhaite que les nombres petits et fréquents occupent un ou deux quartets et les nombres grands et rares trois quartets ou plus.

$\Rightarrow$ la valeur de quartet 0 signifie que le nombre est grand et occupe trois quartets ou plus.

$\Rightarrow$ les valeurs 1 à 13 introduisent des nombres codés sur un ou deux quartets.

7.8. Nombres sur un ou deux quartets

Soit $0 \leq dyn\_f \leq 13$.

- les valeurs de 1 à dyn_f peuvent être utilisées pour coder directement les nombres de 1 à dyn_f sur un quartet ;
- les valeurs de $dyn\_f + 1$ à 13, suivies de quartets de 0 à 15 peuvent être utilisées pour coder les nombres de $dyn\_f + 1$ à $dyn\_f + (13 - dyn\_f) \cdot 16$.

$\Rightarrow$ Un grand nombre est donc un nombre supérieur à $dyn\_f + (13 - dyn\_f) \cdot 16$.

$\Rightarrow$ dyn_f est déterminé pour chaque caractère, de telle sorte qu'il soit codé sur le plus petit nombre de quartets (`Dynamic packing variable` dans la sortie de `pktype`).

7.9. Nombres sur trois quartets ou plus

Principe de base de codage de l'entier i :

- écrire i en hexadécimal en supprimant les zéros au début ;
- compter le nombre n de chiffres hexadécimaux ;
- ajouter $n - 1$ zéros au début du nombre ;

$\Rightarrow$

- les valeurs 0 à 15 occupent un quartet ;
- les valeurs 16 à 255 occupent trois quartets ;
- etc.

En pratique :

- comme on n'a besoin de coder que les valeurs supérieures à $dyn\_f + (13 - dyn\_f) \cdot 16$,
- comme les quartets seuls sont déjà pris,

on applique le schéma précédent à $n - (dyn\_f + (13 - dyn\_f) \cdot 16) + 16$ pour coder le grand nombre n.

7.10. Descripteur de caractère

Chaque caractère commence par des informations générales :

- *flag* : le quartet le plus significatif donne dyn_f ; le bit de poids 8 indique si le premier pixel est noir (si 1) ou blanc (si 0) ;
- longueur de paquet : nombre d'octets depuis le premier indiquant la largeur TFM jusqu'à la fin de la description du caractère ; ceci permet de sauter tout le caractère ;
- code du caractère
- largeur TFM : largeur telle que donnée dans le fichier TFM ou GF ;
- échappement horizontal (pixels) : largeur du caractère en incluant l'espace avant (approche) et après ;
- largeur : largeur de la *bounding box* du caractère ;
- hauteur : hauteur de la *bounding box* du caractère ;
- offset horizontal : offset horizontal (positif vers la droite) entre le pixel supérieur gauche du caractère et le pixel de référence ;
- offset vertical : idem (positif vers le bas).

7.11. Préambule de sortie de `pktype`

Valeurs analogues à celles trouvées dans un fichier GF.

```
This is PKtype, Version 2.3 (C version 6.1)
’METAFONT output 1994.06.12:1754’
Design size = 10485760
Checksum = -124489922
Resolution: horizontal = 272046  vertical = 272046  (300 dpi)
```

