Les fontes sous LaTeX pour les nuls (et les autres)

Vincent Zoonekynd

1998 – juillet 2000

Résumé

Ce document explique comment utiliser LaTeX avec d'autres fontes que les sempiternelles Computer Modern. Le début s'efforce d'être compréhensible par tous, mais les manipulations plus compliquées exposées par la suite sont vivement déconseillées aux utilisateurs non avertis de LaTeX.

Je tiens à préciser que je ne comprends pas toujours ce dont je parle : ce document contient probablement des erreurs ou de mauvais conseils. Ne l'utilisez qu'avec un oeil critique et n'hésitez pas à me signaler ces problèmes.

Comme il s'agit en fait de la liste de mes divers essais d'utilisation de fontes avec LaTeX, ce document risque de paraître un peu brouillon.

De plus, ce document reste incomplet et je ne sais pas si j'aurai le temps de le compléter.

Table des matières

1 Introduction **6**
1.1 Où trouver la dernière version de ce document ? 6
1.2 Quels sont les autres documents sur le sujet ? 6
1.3 Où trouver des fontes Métafont, PostScript ou TTF ? 7

2 Choses simples **11**
2.1 Quelles sont les commandes de changement de fontes ? 11
2.2 Comment écrire plus grand ? 12
2.3 Comment changer de fontes pour tout le document ? 14
2.4 Où trouver les caractères textuels manquants ? 16
2.5 Comment avoir un symbole Euro ? 20
2.6 Comment avoir des « chiffres elzéviriens » ? 22
2.7 Où trouver les caractères mathématiques manquants ? 22
2.8 Comment avoir une fonte mathématique manuscrite ? 24
2.9 Comment avoir une fonte mathématique gothique ? 25
2.10 Comment avoir des lettres mathématiques avec une double barre ? 26
2.11 Comment avoir des symboles mathématiques en gras ? 27

3 Généralités **29**
3.1 Vocabulaire 29
3.2 Quels sont tous ces fichiers ? 31
3.3 Comment passer de l'un de ces format de fichier à un autre ? 47
3.4 Où LaTeX va-t-il chercher tous ces fichiers ? 48
3.5 Quelle est la différence entre `inputenc` et `fontenc` ? 49
3.6 Comment taper les lettres accentuées « normalement » ? 49
3.7 Comment taper le oe normalement ? 49
3.8 Qu'est-ce que le codage ? 49
3.9 Quels sont les différents codages d'entrée ? 50
3.10 Quels sont les différents codages de fontes ? 50
3.11 Est-il vrai que le codage `OT1` pose des problèmes de césure ? 50
3.12 Pourquoi les fontes ont-elles des noms aussi illisibles ? 51
3.13 Pourquoi y a-t-il plein de chiffres dans certains noms de fontes : `ecrm1000.tfm` ? 52

4 NFSS (New Font Selection Scheme) **53**
4.1 Qu'est-ce que NFSS ? 53
4.2 Quelles sont les caractéristiques d'une fonte ? 53
4.3 Comment changer de fonte (commandes de « bas niveau ») ? 53
4.4 Comment avoir des petites capitales italiques ? 54
4.5 Comment utiliser ponctuellement une fonte ? 55
4.6 Comment utiliser ponctuellement une fonte (bis) ? 56
4.7 Quel est le dernier argument des commandes `\DeclareFontFamily` et `\DeclareFontShape` ? 56
4.8 Quelles autres choses peut-on mettre dans un fichier `*.fd` ? 57
4.9 Comment accéder à un caractère d'une fonte en connaissant juste son numéro ? 59
4.10 Comment changer la fonte par défaut du document ? 59
4.11 Comment utiliser plusieurs codages de fonte en même temps ? 60
4.12 Comment utiliser une fonte texte en mode mathématique ? 61
4.13 Comment utiliser une fonte texte comme si c'était une fonte mathématique ? 61

4.14 Comment utiliser une fonte texte comme fonte mathématique par défaut ? Comment modifier certains caractères mathématiques ? . . . 61
4.15 Comment utiliser une fonte mathématique différente ? . . . 64
4.16 Comment (et pourquoi) utiliser différentes versions mathématiques ? . . . 68
4.17 Quelles sont les différentes dimensions présentes dans une fonte ? Comment les modifier ? . . . 69

5 Au delà des caractères latins **70**
5.1 Comment utiliser l'alphabet phonétique ? . . . 70
5.2 Comment écrire des partitions musicales ? . . . 70
5.3 Comment écrire quelques mots en russe ? Comment utiliser un caractère russe en mode mathématique ? . . . 70
5.4 Comment écrire quelques mots en grec ancien ? . . . 71
5.5 Pourquoi n'y a-t-il pas de béta interne ? . . . 71
5.6 Comment écrire quelques mots en japonais ? . . . 72
5.7 Comment écrire quelques mots en Chinois ? en Coréen ? . . . 73
5.8 Comment inclure quelques mots dans une autre langue ? . . . 74
5.9 Qu'est-ce qu'Unicode ? Comment utilise-t-on Omega ? . . . 74

6 Métafont **76**
6.1 Qu'est-ce que métafont ? . . . 76
6.2 Comment utiliser Métafont ? Comment visualiser des fichiers Métafont ? . . . 76
6.3 Comment réaliser un catalogue de fontes Métafont ? . . . 77
6.4 Comment utiliser une fonte Métafont de manière ponctuelle dans un document LaTeX ? . . . 78
6.5 Comment utiliser une fonte MF comme fonte par défaut dans un document LaTeX ? . . . 79
6.6 Comment utiliser plusieures fontes MF comme fonte par défaut dans un document LaTeX ? . . . 79
6.7 Comment peut-on modifier une fonte Métafont ? . . . 79
6.8 Peut-on transformer une fonte Métafont en une fonte PostScript ? . . . 81
6.9 Peut-on transformer une fonte Métafont en une fonte PostSCript de type 1 ? . . . 81
6.10 Peut-on convertir une fonte Métafont en fonte True Type ? Peut-on utiliser les fontes Computer Modern avec d'autres logiciels que TeX ? . . . 81

7 Fontes PostScript **82**
7.1 Qu'est-ce que le PostScript ? . . . 82
7.2 Qu'est-ce qu'une fonte PostScript ? . . . 97
7.3 Comment créer soi-même une fonte PostScript de type 3 ? . . . 98
7.4 Comment créer soi-même une fonte PostScript de type 1 ? . . . 99
7.5 Comment avoir les contours d'une fonte de type 1 ? . . . 99
7.6 Comment transformer une fonte de type 1 en une fonte de type 3 ? . . . 99
7.7 Comment changer le codage d'une fonte PostScript ? . . . 100
7.8 Comment mettre des commandes PostScript dans un fichier LaTeX ? . . . 101
7.9 Comment visualiser une fonte PostScript ? . . . 102
7.10 Comment réaliser un catalogue de fontes PostScript ? . . . 103
7.11 Y a-t-il des versions PostScript des fontes Computer Modern ? . . . 105
7.12 Comment dire à dvips quelles sont les fontes PostScript ? . . . 105
7.13 Comment utiliser une fonte PostScript de manière ponctuelle dans un document LaTeX ? . . . 105
7.14 Comment utiliser une fonte PostScript de manière ponctuelle dans un document LaTeX (en tenant compte du codage) ? . . . 106
7.15 Comment utiliser plusieures fontes PostScript de manière ponctuelle dans un document LaTeX ? . . . 107
7.16 Comment utiliser une fonte PostScript comme fonte par défaut dans un document LaTeX ? . . . 107
7.17 Comment utiliser une fonte PostScript de manière ponctuelle dans un document LaTeX (vieille version de la même question — à effacer) ? . . . 108
7.18 Comment pencher une fonte PostScript ? . . . 109
7.19 Comment élargir ou condenser une fonte PostSCript ? . . . 109
7.20 Comment récupérer une fonte PostScript utilisée dans un fichier PDF ? . . . 110
7.21 Comment récupérer une fonte PostScript utilisée dans un fichier PostScript ? . . . 110

8 Fontes True Type (TTF) 111
8.1 Qu'est-ce qu'une fonte True Type? . . . 111
8.2 Comment créer une fonte True Type? Comment modifier une fonte True Type existante? . . . 111
8.3 Comment visualiser une fonte True Type? . . . 111
8.4 Comment réaliser un catalogue de fontes True Type? . . . 113
8.5 Comment utiliser une fonte True Type de manière ponctuelle dans un document LaTeX? . . . 115
8.6 Comment utiliser une fonte True Type de manière ponctuelle dans un document LaTeX? . . . 116
8.7 Comment utiliser plusieures fontes True Type de manière ponctuelle dans un document LaTeX? . . . 117
8.8 Comment utiliser plusieures fontes True Type comme fontes par défaut dans un document LaTeX? . . . 118
8.9 Comment pencher une fonte True Type? . . . 119
8.10 Comment condenser ou élargir une fonte True Type? . . . 120
8.11 Comment avoir des petites capitales à l'aide d'une fonte TTF? . . . 121
8.12 Comment épaissir une fonte TTF? . . . 122
8.13 Comment automatiser les réponses à toutes les questions précédentes? . . . 122
8.14 Comment faire pour que LaTeX/xdvi/dvips soit capable de calculer les fontes bitmaps tout seul? . . . 134
8.15 Y a-t-il un moyen plus simple d'utiliser des fontes TrueType? Est-il possible d'utiliser des fontes True Type avec pdfTeX? . . . 134
8.16 Comment obtenir les caractères manquants? . . . 136

9 Fontes virtuelles 137
9.1 Qu'est-ce qu'une fonte virtuelle? Que fait-on avec? . . . 137
9.2 Y a-t-il un moyen SIMPLE d'utiliser des fontes virtuelles? . . . 137
9.3 Qu'est-ce que fontinst? . . . 137
9.4 Comment utilise-t-on fontinst avec des fontes Métafont? . . . 137
9.5 Comment utilise-t-on fontinst avec des fontes PostScript? . . . 139
9.6 Comment utilise-t-on fontinst avec des fontes True Type? . . . 139
9.7 Comment changer le codage d'une fonte? Comment pencher une fonte? . . . 139
9.8 Exemples simples avec fontinst . . . 139
9.9 Comment intervertir deux caractères dans une fonte? . . . 140
9.10 Comment aller chercher un caractère dans une autre fonte? . . . 141
9.11 Comment modifier les métriques d'un caractère? . . . 141
9.12 Quelles sont les dimensions d'une fonte? Comment les changer? . . . 141
9.13 Comment ajouter un trait? . . . 142
9.14 Comment ajouter des instructions PostScript? . . . 143
9.15 Peut-on ajouter n'importe quelle instruction PostScript? . . . 143
9.16 Peut-on ajouter n'importe quelle instruction PostScript? . . . 145
9.17 Comment avoir des caractères gras quand il n'y en a pas? . . . 146
9.18 Comment ajouter un caractère? . . . 146
9.19 Comment ajouter une ligature? . . . 146
9.20 Comment avoir des ligatures ct et st? . . . 147
9.21 Comment changer l'espace entre deux caractères? . . . 148
9.22 Comment appliquer une même transformation à tous les caractères d'une fonte? À toutes les majuscules? À toutes les minuscules? . . . 148
9.23 Comment avoir les caractères manquants? . . . 150
9.24 Comment avoir une fonte soulignée? . . . 151
9.25 Comment avoir un j sans point quand il n'y en a pas? . . . 152
9.26 Peut-on manipuler des familles de fontes(plutôt que des fontes isolées) avec fontinst? . . . 152
9.27 Comment avoir une fonte grasse quand il n'y en a pas (sans fontes virtuelles)? . . . 153
9.28 Comment avoir une fonte grasse quand il n'y en a pas (avec des fontes virtuelles, premier essai, infructueux — A EFFACER)? . . . 153
9.29 Comment avoir une fonte grasse quand il n'y en a pas (avec des fontes virtuelles — A EFFACER)? . . . 156
9.30 Comment avoir une fonte « petites capitales » quand les caractères sont là? . . . 157
9.31 Comment avoir une fonte « petites capitales » quand les caractères ne sont pas là? . . . 157
9.32 Comment utiliser `vfinst`? . . . 160
9.33 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (1)? . . . 161
9.34 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (2)? . . . 162
9.35 Comment avoir des majuscules mathématiques droites avec des fontes virtuelles? . . . 164
9.36 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (3)? . . . 164
9.37 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (3)? . . . 165

9.38 Divers (réponses sans questions et questions sans réponses à classer) 165

10 À FAIRE 167

Chapitre 1

Introduction

Les utilisateurs de « mauvais » traitements de texte commerciaux ont l'habitude de changer très souvent de fonte et d'en avoir un choix très vaste. Les choses sont différentes sous LaTeX : on donne essentiellement à l'ordinateur la structure logique d'un document (« voici un titre de chapitre », « voici un titre de sous-chapitre », etc.) plutôt que la manière de formatter les choses (« je veux du times gras 20 points, centré », « je veux de l'helvética 15 points, justifié à droite, suivi d'un filet horizontal »). La puissance de LaTeX réside dans la possibilité de définir une fois pour toutes comment traduire la structure du document en instructions typographiques et ainsi de séparer le travail de la forme (la typographie) et du fond (le contenu sémantique du texte que vous tapez). Nous nous intéresserons à un point particulier de cette transformation : le choix et l'utilisation des fontes.

1.1 Où trouver la dernière version de ce document ?

Probablement sur ma page Web [1].

L'original est en sgml [2] et est est convertit en html [3] en texte [4] et en postScript [5] (ces conversions ne sont pas toujours très satisfaisantes).

1.2 Quels sont les autres documents sur le sujet ?

- Le LaTeX Navigator [6]
- Ce qu'a écrit D. Wright [7] sur l'utilisation des fontes PS,
- Le livre d'A. Hoenig, « TeX unbound », explique la plupart des manipulations possibles avec des fontes virtuelles.
- Using TrueType fonts with (La)TeX and pdf(La)TeX [8] (Damir Rakityansky)
- TrueType with teTEX : Quick and Dirty [9] (Nguyên-Đai Quý)
- Using TrueType fonts with teTeX and dvips [10] (Harald Harders)
- La page Web de `fontinst` [11] et la mailing list correspondante.
- Divers documents sur la typographie, par exemple la FAQ [12] de la mailing list du même nom.
- fntguide [13] pour les commandes NFSS
- Le source de LaTeX, en particulier les répertoires `base`, `mfnfss` et `psnfss`
- Le Metafont book [14] ou la documentation de Metapost [15].

---

1. `http://www.math.jussieu.fr/~zoonek/LaTeX/`
2. `http://www.math.jussieu.fr/~zoonek/LaTeX/Fontes/fontes.sgml`
3. `http://www.math.jussieu.fr/~zoonek/LaTeX/Fontes/fontes.html`
4. `http://www.math.jussieu.fr/~zoonek/LaTeX/Fontes/fontes.txt`
5. `http://www.math.jussieu.fr/~zoonek/LaTeX/Fontes/fontes.ps.bz2`
6. `http://www.loria.fr/tex/`
7. `http://www-theorie.physik.unizh.ch/~ichbin/texfonts/`
8. `http://www.radamir.com/tex/ttf-tex.htm`
9. `http://iris.ltas.ulg.ac.be/viettug/contrib/q/`
10. `ftp://ftp.dante.de/tex-archive/info/TrueType/ttf-tetex.html`
11. `http://www.tug.org/applications/fontinst/index.html`
12. `http://altern.org/typographie/general/`
13. `ftp://ftp.lip6.fr/pub/TeX/CTAN/macros/latex/doc/fntguide.dvi`
14. `ftp://ftp.lip6.fr/pub/TeX/CTAN/systems/knuth/mf/mfbook.tex.gz`
15. `http://cm.bell-labs.com/who/hobby/MetaPost.html`

- Certains articles des Cahiers GUTenberg[16], par exemple sur les fontes mathématiques[17], sur les formats de fichiers dvi, gf, tfm et vf[18], sur hz[19].
- Le texte de D.E. Knuth sur les fontes virtuelles[20]
- La documentation de `fontinst`, `vfinst`, `mathkit` et `mathinst`.
- Les livres d'Adobe[21], en particulier le Blue Book, le Red Book[22], sur le langage PostScript et le Black Book[23] sur les fontes PostScript de type 1. La documentation de ghostscript[24], et tout ce que vous pouvez trouver sur le langage PostScript[25].
- La documentation de `dvips`.
- Les forum de discussion `comp.fonts`[26], `comp.text.pdf`, `comp.text.tex`, `fr.comp.text.tex`, leur FAQ[27] et certaines listes de discussion du même genre[28],
- Pour les fontes en général, voir aussi la page de Luc Devroye[29].
- Font-HOWTO[30]

1.3 Où trouver des fontes Métafont, PostScript ou TTF ?

- Pour les fontes Métafont, il suffit de regarder sur CTAN[31], dans les répertoires `fonts`[32] et `languages`[33]. Il y a quelque temps, j'en ai réalisé un catalogue[34], qui est n'est probablement plus à jour.
- Pour les fontes Postscript, je suggère d'utiliser un moteur de recherche[35, 36, 37, 38]. On en trouve peu : il semblerait que la plupart des fontes PostScript soient commerciales.
- Pour les fontes TTF, je suggère, encore une fois, d'utiliser un moteur de recherche[39, 40, 41, 42]. On en trouve beaucoup plus. Certains de ces liens renvoient sur des fichiers compressés dans des formats assez bizarres (pour machintoshes). Vous pouvez trouver des outils de décompression un peu partout[43, 44], mais ne vous attendez pas à des miracles.

  Pour trouver des fontes sur le Web sans perdre trop de temps, je suggère d'utiliser un aspirateur de pages Web, comme par exemple `wget`. On peut le paralléliser de la manière suivante.

  - Commencer par récupéer une liste de pages Web suceptibles de contenir des fontes.

```
lynx -source http://jeff.cs.mcgill.ca/~luc/fonts.html |
  grep http: |
  perl -p -e 's/.*http:/http:/' |
  perl -p -e 's|[">].*||' |
  sort |
  uniq > todo
```

16. http://www.gutenberg.eu.org/pub/GUTenberg/publications
17. http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPS/25-bouche.ps.gz
18. http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPS/26-roegel.ps.gz
19. http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPS/27-karow.ps.gz
20. ftp://ftp.lip6.fr/pub/TeX/CTAN/info/virtual-fonts.knuth.gz
21. http://partners.adobe.com/asn/developer/technotes.html
22. http://partners.adobe.com/asn/developer/technotes.html
23. http://partners.adobe.com/asn/developer/technotes.html
24. http://www.cs.wisc.edu/~ghost/aladdin/
25. http://www.geocities.com/SiliconValley/5682/postscript.html
26. http://www.nwalsh.com/comp.fonts/FAQ/index.html
27. http://www.faqs.org/faqs/
28. http://www.tug.org/ListsArchives/
29. http://jeff.cs.mcgill.ca/~luc/fonts.html
30. http://ftp.lip6.fr/pub/linux/sunsite/docs/HOWTO/other-formats/html_single/Font-HOWTO.html
31. ftp://ftp.ibp.fr:/pub/TeX/CTAN/
32. ftp://ftp.ibp.fr:/pub/TeX/CTAN/fonts/
33. ftp://ftp.ibp.fr:/pub/TeX/CTAN/languages/
34. http://www.math.jussieu.fr/~zoonek/LaTeX/MF/liste.html
35. http://cuiwww.unige.ch/OSG/Widgets/Fonts/
36. http://www.ma.adfa.oz.au/Local/FontBrowser/fullname.html
37. http://www.cooltext.com/fonts/
38. http://www.loria.fr/services/tex/fontes/softkey/fontes-exemples.html
39. http://www.masterstech-home.com/The{\textunderscore}Library/Font{\textunderscore}Samples/FontIndexMain.html
40. http://www.netexplorers.com/member/Chuck/fonts/index.html
41. http://www.win-uk.net/{\textasciitilde}jherbert/ftp-archive/Windows/TrueType/
42. http://www.fontcafe.com/showcase/font{\textunderscore}garden/a.html
43. http://www.gromwin.demon.co.uk/vanessa/compress.htm
44. http://www.mcad.edu/ericb/xplat.comp.html

– Lancer wget, en parallèle, sur cette liste.

```
#! perl -w

use strict;
use constant AFAIRE => 1;
use constant DEJAFAIT => 2;

# On regarde le contenu du fichier todo.
my %todo;
open(TODO, "todo") || die "cannot open ‘todo’ for reading: $!";
while(<TODO>){
  chomp;
  $todo{$_} = AFAIRE;
}
close TODO;

# On regarde le contenu du fichier ‘done’
if( open(DONE, "done") ){
  while(<DONE>){
    chomp;
    $todo{$_} = DEJAFAIT;
  }
  close DONE;
} else {
  warn "Cannot open ‘done’ for reading: $!";
}

# On regarde ce qu’il y a à faire
my @actions =();
foreach (keys %todo){
  next if $todo{$_} == DEJAFAIT;
  push @actions, [\&doit, $_];
}

# Ce qu’il y aura à faire
sub doit () {
  my ($arg) = @_;
  system ’wget’, ’-r’, ’-l’, ’0’, ’-np’, ’-Q’, ’20m’,
                 ’-R’, ’doc,mov,avi,wav,exe,EXE,zip,ZIP,aif,aiff,au,mpg’,
                 "$arg";
  done("done", "$arg\n");
}

#########################################################

use strict;
use POSIX ":sys_wait_h"; # As the man page says...
use Fcntl ’:flock’; # import LOCK_* constants

my $max_children = 3;
my @pid;

sub execute ($) {
  my ($a) = @_;
  my @b = @$a;
  my $f = $b[0];
  my @arg = @b[1..$#b];
  &{$f}(@arg);
```

```
}

sub done () {
  my ($file, @arg) = @_;
  open(DONE, ">>$file") || die "cannot open $file for appending: $!";
  print STDERR "$$ Trying to write to $file\n";
  flock(DONE,LOCK_EX);
  seek(DONE, 0, 2);
  print STDERR "$$ Access granted\n";
  select DONE;
  print @arg;
  $|=1; # flush
  select STDOUT;
  flock(DONE,LOCK_UN);
  close DONE;
  print STDERR "$$ closing file\n";
}

print STDERR "** spawning $max_children processes\n";

for( my $i=0; $i<$max_children; $i++ ){

  $pid[$i] = fork();
  sleep 2;
  die "cannot fork: $!" if $pid[$i] == -1;
  if( $pid[$i] ){
    print STDERR "** spawned pid $pid[$i]\n";
    pop @actions;
  } else {
    print STDERR "$$ new child\n";
    execute(pop @actions) if @actions;
    print STDERR "$$ dieing\n";
    exit 0;
  }

}

print STDERR "** spawned a few processes\n";
print STDERR "** waiting for a child to die\n";

do {
  sleep 1;
  for( my $i=0; $i<$max_children; $i++ ){
    if( waitpid($pid[$i],&WNOHANG) == -1 ){
      print STDERR "** process number $i (pid:$pid[$i]) died\n";

      $pid[$i] = fork();
      sleep 2;
      die "cannot fork: $!" if $pid[$i] == -1;
      if( $pid[$i] ){
        print STDERR "** spawned pid $pid[$i]\n";
        pop @actions;
      } else {
        print STDERR "$$ new child\n";
        execute(pop @actions) if @actions;
        print STDERR "$$ dieing\n";
        exit 0;
      }
    }
```

```
  }
} while @actions;

print STDERR "** waiting for remaining children to die\n";
for( my $i=0; $i<$max_children; $i++ ){
  while( waitpid($pid[$i],&WNOHANG) != -1 ){ sleep 1 }
}
```

– On récupère ensuite les fichiers `*.zip` qui sont suceptibles de contenir des fontes, puis, à l'intérieur, les fichiers `*.ttf`.

```
mkdir ZIP
cd ZIP
for i in ../**/*.zip
do
  yes n | unzip $i
done
mv **/*(.) ./
rm -rf *(/)
rename 'y/A-Z/a-z/'
rm -f -- ^*.ttf
```

– Enfin, on fabrique un catalogue de fontes, comme expliqué plus loin (8.4)

Chapitre 2

Choses simples

2.1 Quelles sont les commandes de changement de fontes ?

Rappelons, c'est nécessaire, que les commandes LaTeX sont de deux types, les macros, telles que

```
\mamacro{premier argument}{second argument}
```

et les environements, tels que

```
\begin{monenvironnement}
  texte
\end{monenvironnement}
```

(Les macros « à la TeX », du genre `{\toto #1}` ou `{#1 \toto #2}` sont à proscrire, pour d'évidentes raisons pédagogiques.)

En mode texte, il existe des *macros* qui changent la fonte de leur argument, `\textrm`, `\textit`, `\textsl`, `\textbf`, `\texttt`, `\textsc`, `\textsf`, `\emph` et des *environements* `huge`, `LARGE`, `Large`, `large`, `footnotesize`, `small`, `tiny`, `rmfamily`, `sffamily`, `ttfamily`, `bfseries`, `mdseries`, `upshape`, `slshape`, `scshape`, `itshape`, `slshape`, `em`.

En mode mathématique (dès que l'on parle de mathématique, je suppose que l'on utilise les extensions `amsmath` et `amssymb`), il existe des macros `\mathnormal`, `\mathit`, `\mathrm`, `\mathbf`, `\mathfrak`, `\mathsf`, `\mathsl`, `\mathcal`, `\mathbb`, etc. `\pmb`, `\bm` (dans l'extension `bm`, qui remplace désormais `\boldsymbol`)

Vous vous demandez peut-être quelle est la différence entre `\mathit` et `\mathnormal`. La commande `\mathnormal` correspond à la fonte généralement utilisée en mode mathématique, ie, une fonte italique avec des espacements un peu spéciaux : les lettres que l'on tape sont considérées comme des symboles mathématiques distincts et sont donc un peu plus espacés, en particulier, il n'y a pas de ligarure. Au contraire, la fonte `\mathit` est une fonte italique normale, dans laquelle les lettres ne sont pas isolées mais forment *dans leur ensemble* un symbole mathématique. On utilise généralement une fonte droite.

```
\documentclass{article}
\usepackage{amsmath}
\DeclareMathOperator{\Diff}{Diff}
\begin{document}
Nous noterons $\Diff X$ (et pas $Diff\ X$ !)
le groupe des difféomorphismes
de la variété $X$.
\end{document}
```

Nous noterons $\operatorname{Diff} X$ (et pas $Diff\ X$!) le groupe des difféomorphismes de la variété X.

Dans certains cas *très rares*, on peut néanmoins utiliser de l'italique, avec un espacement normal.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{amsmath,amsfonts,mathrsfs}
```

```
%\DeclareMathOperator{\hom}{Hom}
\DeclareMathOperator{\Hom}{\mathit{Hom}}
\begin{document}

Nous noterons $\hom(F, G)$ l'ensemble des morphismes de faisceaux
de $F$ à $G$ et $\Hom(F, G)$ le \emph{faisceau}
des morphismes de $F$ vers $G$.

\end{document}
```

Nous noterons $\hom(\mathscr{F},\mathscr{G})$ l'ensemble des morphismes de faisceaux de $\mathscr{F}$ à $\mathscr{G}$ et $\mathit{Hom}(\mathscr{F},\mathscr{G})$ le *faisceau* des morphismes de $\mathscr{F}$ vers $\mathscr{G}$.

Pour que tout le mode mathématique soit en gras, on peut utiliser (avant de passer en mode mathématique) la commande `\mathversion{bold}`. Pour revenir à une épaisseur normale, la commande est `\mathversion{normal}`. C'est pertinent dans les titres (i.e., quand vous redéfinissez vous-même des commandes du genre `\chapter` ou `\section`.

2.2 Comment écrire plus grand ?

Si vous voulez écrire plus grand pour faire des transparents, regardez les classes du genre `seminar`[1] ou `foiltex`.

Si vous voulez écrire plus grand pour faire un poster, une affiche, vous pouvez utiliser l'extension `a0poster`.

Si vous voulez écrire un document avec la class `article` ou `book` et des fontes plus grosses, vous pouvez utiliser les options « 14pt » ou « 17pt » ; elles ne sont pas définies par défaut, mais vous pouvez récupérer les fichiers correspondants sur le Web[2].

Vous pouvez avoir envie d'écrire plus grand de manière ponctuelle. Par exemple, pour le titre du document, ou le titre des chapitres. Vous connaissez déjà les environements `huge`, `LARGE`, `Large`, `large`, `footnotesize`, `small` et `tiny`. Vous avez peut-être envie de les utiliser ainsi (ne riez pas, l'exemple est authentique – c'était même souligné).

```
\documentclass{article}
\begin{document}
\section{\begin{huge}\underline{Introduction}\end{huge}}
\end{document}
```

C'est une très mauvaise idée, car le titre des chapitres est réutilisé en plein d'autres endroits : en haut des pages ou dans la table des manières. On peut alors penser modifier cet exemple de la manière suivante.

```
\documentclass{article}
\begin{document}
\section[Introduction]{\begin{huge}\underline{Introduction}\end{huge}}
\end{document}
```

Bien que le résultat soit correct (c'est beaucoup trop grand par rapport à la taille du texte, le numéro du chapitre est trop petit et le soulignement est d'un assez mauvais goût — rappelons que sa seule utilisation est le remplacement du gras ou de l'italique quand on n'en a pas, par exemple sur une machine à écrire — mais c'est quand même mieux qu'avant), c'est toujours une mauvaise idée. Cela vous oblige à *répéter* ces instructions typographiques à chaque fois. La meilleure solution consiste à garder le même marquage du texte, à l'aide de la commande `\section`, avec des arguments normaux et lisibles, mais en changeant le comportement typographique de cette commande. La première idée qui vous vient à l'esprit est probablement de redéfinir la commande `\section`.

```
\documentclass{article}
\let\oldsection\section
\renewcommand{\section}[1]{%
  \oldsection[#1]{\begin{huge}\underline{#1}\end{huge}}%
}
\begin{document}
```

1. http://www.tug.org/applications/Seminar/
2. http://www.informatik.uni-freiburg.de/~may/extsizes.html

```
\section{Introduction}
\end{document}
```

C'est beaucoup mieux : le texte que l'on tape n'est plus encombré d'instructions typographiques (elles sont rejetées dans le préambule du document, i.e., à leur place). Mais ça ne marche pas : le commande `\section` est beaucoup plus compliquée que notre nouvelle commande : elle n'accepte pas d'argument facultatif ni de forme étoilée. On pourrait se dire que si on ne les utilise pas, ce n'est pas grave. Le problème, c'est que LaTeX lui-même utilise ces formes plus compliquées de la commande `\section`, par exemple dans la table des matières ou la bibliographie. Que faire si on a besoin d'une table des matières ou d'une bibliographie ? Je suggère de regarder dans le source de LaTeX (quelque part sur votre disque dur vous devriez trouver un gros fichier `latex.ltx` : c'est lui) comment les sections sont définies. On constate que la commande `\section` n'est pas définie dedans (mais c'est quand même le premier endroit où regarder) : elle est définie dans la classe `article.cls`.

```
\newcommand\section{\@startsection {section}{1}{\z@}%
                                   {-3.5ex \@plus -1ex \@minus -.2ex}%
                                   {2.3ex \@plus.2ex}%
                                   {\normalfont\Large\bfseries}}
```

Il va donc recopier ces lignes dans notre fichier et de les modifier.

```
\documentclass{article}
\usepackage{ulem}
% En dehors des fichiers *.sty, il faut toujours entourer
% les passages contenant le caractère @ des commandes
% \makeatletter et \makeatother.
\makeatletter
\renewcommand\section{\@startsection {section}{1}{\z@}%
                                   {-3.5ex \@plus -1ex \@minus -.2ex}%
                                   {2.3ex \@plus.2ex}%
                                   {\normalfont\huge\bfseries\uline}}
\makeatother
\begin{document}
\section{Introduction}
\end{document}
```

Cette fois-ci, ça marche.

Passons au dernier point sur la taille des fontes. Vous voulez peut être des caractères encore plus grands que `\huge` : est-ce possible ? Avant de répondre par l'affirmative, remarquons que c'est souvent une mauvaise idée : on risquerait de se retrouver avec des caractères de taille comparables, presque égales, mais différentes. Cela nuirait à l'homogénéité du document. Voyons maintenant comment faire. En feuilletant la suite de ce document, vous avez peut-être constaté qu'il y avait une commande `\fontsize`. Essayons de l'utiliser.

```
\documentclass{article}
\newcommand{\grand}[1]{\fontsize{100}{110}\selectfont}
\begin{document}
\end{document}
```

On constate que cela ne marche pas, et d'ailleurs, LaTeX nous le dit.

```
  LaTeX Font Warning: Font shape ‘T1/cmr/m/n’ in size <100> not available
(Font)              size <35.83> substituted on input line 7.
```

Pour éviter le problème d'hétérogénéité mentionné plus haut, il n'accepte que certaines tailles, mentionnées dans le fichier `t1cmr.fd`. Une solution consiste à recopier ce fichier dans le répertoire courrant et à le modifier (mais cela suppose que l'on en comprend le contenu, nous y reviendrons donc plus loin en 3.2 et 4.6). Une autre solution, beaucoup plus simple, consiste à utiliser des fontes que LaTeX nous autorise à utiliser à n'importe quelle taille, par exemple les fontes PostScript.

```
\documentclass{article}
\usepackage{pslatex,multido}
\newcommand{\doit}{\multido{\i=1+1}{20}{\fontsize{\i}{\i}\selectfont A}}
```

```
\begin{document}
\doit
\end{document}
```

AAAAAAAAAAAAAAAAAAAA

```
\documentclass{article}
\usepackage{mathpple,multido}
\newcommand{\doit}{\multido{\i=1+1}{20}{\fontsize{\i}{\i}\selectfont A}}
\begin{document}
\doit
\end{document}
```

AAAAAAAAAAAAAAAAAAAA

2.3 Comment changer de fontes pour tout le document ?

Dans cette partie, nous ne traiterons que des réponses simples à cette question : nous supposerons que tout le travail difficile a déjà été fait et se trouve réuni dans une extension (fichier `*.sty`). Il suffit alors de la charger à l'aide de la commande `\usepackage`.

Contrairement à des idées répandues, LaTeX n'est pas limité aux fontes Computer Modern : il peut utiliser les mêmes fontes que ses concurrents, sous à peu près n'importe quel format (Métafont, PostScript, True Type, hbf, etc.). Il faut toutefois prendre garde à ce que les fontes que l'on choisit comportent tous les caractères dont on a besoin : d'une part, les caractères accentués, d'autre part (selon la nature du document à taper), tous les symboles mathématiques. C'est justement là que le bât blesse : il n'y a qu'une demi-douzaine de fontes comportant ces caractères. (Nous signalerons beaucoup plus loin qu'il est possible de contourner ce problème en créant plus ou moins automatiquement des fontes mathématiques qui se marrient bien à une fonte texte donnée).

Voici tout d'abord les fontes gratuites possibles.

– Les fontes Computer Modern sont celles par défaut : il n'y a rien à préciser pour les utiliser.

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ la fonction définie par

$$f(x) = \frac{x + \sqrt{x^2 + 1}}{x^2 + x + 1}.$$

– Les fontes Concrete (pour le texte) et Euler (pour les mathématiques) ont été dessinées par D. Knuth pour son livre « concrete mahtematics ». Pour les utiliser, il suffit de rajouter les lignes suivante dans le préambule de votre document.

```
\usepackage{concrete}
\usepackage[mathcal,mathbf]{euler}
```

Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ la fonction définie par

$$f(x) = \frac{x + \sqrt{x^2 + 1}}{x^2 + x + 1}.$$

– Il y a aussi les fontes `cmbright`.

```
\usepackage{cmbright}
```

Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ la fonction définie par

$$f(x) = \frac{x + \sqrt{x^2 + 1}}{x^2 + x + 1}.$$

Voici maintenant quelques alternatives commerciales.

- Les fontes Mathtimes sont commerciales : il s'agit des fontes PostScript Times (qui sont gratuites) accompagnées d'une fonte mathématique correspondante. Si vous les avez achetées, vous pouvez les utiliser en mettant la ligne suivante dans le préambule de votre document.

```
\usepackage{mathtimes}
```

- Lucida est une fonte commerciale. C'est par exemple elle qui est utilisée dans le *LaTeX Companion*.
- BAMath[3] est une fonte mathématique commerciale complétant la fonte texte Baskerville.
- Informal Math[4] est une fonte mathématique (manuscrite) commerciale.
- Hvmath est une fonte mathématique sans sérif commerciale (Micropress Inc[5]) basée sur Helvética et bien adaptée à la réalisation de transparents. Signalons que les fichiers bitmap `*.pk` sont diffusés gratuitement[6]. C'est peut-être un peu compliqué à installer : j'ai procédé ainsi. Renommer tous les fichiers, de manière qu'ils soient en minuscules.