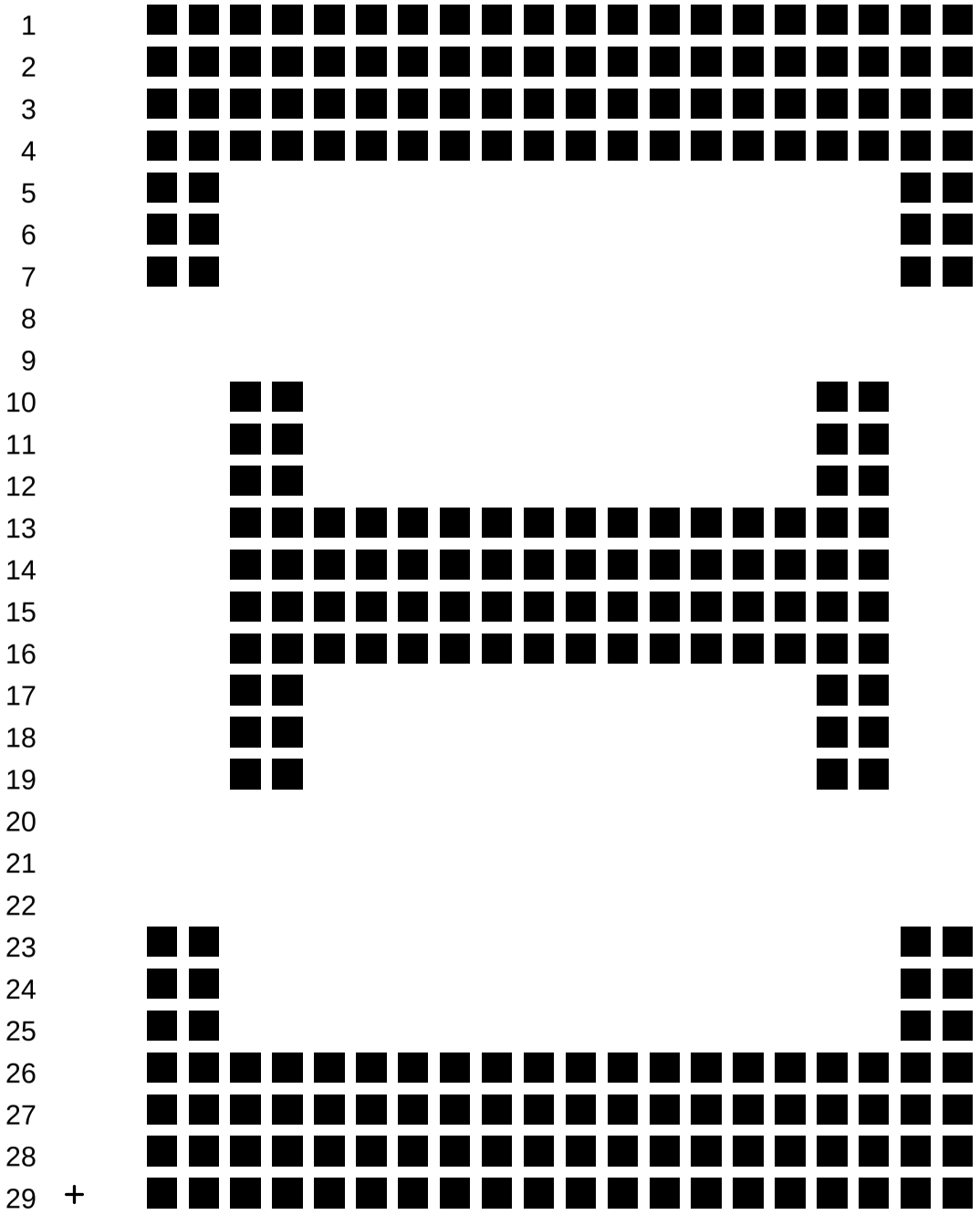
7.12. Caractères dans une sortie de `pktype`

```
50:  Flag byte = 200  Character = 77  Packet length = 50
  Dynamic packing variable = 12
  TFM width = 838858  dx = 2162688
  Height = 25  Width = 27  X-offset = -3  Y-offset = 24
  [1]3(21)7[1](19)9(17)11[1](15)9[1](1)3(13)3(1)6[1](2)3(11)3(2)6(3)3(9)3(3)6
  [1](4)3(7)3(4)6[1](5)3(5)3(5)6[1](6)3(3)3(6)6(7)3(1)3(7)6[1](8)5(8)6[1](9)3
  (9)6[1](21)3
100:  Flag byte = 176  Character = 69  Packet length = 22
  Dynamic packing variable = 11
  TFM width = 652445  dx = 1703936
  Height = 25  Width = 20  X-offset = -3  Y-offset = 24
  (1)42[10](17)[1]18(2)[7]3(17)20(1)19
...
```

7.13. Postambule de sortie de `pktype`

```
330:  Special: ’fontid=MFLOGO’
345:  Special: ’codingscheme=AEFMNOPST only’
374:  Special: ’fontfacebyte’
388:  Num special: 15335424
393:  Special: ’jobname=logo10’
409:  Special: ’mag=1’
416:  Special: ’mode=CanonCX’
430:  Special: ’pixels_per_inch=300’
451:  Special: ’blacker=0’
462:  Special: ’fillin=0.2’
474:  Special: ’o_correction=0.6’
492:  Postamble
493:  No op
494:  No op
495:  No op
496 bytes read from packed file.
```

7.14. Exemple : caractère Ξ



– largeur de la plus petite *bounding box* : 20

– hauteur : 29

– $+$: pixel de référence (en-dehors de la *bounding box*)

– $hoff = -2$

– $voff = 28$

1. Calcul des longueurs de segments et répétitions :

   on supprime les lignes dupliquées qui ne sont pas toutes blanches ou toutes noires et on indique les duplications au premier changement de couleur de la ligne.

   ```
   82 [2] (16) 2 (42) [2] 2 (12) 2 (4) [3]
   16 (4) [2] 2 (12) 2 (62) [2] 2 (16) 82
   ```

   – 82 : 82 pixels noirs

   – [2] : ligne *courante* dupliquée deux fois

   – (16) : 16 pixels blancs

   – 2 : 2 pixels noirs

   – (42) : 42 pixels blancs

   – [2] : ligne *courante* dupliquée deux fois

   – etc.

2. Calcul de *dyn_f* : ici 8.

3. Codage en quartets :

   ```
   D9 E2 97 2 B1 E2 2 93 2 4 E3
   97 4 E2 2 93 2 C5 E2 2 97 D9
   ```

   – $n \to xy\ ((x-1)-8)\cdot 16+8+y+1$

   – 82 $\to$ D9 $((13-1)-8)\cdot 16+8+9+1$

   – [2] $\to$ E2

   – (16) $\to$ 97 $((9-1)-8)\cdot 16+8+7+1$

   – 2 $\to$ 2

   – (42) $\to$ B1 $((11-1)-8)\cdot 16+8+1+1$

   – etc.

4. Paquet final :

Flag byte	88		
Packet length	1A		
Character code	04		
tfm width	09	C7	1C
Horizontal escapement (pixels)	19		
Width of bit map	14		
Height of bit map	1D		
Horizontal offset (signed)	FE		
Vertical offset	1C		
Raster data	D9	E2	97
	2B	1E	22
	93	24	E3
	97	4E	22
	93	2C	5E
	22	97	D9