```
rename 'y/A-Z/a-z/' **/*
```

Ensuite, les fichiers `*.pk` n'ont pas le bon nom : ils s'appellent

```
300/432DPI/mhmi10.pk
```

au lieu de

```
300/mhmi10.432pk
```

On les renome donc :

```
rename 's#300/(.*)dpi/(.*)\.pk$#$2.$1pk#' 300/**/*(.)
```

Il suffit dès lors de mettre les fichiers `*pk`, `*.sty`, `*.fd`, `*.tfm`, etc. là où LaTeX ira les chercher (par exemple, dans le répertoire courrant — voir 3.4 pour une installation plus propre). Cela s'utilise ensuite tout à fait normalement :

```
\usepackage{hvmath}
```

Comme il n'y a que des fontes bitmaps à une résolution de 300 points par pouce, il convient de lancer `dvips` avec l'option `-D 300` (`xdvi` risque de ne pas fonctionner).

Soit $\phi : \mathbf{R} \longrightarrow \mathbf{R}$ la fonction définie par

$$\phi(x) = \frac{x + \sqrt{x^2+1}}{x^2+x+1}.$$

Voici maintenant quelques solutions incomplètes, mais qui peuvent s'avérer acceptables (la liste n'est sûrement pas exhaustive).

- Il existe un équivalent gratuit de Mathtimes : on prend les fontes Times, Helvética et Courrier pour le texte et on essaye de fabriquer une fonte mathématique à l'aide du Times italique et de la fonte Symbol. Il manque néanmoins certains caractères mathématiques importants, comme le coproduit.

```
\usepackage{pslatex}
```

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ la fonction définie par

$$f(x) = \frac{x+\sqrt{x^2+1}}{x^2+x+1}.$$

3. http://www.micropress-inc.com/samples/bafonts.htm
4. http://www.micropress-inc.com/samples/ifonts.htm
5. http://www.micropress-inc.com/fonts/
6. http://www.micropress-inc.com/samples/hfonts.htm

- Les extension `mathptm` et `mathptmx` sont comparables à `pslatex`, avec quelques différences. Voir le fichier `psnfss2e.dvi` pour les détails. ***** DONNER UN EXEMPLE *****
- On peut aussi utiliser la fonte Palatino. Les caractères mathématiques sont encore pris dans la fonte Symbol.

```
\usepackage{mathpple}
```

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ la fonction définie par

$$f(x) = \frac{x + \sqrt{x^2+1}}{x^2+x+1}.$$

Ce sont les seules solutions (toutes faites) qui vous permettent de taper du texte mathématique : nous allons maintenant voir d'autres changements de fontes possibles, mais il ne s'agit que de fontes texte. Si vous avez des mathématiques dans votre document, LaTeX utilisera les fontes Computer Modern (ou Euler, si on lui demande), qui risquent de choquer avec votre nouvelle fonte, car l'épaisseur des traits est différente, de même que la largeur ou la hauteur des caractères. Nous verrons beaucoup plus loin (9.37) comment résoudre ce problème.

- Pandora

```
\usepackage{pandora}
```

Je vous prie d'agréer, monsieur, l'expression de mes salutations distinguées.

Regardons ce qui se passe si on met du texte mathématique dans un document utilisant la fonte Pandora.

Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ une fonction continue et x_0 un réel.

On constate que les caractères sont trop clairs et trop petits. L'exemple suivant utilise la fonte Euler pour les mathématiques.

Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ une fonction continue et x_0 un réel.

S'il n'y a que très peu de mathématiques (juste des lettres ou des chiffres), il est possible d'utiliser la fonte texte en mode mathématique (voir 4.14).

Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ une fonction continue et x_0 un réel.

2.4 Où trouver les caractères textuels manquants ?

Les symboles les plus courrants sont définis dans `textcomp.sty`.

\textcircled	◯	\textquotestraightbase	‚	\textquotestraightdblbase	„
\texttwelveudash	–	\textthreequartersemdash	—	\textleftarrow	←
\textrightarrow	→	\textblank	␢	\textdollar	$
\textquotesingle	'	\textasteriskcentered	∗	\textdblhyphen	=
\textfractionsolidus	/	\textzerooldstyle	0	\textoneoldstyle	1
\texttwooldstyle	2	\textthreeoldstyle	3	\textfouroldstyle	4
\textfiveoldstyle	5	\textsixoldstyle	6	\textsevenoldstyle	7
\texteightoldstyle	8	\textnineoldstyle	9	\textlangle	⟨
\textminus	−	\textrangle	⟩	\textmho	℧
\textbigcircle	◯	\textohm	Ω	\textlbrackdbl	⟦
\textrbrackdbl	⟧	\textuparrow	↑	\textdownarrow	↓
\textasciigrave	`	\textborn	⋆	\textdivorced	⚮
\textdied	†	\textleaf	🍃	\textmarried	⚭
\textmusicalnote	♪	\texttildelow	~	\textdblhyphenchar	=
\textasciibreve	˘	\textasciicaron	ˇ	\textgravedbl	̏
\textacutedbl	˝	\textdagger	†	\textdaggerdbl	‡

`\textbardbl`	‖	`\textperthousand`	‰	`\textbullet`	•
`\textcelsius`	℃	`\textdollaroldstyle`	$	`\textcentoldstyle`	¢
`\textflorin`	ƒ	`\textcolonmonetary`	₡	`\textwon`	₩
`\textnaira`	₦	`\textguarani`	₲	`\textpeso`	₱
`\textlira`	₤	`\textrecipe`	℞	`\textinterrobang`	‽
`\textinterrobangdown`	⸘	`\textdong`	₫	`\texttrademark`	™
`\textpertenthousand`	‱	`\textpilcrow`	¶	`\textbaht`	฿
`\textnumero`	№	`\textdiscount`	⁒	`\textestimated`	℮
`\textopenbullet`	◦	`\textservicemark`	℠	`\textlquill`	⁅
`\textrquill`	⁆	`\textcent`	¢	`\textsterling`	£
`\textcurrency`	¤	`\textyen`	¥	`\textbrokenbar`	¦
`\textsection`	§	`\textasciidieresis`	¨	`\textcopyright`	©
`\textordfeminine`	ª	`\textcopyleft`	🄯	`\textlnot`	¬
`\textcircledP`	℗	`\textregistered`	®	`\textasciimacron`	¯
`\textdegree`	°	`\textpm`	±	`\texttwosuperior`	²
`\textthreesuperior`	³	`\textasciiacute`	´	`\textmu`	µ
`\textparagraph`	¶	`\textperiodcentered`	·	`\textreferencemark`	※
`\textonesuperior`	¹	`\textordmasculine`	º	`\textsurd`	√
`\textonequarter`	¼	`\textonehalf`	½	`\textthreequarters`	¾
`\texteuro`	€	`\texttimes`	×	`\textdiv`	÷
`\textsterling`	£	`\textdollar`	$	`\textperthousand`	‰

On a par exemple un symbole euro (le plus laid que j'aie jamais vu) ; diverses flèches, étoiles ou ronds, utilisables dans des énumérations ; des chiffres *oldstyle* ; la lettre grecque mu droite ; les symboles registered, trademark, copyright, copyleft ; les symboles pourcent, pourmille, degré Celcius, etc.

Il y a d'autres fontes contenant des symboles. Voici les caractères de l'extension `wasy`.

`\HF`	≈	`\Square`	□	`\astrosun`	☉
`\APLinv`	⌹	`\APLminus`	¯	`$\mho$`	℧
`$\Join$`	⋈	`$\Box$`	□	`$\Diamond$`	◇
`$\leadsto$`	⇝	`$\sqsubset$`	⊏	`$\sqsupset$`	⊐
`$\lhd$`	◁	`$\unlhd$`	⊴	`$\LHD$`	◀
`$\rhd$`	▷	`$\unrhd$`	⊵	`$\RHD$`	▶
`$\apprle$`	≲	`$\apprge$`	≳	`$\wasypropto$`	∝
`$\invneg$`	⌐	`$\ocircle$`	○	`$\logof$`	⊛
`$\varint$`	∫	`$\iint$`	∬	`$\iiint$`	∭
`$\varoint$`	∮	`$\oiint$`	∯		

Voici les caractères de l'extension `pifont` obtenus à l'aide de la commande `\Pisymbol{pzd}{???}`, où il faut remplacer `???` par un nombre.

`\Pisymbol{pzd}{33}`	✁	`\Pisymbol{pzd}{34}`	✂	`\Pisymbol{pzd}{35}`	✃
`\Pisymbol{pzd}{36}`	✄	`\Pisymbol{pzd}{37}`	☎	`\Pisymbol{pzd}{38}`	✆
`\Pisymbol{pzd}{39}`	✇	`\Pisymbol{pzd}{40}`	✈	`\Pisymbol{pzd}{41}`	✉
`\Pisymbol{pzd}{42}`	☛	`\Pisymbol{pzd}{43}`	☞	`\Pisymbol{pzd}{44}`	✌
`\Pisymbol{pzd}{45}`	✍	`\Pisymbol{pzd}{46}`	✎	`\Pisymbol{pzd}{47}`	✏
`\Pisymbol{pzd}{48}`	✐	`\Pisymbol{pzd}{49}`	✑	`\Pisymbol{pzd}{50}`	✒
`\Pisymbol{pzd}{51}`	✓	`\Pisymbol{pzd}{52}`	✔	`\Pisymbol{pzd}{53}`	✕
`\Pisymbol{pzd}{54}`	✖	`\Pisymbol{pzd}{55}`	✗	`\Pisymbol{pzd}{56}`	✘
`\Pisymbol{pzd}{57}`	✙	`\Pisymbol{pzd}{58}`	✚	`\Pisymbol{pzd}{59}`	✛
`\Pisymbol{pzd}{60}`	✜	`\Pisymbol{pzd}{61}`	✝	`\Pisymbol{pzd}{62}`	✞
`\Pisymbol{pzd}{63}`	✟	`\Pisymbol{pzd}{64}`	✠	`\Pisymbol{pzd}{65}`	✡
`\Pisymbol{pzd}{66}`	✢	`\Pisymbol{pzd}{67}`	✣	`\Pisymbol{pzd}{68}`	✤
`\Pisymbol{pzd}{69}`	✥	`\Pisymbol{pzd}{70}`	✦	`\Pisymbol{pzd}{71}`	✧
`\Pisymbol{pzd}{72}`	★	`\Pisymbol{pzd}{73}`	✩	`\Pisymbol{pzd}{74}`	✪
`\Pisymbol{pzd}{75}`	✫	`\Pisymbol{pzd}{76}`	✬	`\Pisymbol{pzd}{77}`	✭
`\Pisymbol{pzd}{78}`	✮	`\Pisymbol{pzd}{79}`	✯	`\Pisymbol{pzd}{80}`	✰
`\Pisymbol{pzd}{81}`	✱	`\Pisymbol{pzd}{82}`	✲	`\Pisymbol{pzd}{83}`	✳
`\Pisymbol{pzd}{84}`	✴	`\Pisymbol{pzd}{85}`	✵	`\Pisymbol{pzd}{86}`	✶

\Pisymbol{pzd}{87}	✷	\Pisymbol{pzd}{88}	✸	\Pisymbol{pzd}{89}	✹
\Pisymbol{pzd}{90}	✺	\Pisymbol{pzd}{91}	✻	\Pisymbol{pzd}{92}	✼
\Pisymbol{pzd}{93}	✽	\Pisymbol{pzd}{94}	✾	\Pisymbol{pzd}{95}	✿
\Pisymbol{pzd}{96}	❀	\Pisymbol{pzd}{97}	❁	\Pisymbol{pzd}{98}	❂
\Pisymbol{pzd}{99}	❃	\Pisymbol{pzd}{100}	❄	\Pisymbol{pzd}{101}	❅
\Pisymbol{pzd}{102}	❆	\Pisymbol{pzd}{103}	❇	\Pisymbol{pzd}{104}	❈
\Pisymbol{pzd}{105}	❉	\Pisymbol{pzd}{106}	❊	\Pisymbol{pzd}{107}	❋
\Pisymbol{pzd}{108}	●	\Pisymbol{pzd}{109}	❍	\Pisymbol{pzd}{110}	■
\Pisymbol{pzd}{111}	❏	\Pisymbol{pzd}{112}	❐	\Pisymbol{pzd}{113}	❑
\Pisymbol{pzd}{114}	❒	\Pisymbol{pzd}{115}	▲	\Pisymbol{pzd}{116}	▼
\Pisymbol{pzd}{117}	◆	\Pisymbol{pzd}{118}	❖	\Pisymbol{pzd}{119}	◗
\Pisymbol{pzd}{120}	❘	\Pisymbol{pzd}{121}	❙	\Pisymbol{pzd}{122}	❚
\Pisymbol{pzd}{123}	❛	\Pisymbol{pzd}{124}	❜	\Pisymbol{pzd}{125}	❝
\Pisymbol{pzd}{126}	❞	\Pisymbol{pzd}{161}	❡	\Pisymbol{pzd}{162}	❢
\Pisymbol{pzd}{163}	❣	\Pisymbol{pzd}{164}	❤	\Pisymbol{pzd}{165}	❥
\Pisymbol{pzd}{166}	❦	\Pisymbol{pzd}{167}	❧	\Pisymbol{pzd}{168}	♣
\Pisymbol{pzd}{169}	♦	\Pisymbol{pzd}{170}	♥	\Pisymbol{pzd}{171}	♠
\Pisymbol{pzd}{172}	①	\Pisymbol{pzd}{173}	②	\Pisymbol{pzd}{174}	③
\Pisymbol{pzd}{175}	④	\Pisymbol{pzd}{176}	⑤	\Pisymbol{pzd}{177}	⑥
\Pisymbol{pzd}{178}	⑦	\Pisymbol{pzd}{179}	⑧	\Pisymbol{pzd}{180}	⑨
\Pisymbol{pzd}{181}	⑩	\Pisymbol{pzd}{182}	❶	\Pisymbol{pzd}{183}	❷
\Pisymbol{pzd}{184}	❸	\Pisymbol{pzd}{185}	❹	\Pisymbol{pzd}{186}	❺
\Pisymbol{pzd}{187}	❻	\Pisymbol{pzd}{188}	❼	\Pisymbol{pzd}{189}	❽
\Pisymbol{pzd}{190}	❾	\Pisymbol{pzd}{191}	❿	\Pisymbol{pzd}{192}	➀
\Pisymbol{pzd}{193}	➁	\Pisymbol{pzd}{194}	➂	\Pisymbol{pzd}{195}	➃
\Pisymbol{pzd}{196}	➄	\Pisymbol{pzd}{197}	➅	\Pisymbol{pzd}{198}	➆
\Pisymbol{pzd}{199}	➇	\Pisymbol{pzd}{200}	➈	\Pisymbol{pzd}{201}	➉
\Pisymbol{pzd}{202}	➊	\Pisymbol{pzd}{203}	➋	\Pisymbol{pzd}{204}	➌
\Pisymbol{pzd}{205}	➍	\Pisymbol{pzd}{206}	➎	\Pisymbol{pzd}{207}	➏
\Pisymbol{pzd}{208}	➐	\Pisymbol{pzd}{209}	➑	\Pisymbol{pzd}{210}	➒
\Pisymbol{pzd}{211}	➓	\Pisymbol{pzd}{212}	➔	\Pisymbol{pzd}{213}	→
\Pisymbol{pzd}{214}	↔	\Pisymbol{pzd}{215}	↕	\Pisymbol{pzd}{216}	➘
\Pisymbol{pzd}{217}	➙	\Pisymbol{pzd}{218}	➚	\Pisymbol{pzd}{219}	➛
\Pisymbol{pzd}{220}	➜	\Pisymbol{pzd}{221}	➝	\Pisymbol{pzd}{222}	➞
\Pisymbol{pzd}{223}	➟	\Pisymbol{pzd}{224}	➠	\Pisymbol{pzd}{225}	➡
\Pisymbol{pzd}{226}	➢	\Pisymbol{pzd}{227}	➣	\Pisymbol{pzd}{228}	➤
\Pisymbol{pzd}{229}	➥	\Pisymbol{pzd}{230}	➦	\Pisymbol{pzd}{231}	➧
\Pisymbol{pzd}{232}	➨	\Pisymbol{pzd}{233}	➩	\Pisymbol{pzd}{234}	➪
\Pisymbol{pzd}{235}	➫	\Pisymbol{pzd}{236}	➬	\Pisymbol{pzd}{237}	➭
\Pisymbol{pzd}{238}	➮	\Pisymbol{pzd}{239}	➯	\Pisymbol{pzd}{240}	
\Pisymbol{pzd}{241}	➱	\Pisymbol{pzd}{242}	➲	\Pisymbol{pzd}{243}	➳
\Pisymbol{pzd}{244}	➴	\Pisymbol{pzd}{245}	➵	\Pisymbol{pzd}{246}	➶
\Pisymbol{pzd}{247}	➷	\Pisymbol{pzd}{248}	➸	\Pisymbol{pzd}{249}	➹
\Pisymbol{pzd}{250}	➺	\Pisymbol{pzd}{251}	➻	\Pisymbol{pzd}{252}	➼
\Pisymbol{pzd}{253}	➽	\Pisymbol{pzd}{254}	➾		

Voici les caractères de l'extension `pifont` obtenus à l'aide de la commande `\Pisymbol{psy}{???}`, où il faut remplacer `???` par un nombre.

\Pisymbol{psy}{33}	!	\Pisymbol{psy}{34}	∀	\Pisymbol{psy}{35}	#
\Pisymbol{psy}{36}	∃	\Pisymbol{psy}{37}	%	\Pisymbol{psy}{38}	&
\Pisymbol{psy}{39}	∍	\Pisymbol{psy}{40}	(	\Pisymbol{psy}{41}	)
\Pisymbol{psy}{42}	∗	\Pisymbol{psy}{43}	+	\Pisymbol{psy}{44}	,
\Pisymbol{psy}{45}	−	\Pisymbol{psy}{46}	.	\Pisymbol{psy}{47}	/
\Pisymbol{psy}{48}	0	\Pisymbol{psy}{49}	1	\Pisymbol{psy}{50}	2
\Pisymbol{psy}{51}	3	\Pisymbol{psy}{52}	4	\Pisymbol{psy}{53}	5
\Pisymbol{psy}{54}	6	\Pisymbol{psy}{55}	7	\Pisymbol{psy}{56}	8
\Pisymbol{psy}{57}	9	\Pisymbol{psy}{58}	:	\Pisymbol{psy}{59}	;
\Pisymbol{psy}{60}	<	\Pisymbol{psy}{61}	=	\Pisymbol{psy}{62}	>

`\Pisymbol{psy}{63}`	?	`\Pisymbol{psy}{64}`	≅	`\Pisymbol{psy}{65}`	Α
`\Pisymbol{psy}{66}`	Β	`\Pisymbol{psy}{67}`	Χ	`\Pisymbol{psy}{68}`	Δ
`\Pisymbol{psy}{69}`	Ε	`\Pisymbol{psy}{70}`	Φ	`\Pisymbol{psy}{71}`	Γ
`\Pisymbol{psy}{72}`	Η	`\Pisymbol{psy}{73}`	Ι	`\Pisymbol{psy}{74}`	ϑ
`\Pisymbol{psy}{75}`	Κ	`\Pisymbol{psy}{76}`	Λ	`\Pisymbol{psy}{77}`	Μ
`\Pisymbol{psy}{78}`	Ν	`\Pisymbol{psy}{79}`	Ο	`\Pisymbol{psy}{80}`	Π
`\Pisymbol{psy}{81}`	Θ	`\Pisymbol{psy}{82}`	Ρ	`\Pisymbol{psy}{83}`	Σ
`\Pisymbol{psy}{84}`	Τ	`\Pisymbol{psy}{85}`	Υ	`\Pisymbol{psy}{86}`	ς
`\Pisymbol{psy}{87}`	Ω	`\Pisymbol{psy}{88}`	Ξ	`\Pisymbol{psy}{89}`	Ψ
`\Pisymbol{psy}{90}`	Ζ	`\Pisymbol{psy}{91}`	[	`\Pisymbol{psy}{92}`	∴
`\Pisymbol{psy}{93}`	]	`\Pisymbol{psy}{94}`	⊥	`\Pisymbol{psy}{95}`	_
`\Pisymbol{psy}{96}`	‾	`\Pisymbol{psy}{97}`	α	`\Pisymbol{psy}{98}`	β
`\Pisymbol{psy}{99}`	χ	`\Pisymbol{psy}{100}`	δ	`\Pisymbol{psy}{101}`	ε
`\Pisymbol{psy}{102}`	ϕ	`\Pisymbol{psy}{103}`	γ	`\Pisymbol{psy}{104}`	η
`\Pisymbol{psy}{105}`	ι	`\Pisymbol{psy}{106}`	φ	`\Pisymbol{psy}{107}`	κ
`\Pisymbol{psy}{108}`	λ	`\Pisymbol{psy}{109}`	μ	`\Pisymbol{psy}{110}`	ν
`\Pisymbol{psy}{111}`	ο	`\Pisymbol{psy}{112}`	π	`\Pisymbol{psy}{113}`	θ
`\Pisymbol{psy}{114}`	ρ	`\Pisymbol{psy}{115}`	σ	`\Pisymbol{psy}{116}`	τ
`\Pisymbol{psy}{117}`	υ	`\Pisymbol{psy}{118}`	ϖ	`\Pisymbol{psy}{119}`	ω
`\Pisymbol{psy}{120}`	ξ	`\Pisymbol{psy}{121}`	ψ	`\Pisymbol{psy}{122}`	ζ
`\Pisymbol{psy}{123}`	{	`\Pisymbol{psy}{124}`	|	`\Pisymbol{psy}{125}`	}
`\Pisymbol{psy}{126}`	~	`\Pisymbol{psy}{161}`	ϒ	`\Pisymbol{psy}{162}`	′
`\Pisymbol{psy}{163}`	≤	`\Pisymbol{psy}{164}`	⁄	`\Pisymbol{psy}{165}`	∞
`\Pisymbol{psy}{166}`	ƒ	`\Pisymbol{psy}{167}`	♣	`\Pisymbol{psy}{168}`	♦
`\Pisymbol{psy}{169}`	♥	`\Pisymbol{psy}{170}`	♠	`\Pisymbol{psy}{171}`	↔
`\Pisymbol{psy}{172}`	←	`\Pisymbol{psy}{173}`	↑	`\Pisymbol{psy}{174}`	→
`\Pisymbol{psy}{175}`	↓	`\Pisymbol{psy}{176}`	°	`\Pisymbol{psy}{177}`	±
`\Pisymbol{psy}{178}`	″	`\Pisymbol{psy}{179}`	≥	`\Pisymbol{psy}{180}`	×
`\Pisymbol{psy}{181}`	∝	`\Pisymbol{psy}{182}`	∂	`\Pisymbol{psy}{183}`	•
`\Pisymbol{psy}{184}`	÷	`\Pisymbol{psy}{185}`	≠	`\Pisymbol{psy}{186}`	≡
`\Pisymbol{psy}{187}`	≈	`\Pisymbol{psy}{188}`	…	`\Pisymbol{psy}{189}`	⏐
`\Pisymbol{psy}{190}`	⎯	`\Pisymbol{psy}{191}`	↵	`\Pisymbol{psy}{192}`	ℵ
`\Pisymbol{psy}{193}`	ℑ	`\Pisymbol{psy}{194}`	ℜ	`\Pisymbol{psy}{195}`	℘
`\Pisymbol{psy}{196}`	⊗	`\Pisymbol{psy}{197}`	⊕	`\Pisymbol{psy}{198}`	∅
`\Pisymbol{psy}{199}`	∩	`\Pisymbol{psy}{200}`	∪	`\Pisymbol{psy}{201}`	⊃
`\Pisymbol{psy}{202}`	⊇	`\Pisymbol{psy}{203}`	⊄	`\Pisymbol{psy}{204}`	⊂
`\Pisymbol{psy}{205}`	⊆	`\Pisymbol{psy}{206}`	∈	`\Pisymbol{psy}{207}`	∉
`\Pisymbol{psy}{208}`	∠	`\Pisymbol{psy}{209}`	∇	`\Pisymbol{psy}{210}`	®
`\Pisymbol{psy}{211}`	©	`\Pisymbol{psy}{212}`	™	`\Pisymbol{psy}{213}`	∏
`\Pisymbol{psy}{214}`	√	`\Pisymbol{psy}{215}`	⋅	`\Pisymbol{psy}{216}`	¬
`\Pisymbol{psy}{217}`	∧	`\Pisymbol{psy}{218}`	∨	`\Pisymbol{psy}{219}`	⇔
`\Pisymbol{psy}{220}`	⇐	`\Pisymbol{psy}{221}`	⇑	`\Pisymbol{psy}{222}`	⇒
`\Pisymbol{psy}{223}`	⇓	`\Pisymbol{psy}{224}`	◊	`\Pisymbol{psy}{225}`	〈
`\Pisymbol{psy}{226}`	®	`\Pisymbol{psy}{227}`	©	`\Pisymbol{psy}{228}`	™
`\Pisymbol{psy}{229}`	∑	`\Pisymbol{psy}{230}`	⎛	`\Pisymbol{psy}{231}`	⎜
`\Pisymbol{psy}{232}`	⎝	`\Pisymbol{psy}{233}`	⎡	`\Pisymbol{psy}{234}`	⎢
`\Pisymbol{psy}{235}`	⎣	`\Pisymbol{psy}{236}`	⎧	`\Pisymbol{psy}{237}`	⎨
`\Pisymbol{psy}{238}`	⎩	`\Pisymbol{psy}{239}`	⎪	`\Pisymbol{psy}{240}`	
`\Pisymbol{psy}{241}`	〉	`\Pisymbol{psy}{242}`	∫	`\Pisymbol{psy}{243}`	⌠
`\Pisymbol{psy}{244}`	⎮	`\Pisymbol{psy}{245}`	⌡	`\Pisymbol{psy}{246}`	⎞
`\Pisymbol{psy}{247}`	⎟	`\Pisymbol{psy}{248}`	⎠	`\Pisymbol{psy}{249}`	⎤
`\Pisymbol{psy}{250}`	⎥	`\Pisymbol{psy}{251}`	⎦	`\Pisymbol{psy}{252}`	⎫
`\Pisymbol{psy}{253}`	⎬	`\Pisymbol{psy}{254}`	⎭		

Il y a encore d'autres symboles dans `marvosym`.

`\Stopsign`	🛑	`\Beam`	⎓	`\Bearing`	⧋
`\Loosebearing`	⧋	`\Fixedbearing`	⧋	`\Lefttorque`	↶

\Righttorque	↻	\Force	↓	\Lineload	╥╥╥
\Snowflake	❄	\Kross	✠	\Circles	◎
\MVZero	**0**	\MVOne	**1**	\MVTwo	**2**
\MVThree	**3**	\MVFour	**4**	\MVFive	**5**
\MVSix	**6**	\MVSeven	**7**	\MVEight	**8**
\MVNine	**9**	\Corresponds	≙	\MVAt	**@**
\Pickup	⌕	\Letter	✉	\CEsign	CE
\Lightning	ϟ	\FHBOlogo		\Mobilefone	📱
\Industry	🏭	\Coffeecup	☕	\Yingyang	☯
\Vectorarrowhigh	→	\Rightscissors	✂	\Kutline	---
\Leftscissors	✂	\Telefon	☎	\Clocklogo	⏲
\Checkedbox	☑	\Crossedbox	☒	\Pointinghand	☞
\Writinghand	✍	\EURhv	€	\EURcr	€
\EURtm	€	\FHBOLOGO		\Info	ℹ
\Football	⚽	\Vectorarrow	→	\Cutright	✂
\Cutline	---	\Cutleft	✂	\fax	FAX
\FAX	FAX	\Faxmachine	📠	\Cross	✝
\Celtcross	✥	\Ankh	☥	\Heart	♡
\Flatsteel	▬	\Squarepipe	◻	\Rectpipe	▯
\Lsteel	**L**	\TTsteel	**I**	\Circpipe	**O**
\Tsteel	**T**	\EUR	€	\Frowny	☹
\Smiley	☺	\Bicycle	🚲	\Sun	☉
\Moon	☽	\Mercury	☿	\Venus	♀
\Mars	♂	\Jupiter	♃	\Saturn	♄
\Uranus	♅	\Neptune	♆	\Pluto	♇
\Womanside		\Manside		\Womanfront	
\Manfront		\Aries	♈	\Taurus	♉
\Gemini	♊	\Cancer	♋	\Leo	♌
\Virgo	♍	\Libra	♎	\Scorpio	♏
\Sagittarius	♐	\Capricorn	♑	\Aquarius	♒
\Pisces	♓	\MVA	**A**	\MVp	**p**
\Squaredot	·	\Bat	🦇	\Womanface	
\MartinVogel					

2.5 Comment avoir un symbole Euro ?

Il y a un nombre incroyable de fontes comportant des symboles euro et un nombre encore plus grand d'extensions pour les utiliser. Je choisis (de manière tout à fait arbitraire) l'extension `eurofont`, qui permet d'utiliser le symbole euro provenant de diverses fontes.

Si on ne précise rien, il essaye d'utiliser le symbole Euro dans les fontes qu'il trouve, généralement Text Companion.

```
\documentclass{minimal}
\usepackage{eurofont}
\begin{document}
Ceci est un billet de \euros{500}.
\end{document}
```

Ceci est un billet de €500.

Il est possible d'utiliser les fontes Marvosym (avec les fontes PostScript standard), si elles sont là.

```
\documentclass{minimal}
\usepackage{pslatex,marvosym}
\usepackage[marvosym]{eurofont}
\begin{document}
Ceci est un billet de \euros{500}.
\end{document}
```

Ceci est un billet de € 500.

Il est possible d'utiliser `eurosym`.

```
\documentclass{minimal}
\usepackage[eurosym]{eurofont}
\begin{document}
Ceci est un billet de \euros{500}.
\end{document}
```

Ceci est un billet de €500.

Il est possible d'utiliser les fontes d'Adobe, si elles sont là. Si elles ne sont pas déjà installées, on peut procéder comme suit. Commencer par récupérer le fichier `eurofont`, quelque part sur le Web. Le décompresser,

```
unzip eurofont.zip
```

Renommer les fontes

```
mv _1______.PFB zpeurs.pfb
mv _1I_____.PFB zpeuris.pfb
mv _1B_____.PFB zpeubs.pfb
mv _1BI____.PFB zpeubis.pfb
mv _3______.PFB zpeur.pfb
mv _3I_____.PFB zpeuri.pfb
mv _3B_____.PFB zpeub.pfb
mv _3BI____.PFB zpeubi.pfb
mv _2______.PFB zpeurt.pfb
mv _2I_____.PFB zpeurit.pfb
mv _2B_____.PFB zpeubt.pfb
mv _2BI____.PFB zpeubit.pfb

mv _1______.AFM zpeurs.afm
mv _1I_____.AFM zpeuris.afm
mv _1B_____.AFM zpeubs.afm
mv _1BI____.AFM zpeubis.afm
mv _3______.AFM zpeur.afm
mv _3I_____.AFM zpeuri.afm
mv _3B_____.AFM zpeub.afm
mv _3BI____.AFM zpeubi.afm
mv _2______.AFM zpeurt.afm
mv _2I_____.AFM zpeurit.afm
mv _2B_____.AFM zpeubt.afm
mv _2BI____.AFM zpeubit.afm
```

Dire à dvips de lire le contenu du fichier `zpeu.map`, en modifiant votre `~/.dvipsrc`

```
M ljfive
D 600
% p +zpeu.map
```

```
\documentclass{minimal}
\usepackage[adobeeurofonts]{eurofont}
\begin{document}
Ceci est un billet de \euros{500}.
\end{document}
```

Ceci est un billet de € 500.

Il est aussi possible d'utiliser le caractère de `china2e`, à l'aide de la commande `\chinaeeuro`.

Ceci est un billet de €500.

Revoici une liste des symboles disponibles, à l'aide des commandes `\chinaeeuro`, `\cmeuro`, `\sanseuro`, `\serifeuro`, `\monoeuro`, `\zpeureuro`, `\zpeusseuro`, `\zpeutteuro`, `\marvosymsanseuro` et `\marvosymserifeuro`.

€ €€€€€€€€€

2.6 Comment avoir des « chiffres elzéviriens » ?

(Les anglais parlent de *oldstyle numbers*.) Pour une utilisation ponctuelle, on peut recourrit à la commande `\oldstylenums`. L'extension `eco.sty` répond permet de les utiliser par défaut (pour les fontes Computer Modern, sous le codage T1).

```
\documentclass{minimal}
\usepackage[T1]{fontenc}
\usepackage{eco}
\begin{document}
  Nous sommes le 31 décembre 1999.
\end{document}
```

Nous sommes le 31 décembre 1999.

2.7 Où trouver les caractères mathématiques manquants ?

La plupart des symboles mathématiques dont on a besoin sont dans les extensions `amssymb` et `stmaryrd`.

`$\boxdot$`	$\boxdot$	`$\boxplus$`	$\boxplus$	`$\boxtimes$`	$\boxtimes$
`$\square$`	$\square$	`$\blacksquare$`	$\blacksquare$	`$\centerdot$`	$\centerdot$
`$\lozenge$`	$\lozenge$	`$\blacklozenge$`	$\blacklozenge$	`$\circlearrowright$`	$\circlearrowright$
`$\circlearrowleft$`	$\circlearrowleft$	`$\rightleftharpoons$`	$\rightleftharpoons$	`$\leftrightharpoons$`	$\leftrightharpoons$
`$\boxminus$`	$\boxminus$	`$\Vdash$`	$\Vdash$	`$\Vvdash$`	$\Vvdash$
`$\vDash$`	$\vDash$	`$\twoheadrightarrow$`	$\twoheadrightarrow$	`$\twoheadleftarrow$`	$\twoheadleftarrow$
`$\leftleftarrows$`	$\leftleftarrows$	`$\rightrightarrows$`	$\rightrightarrows$	`$\upuparrows$`	$\upuparrows$
`$\downdownarrows$`	$\downdownarrows$	`$\upharpoonright$`	$\upharpoonright$	`$\downharpoonright$`	$\downharpoonright$
`$\upharpoonleft$`	$\upharpoonleft$	`$\downharpoonleft$`	$\downharpoonleft$	`$\rightarrowtail$`	$\rightarrowtail$
`$\leftarrowtail$`	$\leftarrowtail$	`$\leftrightarrows$`	$\leftrightarrows$	`$\rightleftarrows$`	$\rightleftarrows$
`$\Lsh$`	$\Lsh$	`$\Rsh$`	$\Rsh$	`$\rightsquigarrow$`	$\rightsquigarrow$
`$\leftrightsquigarrow$`	$\leftrightsquigarrow$	`$\looparrowleft$`	$\looparrowleft$	`$\looparrowright$`	$\looparrowright$
`$\circeq$`	$\circeq$	`$\succsim$`	$\succsim$	`$\gtrsim$`	$\gtrsim$
`$\gtrapprox$`	$\gtrapprox$	`$\multimap$`	$\multimap$	`$\therefore$`	$\therefore$
`$\because$`	$\because$	`$\doteqdot$`	$\doteqdot$	`$\triangleq$`	$\triangleq$
`$\precsim$`	$\precsim$	`$\lesssim$`	$\lesssim$	`$\lessapprox$`	$\lessapprox$
`$\eqslantless$`	$\eqslantless$	`$\eqslantgtr$`	$\eqslantgtr$	`$\curlyeqprec$`	$\curlyeqprec$
`$\curlyeqsucc$`	$\curlyeqsucc$	`$\preccurlyeq$`	$\preccurlyeq$	`$\leqq$`	$\leqq$
`$\leqslant$`	$\leqslant$	`$\lessgtr$`	$\lessgtr$	`$\backprime$`	$\backprime$
`$\risingdotseq$`	$\risingdotseq$	`$\fallingdotseq$`	$\fallingdotseq$	`$\succcurlyeq$`	$\succcurlyeq$
`$\geqq$`	$\geqq$	`$\geqslant$`	$\geqslant$	`$\gtrless$`	$\gtrless$
`$\sqsubset$`	$\sqsubset$	`$\sqsupset$`	$\sqsupset$	`$\vartriangleright$`	$\vartriangleright$
`$\vartriangleleft$`	$\vartriangleleft$	`$\trianglerighteq$`	$\trianglerighteq$	`$\trianglelefteq$`	$\trianglelefteq$
`$\bigstar$`	$\bigstar$	`$\between$`	$\between$	`$\blacktriangledown$`	$\blacktriangledown$
`$\blacktriangleright$`	$\blacktriangleright$	`$\blacktriangleleft$`	$\blacktriangleleft$	`$\vartriangle$`	$\vartriangle$
`$\blacktriangle$`	$\blacktriangle$	`$\triangledown$`	$\triangledown$	`$\eqcirc$`	$\eqcirc$
`$\lesseqgtr$`	$\lesseqgtr$	`$\gtreqless$`	$\gtreqless$	`$\lesseqqgtr$`	$\lesseqqgtr$
`$\gtreqqless$`	$\gtreqqless$	`$\Rrightarrow$`	$\Rrightarrow$	`$\Lleftarrow$`	$\Lleftarrow$

`$\veebar$`	$\veebar$	`$\barwedge$`	$\barwedge$	`$\doublebarwedge$`	$\doublebarwedge$
`$\angle$`	$\angle$	`$\measuredangle$`	$\measuredangle$	`$\sphericalangle$`	$\sphericalangle$
`$\varpropto$`	$\varpropto$	`$\smallsmile$`	$\smallsmile$	`$\smallfrown$`	$\smallfrown$
`$\Subset$`	$\Subset$	`$\Supset$`	$\Supset$	`$\Cup$`	$\Cup$
`$\Cap$`	$\Cap$	`$\curlywedge$`	$\curlywedge$	`$\curlyvee$`	$\curlyvee$
`$\leftthreetimes$`	$\leftthreetimes$	`$\rightthreetimes$`	$\rightthreetimes$	`$\subseteqq$`	$\subseteqq$
`$\supseteqq$`	$\supseteqq$	`$\bumpeq$`	$\bumpeq$	`$\Bumpeq$`	$\Bumpeq$
`$\lll$`	$\lll$	`$\ggg$`	$\ggg$	`$\circledS$`	$\circledS$
`$\pitchfork$`	$\pitchfork$	`$\dotplus$`	$\dotplus$	`$\backsim$`	$\backsim$
`$\backsimeq$`	$\backsimeq$	`$\complement$`	$\complement$	`$\intercal$`	$\intercal$
`$\circledcirc$`	$\circledcirc$	`$\circledast$`	$\circledast$	`$\circleddash$`	$\circleddash$
`$\lvertneqq$`	$\lvertneqq$	`$\gvertneqq$`	$\gvertneqq$	`$\nleq$`	$\nleq$
`$\ngeq$`	$\ngeq$	`$\nless$`	$\nless$	`$\ngtr$`	$\ngtr$
`$\nprec$`	$\nprec$	`$\nsucc$`	$\nsucc$	`$\lneqq$`	$\lneqq$
`$\gneqq$`	$\gneqq$	`$\nleqslant$`	$\nleqslant$	`$\ngeqslant$`	$\ngeqslant$
`$\lneq$`	$\lneq$	`$\gneq$`	$\gneq$	`$\npreceq$`	$\npreceq$
`$\nsucceq$`	$\nsucceq$	`$\precnsim$`	$\precnsim$	`$\succnsim$`	$\succnsim$
`$\lnsim$`	$\lnsim$	`$\gnsim$`	$\gnsim$	`$\nleqq$`	$\nleqq$
`$\ngeqq$`	$\ngeqq$	`$\precneqq$`	$\precneqq$	`$\succneqq$`	$\succneqq$
`$\precnapprox$`	$\precnapprox$	`$\succnapprox$`	$\succnapprox$	`$\lnapprox$`	$\lnapprox$
`$\gnapprox$`	$\gnapprox$	`$\nsim$`	$\nsim$	`$\ncong$`	$\ncong$
`$\diagup$`	$\diagup$	`$\diagdown$`	$\diagdown$	`$\varsubsetneq$`	$\varsubsetneq$
`$\varsupsetneq$`	$\varsupsetneq$	`$\nsubseteqq$`	$\nsubseteqq$	`$\nsupseteqq$`	$\nsupseteqq$
`$\subsetneqq$`	$\subsetneqq$	`$\supsetneqq$`	$\supsetneqq$	`$\varsubsetneqq$`	$\varsubsetneqq$
`$\varsupsetneqq$`	$\varsupsetneqq$	`$\subsetneq$`	$\subsetneq$	`$\supsetneq$`	$\supsetneq$
`$\nsubseteq$`	$\nsubseteq$	`$\nsupseteq$`	$\nsupseteq$	`$\nparallel$`	$\nparallel$
`$\nmid$`	$\nmid$	`$\nshortmid$`	$\nshortmid$	`$\nshortparallel$`	$\nshortparallel$
`$\nvdash$`	$\nvdash$	`$\nVdash$`	$\nVdash$	`$\nvDash$`	$\nvDash$
`$\nVDash$`	$\nVDash$	`$\ntrianglerighteq$`	$\ntrianglerighteq$	`$\ntrianglelefteq$`	$\ntrianglelefteq$
`$\ntriangleleft$`	$\ntriangleleft$	`$\ntriangleright$`	$\ntriangleright$	`$\nleftarrow$`	$\nleftarrow$
`$\nrightarrow$`	$\nrightarrow$	`$\nLeftarrow$`	$\nLeftarrow$	`$\nRightarrow$`	$\nRightarrow$
`$\nLeftrightarrow$`	$\nLeftrightarrow$	`$\nleftrightarrow$`	$\nleftrightarrow$	`$\divideontimes$`	$\divideontimes$
`$\varnothing$`	$\varnothing$	`$\nexists$`	$\nexists$	`$\Finv$`	$\Finv$
`$\Game$`	$\Game$	`$\mho$`	$\mho$	`$\eth$`	$\eth$
`$\eqsim$`	$\eqsim$	`$\beth$`	$\beth$	`$\gimel$`	$\gimel$
`$\daleth$`	$\daleth$	`$\lessdot$`	$\lessdot$	`$\gtrdot$`	$\gtrdot$
`$\ltimes$`	$\ltimes$	`$\rtimes$`	$\rtimes$	`$\shortmid$`	$\shortmid$
`$\shortparallel$`	$\shortparallel$	`$\smallsetminus$`	$\smallsetminus$	`$\thicksim$`	$\thicksim$
`$\thickapprox$`	$\thickapprox$	`$\approxeq$`	$\approxeq$	`$\succapprox$`	$\succapprox$
`$\precapprox$`	$\precapprox$	`$\curvearrowleft$`	$\curvearrowleft$	`$\curvearrowright$`	$\curvearrowright$
`$\digamma$`	$\digamma$	`$\varkappa$`	$\varkappa$	`$\Bbbk$`	$\Bbbk$
`$\hslash$`	$\hslash$	`$\hbar$`	$\hbar$	`$\backepsilon$`	$\backepsilon$

`$\shortleftarrow$`	$\shortleftarrow$	`$\shortrightarrow$`	$\shortrightarrow$	`$\shortuparrow$`	$\shortuparrow$
`$\shortdownarrow$`	$\shortdownarrow$	`$\Yup$`	$\Yup$	`$\Ydown$`	$\Ydown$
`$\Yleft$`	$\Yleft$	`$\Yright$`	$\Yright$	`$\varcurlyvee$`	$\varcurlyvee$
`$\varcurlywedge$`	$\varcurlywedge$	`$\minuso$`	$\minuso$	`$\baro$`	$\baro$
`$\sslash$`	$\sslash$	`$\bbslash$`	$\bbslash$	`$\moo$`	$\moo$
`$\varotimes$`	$\varotimes$	`$\varoast$`	$\varoast$	`$\varobar$`	$\varobar$
`$\varodot$`	$\varodot$	`$\varoslash$`	$\varoslash$	`$\varobslash$`	$\varobslash$
`$\varocircle$`	$\varocircle$	`$\varoplus$`	$\varoplus$	`$\varominus$`	$\varominus$
`$\boxast$`	$\boxast$	`$\boxbar$`	$\boxbar$	`$\boxdot$`	$\boxdot$
`$\boxslash$`	$\boxslash$	`$\boxbslash$`	$\boxbslash$	`$\boxcircle$`	$\boxcircle$
`$\boxbox$`	$\boxbox$	`$\boxempty$`	$\boxempty$	`$\lightning$`	$\lightning$
`$\merge$`	$\merge$	`$\vartimes$`	$\vartimes$	`$\fatsemi$`	$\fatsemi$
`$\sswarrow$`	$\sswarrow$	`$\ssearrow$`	$\ssearrow$	`$\curlywedgeuparrow$`	$\curlywedgeuparrow$
`$\curlywedgedownarrow$`	$\curlywedgedownarrow$	`$\fatslash$`	$\fatslash$	`$\fatbslash$`	$\fatbslash$
`$\lbag$`	$\lbag$	`$\rbag$`	$\rbag$	`$\varbigcirc$`	$\varbigcirc$

`$\leftrightarroweq$`	$\leftrightarroweq$	`$\curlyveedownarrow$`	$\curlyveedownarrow$	`$\curlyveeuparrow$`	$\curlyveeuparrow$
`$\nnwarrow$`	$\nnwarrow$	`$\nnearrow$`	$\nnearrow$	`$\leftslice$`	$\leftslice$
`$\rightslice$`	$\rightslice$	`$\varolessthan$`	$\varolessthan$	`$\varogreaterthan$`	$\varogreaterthan$
`$\varovee$`	$\varovee$	`$\varowedge$`	$\varowedge$	`$\talloblong$`	$\talloblong$
`$\interleave$`	$\interleave$	`$\obar$`	$\obar$	`$\obslash$`	$\obslash$
`$\olessthan$`	$\olessthan$	`$\ogreaterthan$`	$\ogreaterthan$	`$\ovee$`	$\ovee$
`$\owedge$`	$\owedge$	`$\oblong$`	$\oblong$	`$\inplus$`	$\inplus$
`$\niplus$`	$\niplus$	`$\nplus$`	$\nplus$	`$\subsetplus$`	$\subsetplus$
`$\supsetplus$`	$\supsetplus$	`$\subsetpluseq$`	$\subsetpluseq$	`$\supsetpluseq$`	$\supsetpluseq$
`$\Lbag$`	$\Lbag$	`$\Rbag$`	$\Rbag$	`$\llparenthesis$`	$\llparenthesis$
`$\rrparenthesis$`	$\rrparenthesis$	`$\binampersand$`	$\binampersand$	`$\bindnasrepma$`	$\bindnasrepma$
`$\trianglelefteqslant$`	$\trianglelefteqslant$	`$\trianglerighteqslant$`	$\trianglerighteqslant$	`$\ntrianglelefteqslant$`	$\ntrianglelefteqslant$
`$\ntrianglerighteqslant$`	$\ntrianglerighteqslant$	`$\llfloor$`	$\llfloor$	`$\rrfloor$`	$\rrfloor$
`$\llceil$`	$\llceil$	`$\rrceil$`	$\rrceil$	`$\arrownot$`	$\arrownot$
`$\Arrownot$`	$\Arrownot$	`$\Mapstochar$`	$\Mapstochar$	`$\mapsfromchar$`	$\mapsfromchar$
`$\Mapsfromchar$`	$\Mapsfromchar$	`$\leftrightarrowtriangle$`	$\leftrightarrowtriangle$	`$\leftarrowtriangle$`	$\leftarrowtriangle$
`$\rightarrowtriangle$`	$\rightarrowtriangle$	`$\bigtriangledown$`	$\bigtriangledown$	`$\bigtriangleup$`	$\bigtriangleup$
`$\bigcurlyvee$`	$\bigcurlyvee$	`$\bigcurlywedge$`	$\bigcurlywedge$	`$\bigsqcap$`	$\bigsqcap$
`$\bigbox$`	$\bigbox$	`$\bigparallel$`	$\bigparallel$	`$\biginterleave$`	$\biginterleave$
`$\bignplus$`	$\bignplus$	`$\llbracket$`	$\llbracket$	`$\rrbracket$`	$\rrbracket$
`$\varcopyright$i`	$\varcopyright$i	`$\longarrownot$`	$\longarrownot$	`$\Longarrownot$`	$\Longarrownot$
`$\Mapsto$`	$\Mapsto$	`$\mapsfrom$`	$\mapsfrom$	`$\Mapsfrom$`	$\Mapsfrom$
`$\Longmapsto$`	$\Longmapsto$	`$\longmapsfrom$`	$\longmapsfrom$	`$\Longmapsfrom$`	$\Longmapsfrom$

Il y a une version PostScript de ce catalogue sur CTAN[7]. Voir aussi qsymbols.ps[8].
L'extension `yhmath` définit des commandes `\widetilde`, `\widehat`, `\widetriangle`, `\wideparen` et `\widering`.

`$\wideringA\cap B$`	$\widering{A\cap B}$	`$\widetildeA+B$`	$\widetilde{A+B}$	`$\widehatA+B$`	$\widehat{A+B}$
`$\widetriangleA+B$`	$\widetriangle{A+B}$	`$\wideparenA+B$`	$\wideparen{A+B}$		

2.8 Comment avoir une fonte mathématique manuscrite?

Plusieures possibilités s'offrent à vous.

– La commande `\mathcal` (c'est probablement ce que vous avez essayé et qui ne vous satisfait pas).

Soit f une fonction de classe $\mathcal{C}^1$ et $\mathcal{L}f$ sa transformée de Laplace.

$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

– La commande `\mathscr` avec l'extension `euler`

Soit f une fonction de classe $\mathscr{C}^1$ et $\mathscr{L}$f sa transformée de Laplace.

$\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

– La commande `\mathscr` avec l'extension `mathrsfs` (c'est probablement cette réponse que voulez)

Soit f une fonction de classe $\mathscr{C}^1$ et $\mathscr{L}f$ sa transformée de Laplace.

$\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

Si vous voulez aussi des lettres minuscules, vous pouvez essayer de trouver une fonte (texte) manuscrite qui se marrie bien avec les fontes que vous utilisez, mais c'est plus compliqué à installer (voir 4.13). Voici un exemple avec la fonte `hershey` (qui

7. `ftp://ftp.lip6.fr/pub/TeX/CTAN/info/symbols.ps`
8. `ftp://ftp.lip6.fr/pub/TeX/CTAN/macros/latex/contrib/supported/qsymbols/qsymbols.ps.gz`

est beaucoup trop épaisse par rapport aux Computer Modern, dont les minuscules sont un peu trop petites et les majuscules trop grandes).

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{amsmath,amssymb}

\DeclareFontFamily{U}{manuscrit}{}
\DeclareFontShape{U}{manuscrit}{m}{n}
         { <5><6>
           <7><8><9>
           <10><10.95><12><14.4><17.28><20.74><24.88> hscs10
         }{}
\DeclareSymbolFont{manuscrit}{U}{manuscrit}{m}{n}
\DeclareSymbolFontAlphabet{\mathm}{manuscrit}
\newcommand{\textm}[1]
{\fontencoding{U}\fontfamily{manuscrit}\selectfont #1}

\begin{document}
Soit $\mathm g$ une fonction de classe $\mathm C^1$ et $\mathm{Lg}$
sa transformée de Laplace.
\end{document}
```

Soit $\mathcal{g}$ une fonction de classe $\mathcal{C}^1$ et $\mathcal{Lg}$ sa transformée de Laplace.

2.9 Comment avoir une fonte mathématique gothique ?

Pour les fontes gothiques, il suffit d'utiliser la commande `\mathfrak` avec l'extension `amssymb` ou `euler`. On remarquera qu'on peut avoir à la fois du gothique maigre et du gothique gras.

```
\documentclass[a4paper,12pt]{article}
\usepackage{bm,amsmath,amssymb}
\usepackage{french}
\def\cat#1{{\mathfrak{#1}}}
\def\dcat#1{{\bm{\mathfrak{#1}}}}
\begin{document}
Notons $\cat{Ens}$ la catégorie des ensembles et
$\dcat{Cat}$ la $2$-catégorie des catégories.
\end{document}
```

Notons $\mathfrak{Ens}$ la catégorie des ensembles et $\boldsymbol{\mathfrak{Cat}}$ la 2-catégorie des catégories.

Voici la liste des caractères. On s'apperçoit en particulier que le drôle de signe utilisé pour désigner le groupe symétrique n'est qu'un S majuscule gothique.

$\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$
$\mathfrak{abcdefghijklmnopqrstuvwxyz}$

Un problème demeure : il s'agit de fontes mathématiques, il n'y a donc pas d'accents. Que se passe-t-il si, par exemple, on veut appeler `Groupoïdes`, en gothique gras (et avec un tréma sur le i), la 2-catégorie des groupoïdes ? Une solution consiste à trouver une fonte gothique *textuelle*, à l'installer comme nous le verrons plus loin (4.6, 8.13) et à l'utiliser en mode mathématique à l'aide de la commande `\text` d'`amsmath`.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
```

```
\usepackage{amsmath,amsfonts,oldeng}
\def\cat#1{\text{\oldeng\mdseries #1}}
\def\dcat#1{\text{\oldeng\bfseries #1}}

\begin{document}
Les noms de catégorie seront écrits dans une fonte gothique (par
exemple $\cat{Ens}$ pour la catégorie des ensembles, $\cat{Sh}\,X$
pour la catégorie des faisceaux étales sur~$X$) et les $2$-catégories
par une fonte gothique grasse (par exemple $\dcat{Cat}$ pour la
$2$-catégorie des catégories ou $\dcat{Groupoïdes}$ pour celle des
groupoïdes).

Nous n'utiliserons pas de fonte gothique italique
(\textoldeng{\textsl{Ens}}) ou gothique italique grasse
(\textoldeng{\slshape\bfseries Groupoïdes}).
\end{document}
```

Les noms de catégorie seront écrits dans une fonte gothique (par exemple $\mathfrak{Ens}$ pour la catégorie des ensembles, $\mathfrak{Sh}\,X$ pour la catégorie des faisceaux étales sur X) et les 2-catégories par une fonte gothique grasse (par exemple **𝕮𝖆𝖙** pour la 2-catégorie des catégories ou **𝕲𝖗𝖔𝖚𝖕𝖔ï𝖉𝖊𝖘** pour celle des groupoïdes).

Nous n'utiliserons pas de fonte gothique italique (*𝔈𝔫𝔰*) ou gothique italique grasse (***𝕲𝖗𝖔𝖚𝖕𝖔ï𝖉𝖊𝖘***).

2.10 Comment avoir des lettres mathématiques avec une double barre?

On parle souvent de « gras du tableau noir » pour désigner ces lettres : c'était la manière de retranscrire au tableau la fonte grasse utilisée dans les livres.

- La manière la plus simple d'obtenir ces lettres est donc d'utiliser du gras, à l'aide de la commande `\mathbf`.

  Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ une fonction...

- On peut aussi utiliser du gras non étendu.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{french}

\DeclareSymbolFont{gras}{T1}{cmr}{b}{n}
\DeclareSymbolFontAlphabet{\mathbb}{gras}

\begin{document}
Soit $f : \mathbb R \longrightarrow \mathbb R$ une fonction...
\end{document}
```

  Soit $f : \mathbf{R} \longrightarrow \mathbf{R}$ une fonction...

- Les caractères qui ressemblent le plus à ceux que l'on trace au tableau, en France (par exemple, la double barre du N est sur le trait vertical de gauche et pas sur le trait en biais au milieu) et dont le gris est comparable à celui des fontes Computer

Modern sont celles de la fonte *doublestroke*. On les obtient à l'aide de la commande `\mathds` avec l'extension `dsfont`.

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction...
Nous noterons $\mathbb{1}$ l'application identité...
Soit $\mathbb{k}$ un corps de caractéristique nulle...

– Néanmoins, la fonte la plus utilisée est celle de l'extension `amssymb`, obtenue par la commande `\mathbb`.

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction...

– On peut aussi utiliser la commande `\mathbb` avec l'extension `bbold`.

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction...
Nous noterons $\mathbb{1}$ l'application identité...
Soit $\mathbb{k}$ un corps de caractéristique nulle...

– On peut aussi utiliser la commande `\mathbbm` avec l'extension `bbm`.

Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction...
Nous noterons $\mathbb{1}$ l'application identité...
Soit $\mathbb{k}$ un corps de caractéristique nulle...

– Ces caractères sont aussi présents dans `china2e` (très laid).

Soit $f : \mathsf{R} \longrightarrow \mathsf{R}$ une fonction...

Vous pouvez trouver un fichier comparant ces fontes sur CTAN[9].

2.11 Comment avoir des symboles mathématiques en gras?

La commande `\bm`, dans l'extension `bm`, met son argument en gras. (Cette commande remplace la commande `\boldsymbol` d'`amsmath`.)

```
\documentclass{article}
\usepackage{bm}
\begin{document}
\[  A              \bm{A}
    \alpha         \bm\alpha
    \infty         \bm\infty
    \oplus         \bm\oplus
    \bigoplus      \bm\bigoplus
    \mathfrak{Cat} \bm{\mathfrak{Cat}}
\]
\end{document}
```

$$A\boldsymbol{A}\alpha\boldsymbol{\alpha}\infty\boldsymbol{\infty}\oplus\boldsymbol{\oplus}\bigoplus\boldsymbol{\bigoplus}\mathfrak{Cat}\boldsymbol{\mathfrak{Cat}}$$

On peut aussi vouloir que *tout* ce qui est en mode mathématique soit en gras. La commande `\mathversion` (voir 4.16) permet de choisir entre plusieurs ensembles de fontes mathématiques : par défaut, il y juste le maigre (`normal`) et le gras (`bold`). C'est juste pertinent dans les titres. Il faut etre conscient que cela ne permet plus de faire la différence entre une quantité mathématique notée en maigre (par exemple, un réel) et une quantité mathématique notée par la même lettre, mais en gras (par exemple, un vecteur).

```
\documentclass{article}
\makeatletter
\renewcommand\section{\@startsection {section}{1}{\z@}%
                                     {-3.5ex \@plus -1ex \@minus -.2ex}%
                                     {2.3ex \@plus.2ex}%
```

9. `ftp://ftp.lip6.fr/pub/TeX/CTAN/info/blackboard/blackboard.ps`

```
                                    {\mathversion{bold}\normalfont\huge\bfseries}}
\makeatother
\begin{document}
\section{Élements de $K$-théorie}
La $K$-théorie...
\end{document}
```

1 Élements de $\boldsymbol{K}$-théorie

La K-théorie...

Une autre idée (moins bonne) serait d'utiliser la commande `\everymath` qui permet d'exécuter des commandes à chaque fois que l'on passe en mode mathématique. L'un des problème, c'est que la commande pour passer en gras en mode mathématique, `\mathversion{bold}`, ne s'utilise pas en mode mathématique. Une utilisation plus correcte de la commande `\everymath` consiste à changer la couleur du texte mathématique.

```
\documentclass{demo}
\usepackage{color,amsmath,amssymb}
\everymath{\color{red}}
\newcommand{\RR}{\mathbb{R}}
\begin{document}
Résoudre l'équation
$ x^2 - 3x + 2 = 0 $ dans $\RR$.
\end{document}
```

Résoudre l'équation $x^2 - 3x + 2 = 0$ dans $\mathbb{R}$.

Chapitre 3

Généralités

3.1 Vocabulaire

Bitmap (fonte) C'est une fonte dans laquelle les caractères sont décrits par des points, des pixels. À une taille assez basse, c'est très correct, mais si on tente de grossir les caractère, les pixels deviennent visibles.

En fait, TeX utilise des fontes Bitmap, mais si on demande une fonte à une certaine taille, il va recréer les fichiers bitmaps pour cette taille, si bien que l'on ne voit jamais les pixels.

Codage d'une fonte Une fonte est un tableau de caractères. Le codage est la donnée de l'emplacement des caractères dans le tableau. On ne confondra pas avec la notion de codage d'entrée, qui est le codage dans lequel le texte a été tapé (latin1 sur une machine Unix, applemac sur un Mac) et qui n'a rien à voir avec les fontes.

dpi Abréviation anglaise de « dots per inch », ie, « points par pouce ». C'est la résolution d'une fonte bitmap ou d'une imprimante. Ainsi, TeX a besoin de connaître la résolution de l'imprimante avant de transformer les fontes Métafont en fichiers bitmaps.

Glyphe Dessin d'un caractère

Gris Quand on regarde une page imprimée de loin, de manière à ne plus pouvoir distinguer les caractères individuels, la page semble grise. Si les caractères utilisés appartiennent tous à la même fonte et si cette fonte est homogène, ce gris semblera uniforme. La profondeur de ce gris s'appelle justement le gris de la fonte. Quand on mélange plusieures fontes (en particulier une fonte texte et une fonte mathématique), il faut prendre garde à ce que ces fontes aient le même gris, de manière que la page reste uniforme. Le gris dépend, entre autres, de la taille des caractères et de l'épaisseur des traits.

Jeu de caractères C'est un ensemble de caractères, chacun étant identifié par son nom (par exemple `LATIN LETTER A`), indépendemment de tout codage.

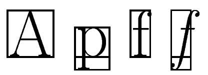
Kerning (Si quelqu'un connait la traduction française de ce mot, qu'il n'hésite pas à me la donner.) Les caractères ne sont pas juxtaposés les uns aux autres en ne tenant compte que de la taille des caractères : selon la forme des lettres, il peut être nécessaire de rapprocher certaines lettres. C'est le cas par exemple d'un A suivi d'un V.

AVAL

AVAL

Ligature Quand deux caractères sont proches, on choisit parfois de les dessiner un peu différemment : c'est par exemple le cas du `f` et du `i`, le point se mélangeant à la boucle. Les fontes True Type comportent souvent les ligatures `fi` et `fl`.

Métriques Une fonte ne contient pas que le dessin des caractères, mais aussi leur taille (hauteur, largeur, profondeur). On pourrait croire que cette information est supperflue, que l'on peut la retrouver en prenant juste la taille du dessin des caractères, mais il n'en est rien. Par exemple, la largeur d'un caractère italique est inférieure à celle du dessin de ce caractère.



Les métriques contiennent aussi d'autres informations, comme le « Kerning ».

Multimaster Les fontes Métapost utilisées par LaTeX changent selon la taille, de manière à être homogènes (si on se contentait de les grossir, les traits deviendraient plus épais ou plus fins, ce qui n'est pas souhaitable) et plus lisibles (les caractères de petite taille sont relativement plus larges) ; voir 3.13. C'est ce que l'on appelle en anglais la *metaness* d'une fonte : à partir d'un seul fichier on peut créer toute une famille de fontes, simplement en changeant certains paramètres.

Les fontes PostScript ne permettent pas vraiment ce genre de chose (avec des fontes de type 3, c'est toutefois possible). C'est là qu'interviennent les fontes multimaster[1] : c'est un nouveau standard qui permet de définir en une seule fois tout une famille de fontes, qui dépendra de plusieurs paramètres (par exemple la taille, la graisse, l'inclinaison, et pourquoi pas la taille des empatements).

Il semble que les fontes Multimaster aient été un échec commercial.

Outline C'est une fonte vectorielle dans laquelle les caractères sont définis par un contour : les caractères ne sont pas définis par des instructions « tracer une ligne, une courbe », mais simplement par « colorier l'intérieur d'une courbe fermée ». Les fontes Outline sont des fontes vectorielles, mais les fontes vectorielles ne sont pas toutes des fontes outline (c'est le cas des fontes MétaFont). Elles permettent divers effets graphiques : on peut ne tracer que le contour des caractères (sans colorier l'intérieur), on peut aussi utiliser la commande PostScript `clip` sur ce contour.

ABCabc

ABCabc

ABCabc

Sans Sérif Se dit d'une fonte dont les glyphes n'ont pas d'empatement. Dans la figure suivante, la lettre de droite est sans sérif, contrairement à celle de gauche.

AA

1. `http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/26-bouche.pdf`

On considère généralement que les fontes Sans Sérif sont très belles dans les titres, mais on constate qu'elles sont plus pénibles à lire.

Vectorielle (fonte) C'est une fonte dans laquelle les caractères sont décrits non pas par des pixels, mais par des instrucations graphiques (tracer une ligne, une courbe, colorier l'intérieur d'une courbe fermée, etc.) On peut donc afficher une telle fonte en n'importe quelle taille sans problème de pixellisation, contrairement aux fontes bitmap.

3.2 Quels sont tous ces fichiers ?

***.afm** Métriques d'une fonte PostScript (c'est un fichier texte) Voici un extrait d'un tel fichier : c'est compréhensible à peu près directement. Après l'en-tête, qui contient des paramètres relatifs à la fonte dans son ensemble (le codage, la « bounding box » (une boite dans laquelle on peut mettre n'importe quel glyphe de la fonte, utilisée lors de la rastérisation), l'angle des caractères italiques, la hauteur des majuscules, des minuscules, la profondeur à laquelle les caractères qui s'étendent en dessous de la ligne de base (tels que y, p, q, g ou j) s'enfoncent, etc.), viennent les caractèrtes eux-même : leur numéro, leur largeur, leur nom, leur « bounding box » (une boite dans laquelle on peut mettre le glyphe correspondant). On remarquera que certains caractères n'ont pas de numéro, mais juste un nom : ils sont absents du codage ; si on veut les utiliser, il faut donc de changer le codage de la fonte. Le fichier s'achève sur le kerning et les caractères obtenus en combinant plusieurs autres caractères (généralement, une lettre et un accent).

```
StartFontMetrics 2.0
Comment Copyright (c) 1985, 1987, 1989, 1990 Adobe Systems Incorporated.
                      All Rights Reserved.
Comment Creation Date: Tue Mar 20 12:15:44 1990
Comment UniqueID 28416
Comment VMusage 30487 37379
FontName Times-Roman
FullName Times Roman
FamilyName Times
Weight Roman
ItalicAngle 0
IsFixedPitch false
FontBBox -168 -218 1000 898
UnderlinePosition -100
UnderlineThickness 50
Version 001.007
Notice Copyright (c) 1985, 1987, 1989, 1990 Adobe Systems Incorporated.  All Rights Reserved.Times is a tra
e AG and/or its subsidiaries.
EncodingScheme AdobeStandardEncoding
CapHeight 662
XHeight 450
Ascender 683
Descender -217
StartCharMetrics 228
C 32 ; WX 250 ; N space ; B 0 0 0 0 ;
C 33 ; WX 333 ; N exclam ; B 130 -9 238 676 ;
C 34 ; WX 408 ; N quotedbl ; B 77 431 331 676 ;
C 35 ; WX 500 ; N numbersign ; B 5 0 496 662 ;
C 36 ; WX 500 ; N dollar ; B 44 -87 457 727 ;

C 65 ; WX 722 ; N A ; B 15 0 706 674 ;
C 66 ; WX 667 ; N B ; B 17 0 593 662 ;
C 67 ; WX 667 ; N C ; B 28 -14 633 676 ;
C 68 ; WX 722 ; N D ; B 16 0 685 662 ;

C -1 ; WX 611 ; N Zcaron ; B 9 0 597 886 ;
C -1 ; WX 444 ; N ccedilla ; B 25 -215 412 460 ;
C -1 ; WX 500 ; N ydieresis ; B 14 -218 475 623 ;
EndCharMetrics
StartKernData
```

```
StartKernPairs 283
KPX A y -92
KPX A w -92
KPX A v -74
KPX A u 0
KPX A quoteright -111
KPX A quotedblright 0
KPX A p 0

KPX z o 0
KPX z e 0
EndKernPairs
EndKernData

StartComposites 58
CC Aacute 2 ; PCC A 0 0 ; PCC acute 195 212 ;
CC Acircumflex 2 ; PCC A 0 0 ; PCC circumflex 195 212 ;
CC Adieresis 2 ; PCC A 0 0 ; PCC dieresis 195 212 ;

EndComposites
EndFontMetrics
```

***.def** Ce sont des fichiers décrivant des codages d'entrée (pas le codage d'une fonte, mais le codage d'un texte) : `ascii`, `latin1`, `applemac`, `ansinew`, `cp850`, etc.

Un tel fichier contient des lignes du genre

```
\DeclareInputText{191}{\textquestiondown}
\DeclareInputText{192}{\@tabacckludge`A}
\DeclareInputText{208}{\DH}
\DeclareInputText{193}{\@tabacckludge'A}
\DeclareInputText{209}{\~N}
\DeclareInputText{194}{\^A}
\DeclareInputText{210}{\@tabacckludge`O}
\DeclareInputText{195}{\~A}
```

Par exemple, la dernière ligne signifie que le caractère de code 195 est un A tilde (Ã). Cela permet donc de taper « Ã » au lieu de « `\~A` » : c'est beaucoup plus lisible.

Ces fichiers sont lus par LaTeX, par exemple quand on lui dit

```
\usepackage[latin1]{inputenc}
```

Mais on peut aussi vouloir utiliser simultanément plusieurs codage d'entrée : on rencontre ce problème si on doit compiler ensemble (comme différentes parties d'un même document) des fichiers de provenance diverses (machine Unix, PC sous Ouindaube, Mac sous MacOS). Il suffit de charger `inputenc` avec toutes les options voulues et on peut dès lors passer d'un codage à l'autre à l'aide de la commande `\inputencoding` Par exemple, pour passer au codage `latin1`, on dira `\inputencoding{latin1}`.

***enc.def** Codage d'une fonte (i.e., correspondance entre le nom des caractères et leur position dans la fonte), sous une forme compréhensible par LaTeX. Voici un extrait du fichier `t1enc.def`.

```
\ProvidesFile{t1enc.def}
\DeclareFontEncoding{T1}{}{}

\DeclareTextAccent{\`}{T1}{0}
\DeclareTextAccent{\'}{T1}{1}
\DeclareTextAccent{\^}{T1}{2}
\DeclareTextAccent{\~}{T1}{3}
\DeclareTextAccent{\"}{T1}{4}
```

```
\DeclareTextCommand{\b}{T1}[1]
   {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{29}%
     \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}}}
\DeclareTextCommand{\c}{T1}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
     \else{\ooalign{\hidewidth\char11\hidewidth
        \crcr\unhbox\z@}}\fi}
\DeclareTextCommand{\d}{T1}[1]
   {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{10}.\hidewidth}}}
\DeclareTextCommand{\k}{T1}[1]
   {\oalign{\null#1\crcr\hidewidth\char12}}
\DeclareTextCommand{\textperthousand}{T1}
   {\%\char 24 }          % space or `relax as delimiter?
\DeclareTextCommand{\textpertenthousand}{T1}
   {\%\char 24\char 24 }  % space or `relax as delimiter?

\DeclareTextSymbol{\guillemotleft}{T1}{19}
\DeclareTextSymbol{\guillemotright}{T1}{20}
\DeclareTextSymbol{\AE}{T1}{198}
\DeclareTextSymbol{\OE}{T1}{215}
\DeclareTextSymbol{\o}{T1}{248}

\DeclareTextSymbol{\textunderscore}{T1}{95}
\DeclareTextSymbol{\textvisiblespace}{T1}{32}

\DeclareTextComposite{\c}{T1}{C}{199}
\DeclareTextComposite{\c}{T1}{c}{231}
```

Ce fichier est structuré ainsi : on commence par dire à quel endroit de la fonte (vue comme un tableau de glyphes, indexé par des nombres) se trouvent les accents ; viennent ensuite des commandes pour créer les lettrres accentuées si elles ne sont pas déjà présentes dans le fonte ; puis quelques symboles ; puis les lettres accentuées (ici, Ç et ç) qui sont présentes dans la fonte.

***.enc** Fichier décrivant le codage d'une fonte (utilisé par `dvips`). Il s'agit d'un fichier PostScript qui contient la liste des caractères dans une fonte, dans l'ordre. Voici un extrait du fichier `cork.enc`.

```
/CorkEncoding [          % now 256 chars follow
% 0x00
  /grave /acute /circumflex /tilde /dieresis /hungarumlaut /ring /caron
  /breve /macron /dotaccent /cedilla
  /ogonek /quotesinglbase /guilsinglleft /guilsinglright
% 0x10
  /quotedblleft /quotedblright /quotedblbase /guillemotleft
  /guillemotright /endash /emdash /compworkmark
  /perthousandzero /dotlessi /dotlessj /ff /fi /fl /ffi /ffl
% 0x20
  /visualspace /exclam /quotedbl /numbersign
  /dollar /percent /ampersand /quoteright
  /parenleft /parenright /asterisk /plus /comma /hyphen /period /slash
...
] def
```

On utilise la variable ainsi créée (ici, `CorkEncoding`) par exemple dans le fichier `psfonts.map` ou `ttfonts.map`.

***.etx** Fichier décrivant un codage (utilisé par `fontinst`, voir 9.3). On trouve essentiellement deux types de fichiers. Tout d'abord, les normaux, par exemple, `OT1.etx`, dont voici un extrait.

```
\relax
\encoding
\needsfontinstversion{1.6}
\comment{\section{Default values}}
```

```
\setstr{codingscheme}{EXTENDED TEX FONT ENCODING - LATIN}

\setcommand\lc#1#2{#2}
\setcommand\uc#1#2{#1}
\setcommand\lctop#1#2{#2}
\setcommand\uctop#1#2{#1}
\setcommand\lclig#1#2{#2}
\setcommand\uclig#1#2{#1}
\setcommand\digit#1{#1}

\setint{italicslant}{0}

\ifisglyph{x}\then
   \setint{xheight}{\height{x}}
\else
   \setint{xheight}{500}
\fi

\ifisglyph{space}\then
   \setint{interword}{\width{space}}
\else\ifisglyph{i}\then
   \setint{interword}{\width{i}}
\else
   \setint{interword}{333}
\fi\fi

[...]

\setint{fontdimen(1)}{\int{italicslant}}              % italic slant
\setint{fontdimen(2)}{\int{interword}}                % interword space
\ifisint{monowidth}\then
   \setint{fontdimen(3)}{0}                           % interword stretch
   \setint{fontdimen(4)}{0}                           % interword shrink
\else
   \setint{fontdimen(3)}{\scale{\int{interword}}{600}}% interword stretch
   \setint{fontdimen(4)}{\scale{\int{interword}}{240}}% interword shrink
\fi
\setint{fontdimen(5)}{\int{xheight}}                  % x-height
\setint{fontdimen(6)}{1000}                           % quad
\ifisint{monowidth}\then
   \setint{fontdimen(7)}{\int{interword}}             % extra space after .
\else
   \setint{fontdimen(7)}{\scale{\int{interword}}{240}}% extra space after .
\fi
\setint{fontdimen(8)}{\int{capheight}}   % cap height
\setint{fontdimen(9)}{\int{ascender}}  % ascender
\setint{fontdimen(10)}{\int{acccapheight}} % accented cap height
\setint{fontdimen(11)}{\int{descender}} % descender's depth
\setint{fontdimen(12)}{\int{maxheight}} % max height
\setint{fontdimen(13)}{\int{maxdepth}} % max depth
\setint{fontdimen(14)}{\int{digitwidth}} % digit width
\setint{fontdimen(15)}{\int{capstem}} % cap_stem
\setint{fontdimen(16)}{\int{baselineskip}} % baselineskip

[...]

\setslot{\lc{Grave}{grave}}
   \comment{The grave accent `\`a'.}
```

```
\endsetslot

\setslot{\lc{Acute}{acute}}
   \comment{The acute accent `\'a'.}
\endsetslot

\setslot{\lc{Circumflex}{circumflex}}
   \comment{The circumflex accent `\^ a'.}
\endsetslot

\setslot{\lc{Tilde}{tilde}}
   \comment{The tilde accent `\~a'.}
\endsetslot

\setslot{\lc{Dieresis}{dieresis}}
   \comment{The umlaut or dieresis accent `\"a'.}
\endsetslot

[...]

\setslot{equal}
   \comment{The equals sign `='.}
\endsetslot

\setslot{greater}
   \ligature{LIG}{greater}{guillemotright}
   \comment{The greater-than sign `$>$'.}
\endsetslot

\setslot{question}
   \ligature{LIG}{quoteleft}{questiondown}
   \comment{The question mark `?'.}
\endsetslot

\setslot{at}
   \comment{The at sign `@'.}
\endsetslot

\setslot{\uc{A}{a}}
   \comment{The letter `{A}'.}
\endsetslot

\setslot{\uc{B}{b}}
   \comment{The letter `{B}'.}
\endsetslot

\setslot{\uc{C}{c}}
   \comment{The letter `{C}'.}
\endsetslot

[...]

\setslot{\lctop{Yacute}{yacute}}
   \comment{The letter `\' y'.}
\endsetslot

\setslot{\lc{Thorn}{thorn}}
   \comment{The Icelandic lowercase letter thorn, similar to a `p' with an
      ascender rising from the stem.  It is unavailable in \plain\ \TeX.}
```

```
\endsetslot

\setslot{\lc{SS}{germandbls}}
   \comment{The letter `\ss'.}
\endsetslot

\endencoding
```

Le fichier commence par définir les commandes `\lc`, `\uc`, `\lctop`, `\uctop`, `\lclig`, `\uclig`, `\digit`, qui seront utilisées lors de la définition du codage. Pour modifier l'apparence de la fonte (avoir des vieux chiffres, des petites capitales) il suffit de définir préalablement ces commandes : nous y reviendrons.
Le fichier définit ensuite les dimensions suivantes : `xheight`, `interword`, `capheight`, `ascender`, `acccapheight`, `descender`, `maxheight`, `maxdepth`, `digitwidth`, `capstem`, `baselineskip`.
Ensuite, il définit les dimensions que LaTeX connaîtra (`\fontdimen`). Enfin, il définit les caractères présents dans la fonte.
Les autres fichiers correspondent à des *modifications* de fichiers `*.etx` « normaux ». Extérieurement, le codage sera le même qu'avec les fichiers normaux, mais les caractères seront différents. Par exemple, voici le fichier `OT19.etx`, qui remplace les chiffres par des « vieux chiffres ». Cela suppose que ces vieux chiffres sont présents, avec les noms `oneoldstyle`, `twooldstyle`, etc., commne dans les fontes PostScript « expert ».

```
\relax
\encoding
% We define the parameters for ot1.etx to produce a font with old
% style digits.
\setcommand\lc#1#2{#2}
\setcommand\uc#1#2{#1}
\setcommand\lclig#1#2{#2}
\setcommand\uclig#1#2{#1}
\setcommand\digit#1{#1oldstyle}
\setcommand\currency{dollar}
% Then we call ot1.etx.
\inputetx{OT1}
\endencoding
```

Voici un autre exemple, les petites capitales, `OT1c.etx` : ici encore, on suppose que les petites capitales sont présentes dans la fontes, sous les noms `Asmall`, `Bsmall`, etc.

```
\relax
\encoding

\setcommand\lc#1#2{#1small}
\setcommand\uc#1#2{#1}
\setcommand\lclig#1#2{#1small}
\setcommand\uclig#1#2{#1}
\setcommand\digit#1{#1}
\setcommand\currency{dollar}

% The default letter spacing is 1/10 of the design size, with small
% caps 80% the size of large caps.

\setint{smallcapsscale}{800}

\ifisglyph{space}\then
   \setint{interword}{\width{space}}
\else\ifisglyph{i}\then
   \setint{interword}{\width{i}}
\else
   \setint{interword}{333}
\fi\fi
```

```
\ifisint{monoweight}\then
   \setint{smallcapsextraspace}
      {\div{\scale{\int{interword}}{\sub{1000}{\int{smallcapsscale}}}}{2}}
   \setint{smallcapsspacing}{0}
\else
   \setint{smallcapsextraspace}{0}
   \setint{smallcapsspacing}{50}
\fi

\ifisint{capspacing}\then
   \setint{letterspacing}{\int{capspacing}}
\else
   \setint{letterspacing}{\int{smallcapsspacing}}
\fi

\inputetx{OT1}
\endencoding
```

En résumé, les fichiers `*.etx` permettent de définir un codage en précisant où (dans la fonte initiale, vue juste comme une correspondance entre les noms de glyphes et les glyphes) il faut aller chercher les caractères. Il manque donc la connaissance de la correspondance entre le nom des glyphes et leur emplacement dans la fonte : c'est le rôle des fichiers `*enc.def` (par exemple `t1enc.def`).

***.fd** Le nom de ces fichiers est constitué par : le nom du codage (U, T1, OT1, etc.), en minuscules, suivi du nom d'une famille de fontes. Il est chargé automatiquement par LaTeX quand on cherche à utiliser une fonte de cette famille. Il contient la correspondance entre le nom d'une fonte (par exemple « Computer Modern, droit, gras étendu, 10pt ») et le nom du fichier la contenant. Il peut aussi préciser ce qu'il faut faire si une fonte n'existe pas (par exemple : « s'il n'y a pas de gras étendu, alors prendre du gras normal » — mais la plupart de ces substitutions sont automatiques). Voici un exemple de tel fichier.

```
\ProvidesFile{ot1cmr.fd}
\DeclareFontFamily{OT1}{cmr}{\hyphenchar\font45 }
\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}{}
\DeclareFontShape{OT1}{cmr}{m}{sl}{%
      <5><6><7>cmsl8%
      <8><9>gen*cmsl%
      <10><10.95>cmsl10%
      <12><14.4><17.28><20.74><24.88>cmsl12%
      }{}
\DeclareFontShape{OT1}{cmr}{m}{it}{%
      <5><6><7>cmti7%
      <8>cmti8%
      <9>cmti9%
      <10><10.95>cmti10%
      <12><14.4><17.28><20.74><24.88>cmti12%
      }{}
\DeclareFontShape{OT1}{cmr}{m}{sc}{%
      <5><6><7><8><9><10><10.95><12>%
      <14.4><17.28><20.74><24.88>cmcsc10%
      }{}
\DeclareFontShape{OT1}{cmr}{bx}{n}{%
      <5><6><7><8><9>gen*cmbx%
      <10><10.95>cmbx10%
      <12><14.4><17.28><20.74><24.88>cmbx12%
      }{}
\DeclareFontShape{OT1}{cmr}{bx}{sl}{%
```

```
      <5><6><7><8><9>%
      <10><10.95><12><14.4><17.28><20.74><24.88>cmbxsl10%
      }{}
\DeclareFontShape{OT1}{cmr}{bx}{it}{%
      <5><6><7><8><9>%
      <10><10.95><12><14.4><17.28><20.74><24.88>cmbxti10%
      }{}
\endinput
```

La commande `\DeclareFontFamily` définit une nouvelle famille (ensemble de fontes) : son premier argument est le codage, le second, le nom de la fonte. La commande `\DeclareFontShape` définit une fonte dans une famille : ses arguments sont le codage, le nom de la famille, la série (maigre, gras, etc.), la forme (droit, italique, penché, petites capitales) et la liste des tailles autorisées et les fichiers correspondants. Le dernier argument est généralement vide (voir 4.7). Nous reverrons tout cela en détails en 4.8.

***.gf** Fonte au format bitmap (non compressé) provenant d'un fichier Métafont et calculée pour une certaine imprimante (plus la résolution de votre imprimante est élevée, plus il faudra utiliser une résolution élevée). On peut en rendre le contenu compréhensible à l'aide de la commande `gftype` et le visualiser à l'aide de la commande `gftodvi`.

***.hbf** Fonte Bitmap pour les langues asiatiques, que l'on peut convertir en fichiers *.pk pour les utiliser sous LaTeX, par exemple à l'aide de CJK (voir 5.6).

***.mf** Fichier source d'une fonte (vectorielle) au format Métafont à partir duquel on peut calculer les métriques et créer les fichiers bitmaps utilisés par `xdvi` ou `dvips`. Voir le chapitre 6 sur Métafont. Voici un exemple d'un tel fichier. Ça n'a pas l'air très lisible comme cela, mais c'est quand-même très puissant. Je profite de l'occation pour signaler que Métafont a donné naissance à un logiciel de dessin, Métapost [2].

```
cmchar "Lowercase Greek beta (variant)";
beginchar("b",9u#,asc_height#,0);
italcorr .5[x_height#,asc_height#]*slant-u#;
adjust_fit(0,0);

pickup pencircle scaled 0.15pt;

pos1(hair,160);
pos2(vair,90);
pos3(stem,0);
pos4(vair,-90);
pos5(0.5u,-180);
pos6(0.5u,-270+45);
pos7(0.8curve,-360);
pos8(vair,-450-30);
pos9(hair,-520);

x1=x9=x5+1u;
x4=x2=x8=x6=0.5w;
lft x5l = hround(1.5u-0.5hair); % l ou r ?
rt x3l = hround(w-2.5u);
rt x7r = hround(w-2.5u+.5curve);

top y6l=h+oo;
y8=x_height+u;
y9=y8+u;
y1=y2-u;
y7=1/2[y6,y9];
y3=1/2[y1,y4];
y2=1/2[y8,y4];
y4l=-oo;
y5=1/2[y2,y8];
```

2. `http://cm.bell-labs.com/who/hobby/MetaPost.html`

```
%filldraw stroke z1..z2..z3..z4..{up}z5..z6..z7..z8..z9;
filldraw z1---z2r..z3r..z4r..z5r..z6r..z7r..z8r---z9---z8l..z7l..z6l..z5l..z4l..
z3l..z2l---cycle ;

penlabels(1,2,3,4,5,6,7,8,9); endchar;

cmchar "Lowercase Greek beta";
beginchar(oct"014",9.5u#,asc_height#,desc_depth#);
italcorr .5[x_height#,asc_height#]*slant-u#;
adjust_fit(0,0); pickup fine.nib;
pos0(hair,180); pos1(hair,180); pos2(vair,90);
pos3(stem,0); pos4(vair,-90); pos5(hair,-180);
pos6(vair,-270); pos7(curve,-360); pos8(vair,-450); pos9(hair,-540);
x0=x1=x9; lft x0l=hround(1.5u-.5hair); x2=x4=x6=x8=.5w+.25u;
rt x3r=hround(w-1.5u); rt x7r=hround(w-1.5u+.5curve); rt x5l=hround(x4-u);
bot y0=-d; y1=top y6r=x_height; top y2r=h+oo; y3=.5[y2,y4];
y5=.5[y4,y6]; top y6r-bot y4r=vstem+eps; bot y8=-oo; y7=y9=.5[y6,y8];
if y6l < y4l: y6l:=y4l:=y5; fi
filldraw stroke z0e---z1e...pulled_arc.e(2,3)
 & pulled_arc.e(3,4)...{up}z5e;  % stem and upper bowl
filldraw stroke z5e{up}...pulled_arc.e(6,7)
 & pulled_arc.e(7,8)...{up}z9e;  % lower bowl
penlabels(0,1,2,3,4,5,6,7,8,9); endchar;
```

***.mtx** Fichier utilisé par `fontinst` (9.3) décrivant une fonte (virtuelle ou non) ou des opérations pour en créer une nouvelle. Le premier cas est celui de la description d'une fonte : ce fichier est créé automatiquement par `fontinst` à partir du fichier `*.afm` ou `*.pl` correspondant.

Voici un extrait d'un tel fichier.

```
%Filename: ecrm1000.mtx
%Created by: tex essai
%Created using: \pltomtx{ecrm1000}{ecrm1000}

%This file contains the information of ecrm1000.pl in a form
%more easily read by TeX. It is used by the fontinst package.

%THIS FILE CAN BE DELETED.

\relax
\metrics

\needsfontinstversion{1.6}

\setdim{designsize}{10.0pt}
\setint{italicslant}{0000}
\setint{interword}{0333}
\setint{stretchword}{0166}
\setint{shrinkword}{0111}
\setint{xheight}{0430}
\setint{quad}{0999}
\setint{extraspace}{0111}

\setkern{f}{quoteright}{0077}
\setkern{R}{U}{-0027}
\setkern{A}{Y}{-0083}
\setkern{L}{W}{-0111}
\setkern{Lslash}{W}{-0111}

\setrawglyph{grave}{ecrm1000}{10.0pt}{0}{499}{641}{0}{0}
```

```
\setrawglyph{acute}{ecrm1000}{10.0pt}{1}{499}{641}{0}{0}
\setrawglyph{circumflex}{ecrm1000}{10.0pt}{2}{499}{641}{0}{0}
\setrawglyph{tilde}{ecrm1000}{10.0pt}{3}{499}{641}{0}{0}
\setrawglyph{dieresis}{ecrm1000}{10.0pt}{4}{499}{663}{0}{0}
\setrawglyph{ff}{ecrm1000}{10.0pt}{27}{583}{688}{0}{77}
\setrawglyph{fi}{ecrm1000}{10.0pt}{28}{555}{688}{0}{0}
\setrawglyph{fl}{ecrm1000}{10.0pt}{29}{555}{688}{0}{0}
\setrawglyph{ffi}{ecrm1000}{10.0pt}{30}{833}{688}{0}{0}
\setrawglyph{ffl}{ecrm1000}{10.0pt}{31}{833}{688}{0}{0}
\setrawglyph{visiblespace}{ecrm1000}{10.0pt}{32}{499}{430}{97}{0}
\setrawglyph{A}{ecrm1000}{10.0pt}{65}{749}{688}{0}{0}
\setrawglyph{B}{ecrm1000}{10.0pt}{66}{708}{688}{0}{0}
\setrawglyph{C}{ecrm1000}{10.0pt}{67}{722}{688}{0}{0}
\setrawglyph{D}{ecrm1000}{10.0pt}{68}{763}{688}{0}{0}

\endmetrics
```

Pour ceux qui veulent savoir, la commande `\setdim` définit les dimensions globales de la fonte, la commande `setkern` définit les informations de kerning et la commande `setrawglyph` définit chaque caractère (ses arguments sont : le nom du caractère, le nom de la fonte, la taille de la fonte, l'emplacement du caractère dans la fonte, la largeur, la hauteur, la profondeur et la correction italique).

L'autre type de fichier `*.mtx` ne contient pas la description d'une fonte entière mais juste des modifications à effectuer sur des caractères, par exemple intervertir des caractères, les remplacer par d'autres d'une autre fonte, etc. Ce sont les fichiers avec lesquels on programme sous `fontinst`.

Voici par exemple un extrait du fichier `uline.mtx`. Comme nous reviendrons très longuement sur ces fichiers dans la partie sur `fontinst` (9.3), nous ne donnons pas plus d'explications ici.

```
\relax

This files underlines all the glyphs mentioned in either of the files
OT1.etx or T1.etx.

\metrics

\setcommand\uline#1{%   underlining
  \ifisglyph{#1}\then
    \resetglyph{#1}
      \push\moveup{-150}\glyphrule{\width{#1}}{40}\pop
      \glyph{#1}{1000}
    \endresetglyph
  \fi
}

\uline{AE}
\uline{Aacute}
\uline{Abreve}
\uline{Acircumflex}
\uline{A}
\uline{B}
\uline{C}

[...]

\uline{a}
\uline{b}
\uline{c}
\uline{d}

[...]
```

```
\endmetrics
```

***.pdf** Fichier PDF[3] ; le PDF est le format qui va peut-être bientôt remplacer le PostScript.

***.pfa** Fonte PostScript au format ASCII. C'est un programme en PostScript, presque humainement lisible. Voici un extrait d'un tel fichier

```
%!PS-AdobeFont-1.0: Brush-Script-Italic 001.000
%%CreationDate: Friday, 10-Feb-95 12:29:24
% Generated with TypeSmith Amiga v2.5a
% Copyright © 1991-1994 Relog, Inc.
% Registered to Maurizio LORETI, U.of Padova, Dept.of Physics
%%EndComments
12 dict begin
/FontInfo 9 dict dup begin
/version (001.000) readonly def
/Notice () readonly def
/FullName (Brush Script Italic) readonly def
/FamilyName (Brush Script) readonly def
/Weight (normal) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -102 def
/UnderlineThickness 102 def
end readonly def
/FontName /pbsi0 def
/PaintType 0 def
/StrokeWidth 0 def
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0] readonly def
/FontBBox {-160 -292 1151 886} readonly def
/UniqueID 9357 def
/Encoding StandardEncoding def
currentdict end
currentfile eexec
D9D66F633B846A97B686A97E45A3D0AA052C9E907651DA770357949E564892BC5F9CE43497449D23
17C14FE8F0526C0681C4377CD16AE44D8579E73929B856A43EAAB519BA84DB1570EC80712CF30FF7
F7F6641333CC5B1D5B28DD0EB382D63419770B92983B0C388B3D35BB415FF85CF6F94EC96C52307F

[...]

D619609B31808B6348A68E83C34AECDDA1BBFBC911D2FD1D9F0DF9C597B83698EF6252D0D6835217
D15606F84F4956CB99BC9AA70B13AC26ED50AA9B65F8E93B621FF67A5D78CAB3DCC2F0F8216AD102
FC6CBB1BE72260
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
cleartomark
```

On peut le rendre lisible à l'aide de la commande `t1disasm` (la commande `t1asm` effectue l'opération inverse). Voici ce que l'on obtient.

```
%!PS-AdobeFont-1.0: Brush-Script-Italic 001.000
```

3. http://www.pdfzone.com/resources/technicalinfo.html

```
%%CreationDate: Friday, 10-Feb-95 12:29:24
% Generated with TypeSmith Amiga v2.5a
% Copyright © 1991-1994 Relog, Inc.
% Registered to Maurizio LORETI, U.of Padova, Dept.of Physics
%%EndComments
12 dict begin
/FontInfo 9 dict dup begin
/version (001.000) readonly def
/Notice () readonly def
/FullName (Brush Script Italic) readonly def
/FamilyName (Brush Script) readonly def
/Weight (normal) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -102 def
/UnderlineThickness 102 def
end readonly def
/FontName /pbsi0 def
/PaintType 0 def
/StrokeWidth 0 def
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0] readonly def
/FontBBox {-160 -292 1151 886} readonly def
/UniqueID 9357 def
/Encoding StandardEncoding def
currentdict end
currentfile eexec
dup/Private 10 dict dup begin
/RD{string currentfile exch readstring pop}executeonly def
/ND{noaccess def}executeonly def
/NP{noaccess put}executeonly def
/BlueValues [-5 0 373 373 676 676 610 610]def
/OtherBlues [-248 -205]def
/ForceBold true def
/MinFeature {16 16} def
/password 5839 def
/Subrs 260 array
dup 0 {
        3 0 callothersubr
        pop
        pop
        setcurrentpoint
        return
        } NP
dup 1 {
        0 1 callothersubr
        return
        } NP
dup 2 {
        0 2 callothersubr
        return
        } NP
dup 3 {
        return
        } NP
dup 4 {
        return
        } NP
dup 5 {
```

```
        -61 0 rlineto
        -48 -43 -14 -42 0 -32 rrcurveto
        0 -31 21 -22 28 0 rrcurveto
        39 0 53 27 46 45 rrcurveto
        0 16 1 15 4 16 rrcurveto
        -41 -34 -35 -20 -22 0 rrcurveto
        -14 0 -10 12 0 14 rrcurveto
        0 27 13 19 40 33 rrcurveto
        closepath
        return
        } NP
dup 6 {
        17 -17 34 7 26 26 rrcurveto
        26 26 8 35 -17 17 rrcurveto
        -17 17 -36 -7 -26 -26 rrcurveto
        -26 -26 -6 -35 17 -17 rrcurveto
        closepath
        return
        } NP
dup 7 {
        -21 -23 -35 -16 -30 0 rrcurveto
        -36 0 -32 23 -21 40 rrcurveto
        -2 4 0 1 -2 0 rrcurveto
        -3 0 -11 -8 -10 -8 rrcurveto
        -7 -5 -5 -9 0 -7 rrcurveto
        0 -43 68 -50 50 0 rrcurveto
        88 0 51 78 0 41 rrcurveto
        0 12 -4 2 -5 0 rrcurveto
        -1 0 -3 -1 -2 -3 rrcurveto
        -27 -28 rlineto
        closepath
        return
        } NP
```

***.pfb** Fonte postScript au format binaire (c'est comme une fonte postScript au format ASCII `*.pfa`, mais un peu compressé). C'est un fichier binaire, qui n'est donc plus lisible. On peut convertir une fonte `*.pfa` en `*.pfb` à l'aide de la commande `t1binary` et réciproquement à l'aide de `t1ascii` ou `pfb2pfa`.

***.pfm** Métriques d'une fonte PostScript pour Windaube, contenant (je crois) un peu moins d'informations que les fichiers `*.afm`. Il n'est pas humainement lisible.

***.pk** Fonte au format bitmap (compressé, contrairement aux `*.gf`) provenant souvent d'un fichier Métafont (mais parfois aussi d'une fonte PostScript (pour pouvoir l'utiliser avec `xdvi`) ou TTF). La résolution est indiquée juste avant, par exemple `cmr17.518pk` pour la fonte `cmr17` à 518dpi. On peut passer du format `*.pk` au format `*.gf` à l'aide de `pktogf` et réciproquement à l'aide de `gftopk`. La commande `pktype` convertit un fichier `*.pk` en un fichier humainement lisible.

***.pl** Fichier « property list » : c'est la version humainement lisible d'un fichier de métriques `*.tfm`, obtenue à l'aide de `tftopl`. On l'utilise comme point de départ quand on joue avec des fontes virtuelles. Le contenu d'un tel fichier ressemble à du Lisp : voici un extrait d'un tel fichier.

```
(FAMILY HELVETICA)
(FACE F MRR)
(CODINGSCHEME ADOBESTANDARDENCODING)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 37370204563)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.278)
   (STRETCH R 0.3)
   (SHRINK R 0.1)
```

```
   (XHEIGHT R 0.5)
   (QUAD R 1.0)
   )
(LIGTABLE
   (LABEL C f)
   (LIG C i O 256)
   (LIG C l O 257)
   (STOP)
   (LABEL O 140)
   (LIG O 140 O 252)
   (STOP)
   (LABEL O 47)
   (LIG O 47 O 272)
   (STOP)
   (LABEL O 55)
   (LIG O 55 O 261)
   (STOP)
   (LABEL O 261)
   (LIG O 55 O 320)
   (STOP)
   (LABEL O 41)
   (LIG O 140 O 241)
   (STOP)
   (LABEL O 77)
   (LIG O 140 O 277)
   (STOP)
   )
(CHARACTER O 40
   (CHARWD R 0.278)
   )
(CHARACTER O 41
   (CHARWD R 0.278)
   (CHARHT R 0.712)
   (COMMENT
      (LIG O 140 O 241)
      )
   )
(CHARACTER O 42
   (CHARWD R 0.355)
   (CHARHT R 0.712)
   )
```

Le fichier est directement compréhensible. Notons que la table de ligatures est présente deux fois : d'une part, au début du fichier, d'autre part, sous forme de commentaires (ignorés par le programme : c'est juste pour le lecteur humain), lors de la description de chaque caractère.

***.pro** Fichier prologue pour `dvips` : ce sont des commandes PostScript à mettre au début d'un fichier pour faire certaines initialisations (par exemple, définir des commandes). Voici un extrait d'un tel fichier.

```
%!
% PostScript prologue for pstricks.tex.
% Version 97 patch 2, 97/12/12
% For copying restrictions, see pstricks.tex.
%
/tx@Dict 200 dict def
tx@Dict begin
/ADict 25 dict def
/CM { matrix currentmatrix } bind def
/SLW /setlinewidth load def
/CLW /currentlinewidth load def
```

```
/CP /currentpoint load def
/ED { exch def } bind def
/L /lineto load def
/T /translate load def

[...]

end
% END pstricks.pro
```

La première ligne crée un nouveau dictionnaire (on peut voir cela comme un ensemble de variables locales que l'on pourrait rappeler quand on en a besoin, dans le reste du fichier) de 200 mots et la seconde le charge. On définit ensuite diverses commandes. L'avant dernière ligne referme le dictionnaire.

***.ps** Fichier PostScript[4,5,6,7,8,9,10], souvent le résultat final de la compilation d'un fichier LaTeX (après passage par `dvips`). Nous y reviendrons plus loin (7.1).

***.tfm** Métriques d'une fonte, sous une forme lisible par LaTeX mais pas par un humain. On peut les convertir en fichiers `*.pl`, humainement lisibles, à l'aide de `tftopl`.

***.ttc** Fonte True Type Unicode (cela dit, la seule fonte True Type Unicode que j'aie jamais vue s'appelait `*.ttf`)

***.ttf** Fonte True Type (voir chapitre 8). Ce sont des fontes vectorielles essentiellement utilisée sous Windaube. Pour les utiliser sous LaTeX, on peut les convertir en fontes bitmap (`*.pk`) à l'aide de `ttf2pk` ou en fontes PostScript à l'aide de `ttf2pt1`.

***.vf** Fonte virtuelle, créée par la commande `vptovf`, que l'on peut convertir en une forme lisible grace à la commande `vftovp`.

***.vpl** « Virtual property list » : c'est une fonte virtuelle rendue humainement lisible par la commande `vftovp`. Son contenu ressemble beaucoup aux fichier `*.pl`, mais il contient des informations supplémentaires : il nous dit comment créer chaque caractère de la fonte. Par exemple « aller chercher tel caractère dans telle autre fonte » ou « superposer ces deux caractères ». Voici quelques exemples de ce que l'on peut y trouver (en vrac, en provenance de différentes fontes). Nous n'utiliserons pas directement ce genre de fichier, nous verrons plus loin comment `fontinst` permet de manipuler des fontes virtuelles plus facilement.

```
(CHARACTER D 75 (COMMENT K)
   (CHARWD R 6.41992)
   (CHARHT R 6.5299)
   (CHARDP R 0.0)
   (CHARIC R 0.79993)
   (MAP
      (SELECTFONT D 2) (COMMENT ptmri8r at 10.0pt)
      (SETCHAR D 75) (COMMENT K)
      (MOVERIGHT R -0.25)
      )
   )
```

```
(CHARACTER O 1
   (CHARWD R 0.5)
   (CHARHT R 0.492499)
   (MAP
      (SETRULE R 0.5 R 0.5)
      (SPECIAL Warning: missing glyph ‘Delta’)
      )
   )
```

```
(CHARACTER O 13
```

4. `http://partners.adobe.com/asn/developer/technotes.html`
5. `http://bluesky.ecas.ou.edu/{\textasciitilde}bfiedler/5123/post/`
6. `http://alpha.iae.nsk.su/{\textasciitilde}shapiro/misha/ps/tutor.htm`
7. `http://www.quite.com/ps/`
8. `http://www-isis.ecs.soton.ac.uk/computing/system/postscript.html`
9. `http://www.cs.indiana.edu/docproject/programming/postscript/postscript.html`
10. `http://www.geocities.com/SiliconValley/5682/postscript.html"`

```
   (CHARWD R 0.655994)
   (CHARHT R 0.710993)
   (COMMENT
      (LIG C i O 16)
      (LIG C l O 17)
      (KRN O 47 R 0.055994)
      (KRN O 140 R 0.037)
      (KRN O 134 R 0.037)
      (KRN O 42 R 0.055994)
      )
   (MAP
      (SETCHAR C f)
      (SETCHAR C f)
      )
   )
```

```
(CHARACTER D 3
   (COMMENT 0 and 2)
   (CHARWD D 14)     (CHARHT R 14)     (CHARDP R  0)     (CHARIC R  0.0)
   (MAP
      (SETRULE R 14 R 1)(MOVERIGHT R 1)
      (SETRULE R 14 R 1)(MOVERIGHT R 2)
      (SETRULE R 14 R 2)(MOVERIGHT R 1)
      (SETRULE R 14 R 2)(MOVERIGHT R 1)
      (SETRULE R 14 R 1)(MOVERIGHT R 2)
      )
   )
```

psfonts.map Fichier indiquant la correspondance entre les noms de fontes PostScript utilisées par LaTeX (premier champ), le nom PostScript de ces fontes (second champ) et éventuellement le nom des fichiers les contenant ou des noms d'autres fichiers à inclure (par exemple, pour changer le codage). Les lignes peuvent correspondre à des fontes connues par l'imprimante (on ne précise pas de fichier `*.pfa` ou `*.pfb`)

```
pbsi0 <pbsi.pfa
```

ou a des fontes qu'il faudra envoyer à l'imprimante.

```
xycmat10 XYCMAT10 <xycmat10.pfb
```

On rajoute parfois des fichiers contenant des instructions PostScript, pour modifier la fonte (par exemple, le fichier `8r.enc` va permettre définir la commande `TeXBase1Encoding`, qui sera utilisée pour changer le codage de la fonte). On peut aussi rajouter des commandes PostScript, par exemple `TeXBase1Encoding ReEncodeFont` pour changer le codage de la fonte ou `.167 SlantFont` pour pencher un peu la fonte.

```
ptmr8r   Times-Roman  "TeXBase1Encoding ReEncodeFont"                  <8r.enc
ptmro8r  Times-Roman  " .167 SlantFont TeXBase1Encoding ReEncodeFont " <8r.enc
punb8r   Univers-Bold "TeXBase1Encoding ReEncodeFont"                  <8r.enc <punb8a.pfb
xycmat10 XYCMAT10 <xycmat10.pfb
xycmat11 XYCMAT10 "1.1 ExtendFont" <xycmat10.pfb
xycmat12 XYCMAT10 "1.2 ExtendFont" <xycmat10.pfb
ptmro8r  Times-Roman " .167 SlantFont TeXBase1Encoding ReEncodeFont " <8r.enc
ptmrr8re Times-Roman " 1.2 ExtendFont TeXBase1Encoding ReEncodeFont " <8r.enc
ptmrr8rn Times-Roman " .82 ExtendFont TeXBase1Encoding ReEncodeFont " <8r.enc
```

On peut mettre ses ajouts au fichier `psfonts.map` dans un fichier `mypsfonts.map` et l'appeler en ajoutant les lignes suivantes dans le fichier `~/.dvipsrc/` (seule la ligne commençant par `p` est importante).

```
% original
% Default resolution.
D 600
M ljfive
p +mypsfonts.map
a a4
```

***.map** Fichiers que l'on peut rajouter à `psfonts.map`

***.map** Il a d'autres fichiers `*.map` (dans le répertoire `fontname`) qui indiquent la correspondance entre les noms de fonte lisibles, les noms de fonte illisibles utilisés par TeX et les noms de fontes illisibles utilisés par OuinDaube.

ttfonts.map Fichier indiquant la correspondance entre le nom des fontes TTF utilisées par LaTeX et le nom des fichiers les contenant. C'est l'analogue du fichier `psfonts.map` pour les fontes TTF.

Voici un exemple d'un tel fichier : on précise le nom de la fonte pour LaTeX, le nom du fichier TTF, et on peut rajouter le codage désiré (sous forme de fichier `*.enc`, qui est un fichier PostScript) et l'inclinaison.

```
comic_slanted   comic.ttf  Slant=0.25  Encoding=EC.enc
comic           comic.ttf              Encoding=EC.enc
REC_slant_condensed_comic   comic.ttf Slant=.25 Extend=.7 Encoding=T1-WGL4.enc
```

3.3 Comment passer de l'un de ces format de fichier à un autre?

gftodvi Transforme une fonte bitmap `*.gf` en un fichier `*.dvi`.

gftopk Pour compresser un fichier `*.gf` en un fichier `*.pk`.

gftype Transforme une fonte bitmap `*.gf` en un fichier humainement lisible.

pf2afm Transforme des métriques pour OuinDaube `*.pfm` en métriques _< normales ». Comme il y a un peu moins d'information dans les fichiers `*.pfm`, le résultat ne sera pas toujours satisfaisant. On trouve ce programme sur CTAN[11]

pfb2pfa Transforme des fontes PostScript binaires `*.pfb` en fontes PostScript ASCII `*.pfa`. On trouve ce programme sur CTAN[12]. Voir aussi `t1utils`.

pfb2pfa Idem, toujours sur CTAN[13], mais à un autre endroit. Voir aussi `t1utils`.

pktogf Pour décompresser un fichier `*.pk` et obtenir un fichier `*.gf`

pktype Pour rendre humainement lisible le contenu d'un fichier `*.pk`.

t1ascii Pour convertir un fichier `*.pfb` en `*.pfa`.

t1asm Pour rendre le contenu d'un fichier `*.pfa` humainement lisible.

t1binary Pour convertir un fichier `*.pfa` en `*.pfb`.

t1disasm Pour rendre le contenu d'un fichier `*.pfa` humainement illisible.

t1utils Ensemble de commandes (`t1asm`, `t1disasm`, `t1ascii`, `t1binary`, `unpost`) pour manipuler des fontes PostScript Type 1.

tftopl Transforme un fichier `*.tfm` en `*.pl`. Par exemple

```
tftopl toto > toto.pl
```

ttf2pk Transforme une fonte `*.ttf` en fonte bitmap utilisable par `dvips`, voir chapitre 8

ttf2pt1 Transforme une fonte True Type en fonte PostScript, voir 8.15.

ttf2tfm Extrait les métriques d'un fichier `*.ttf`

unpost Pour convertir des fontes PostScript pour Mac (fichiers POST) en fontes PostScript utilisables sur des machines normales.

vftovp Transforme une fonte virtuelle `*.vf` (et ses métriques `*.tfm`) en `*.vpl`.

vptovf Transforme un fichier `*.vpl` en `*.tfm` et `*.vf`.

11. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/pfm2afm`
12. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/ps2pk/ps2pk15/misc/pfb2pfa.c.gz`
13. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/ps2mf/pfb2pfa.c.gz`

3.4 Où LaTeX va-t-il chercher tous ces fichiers?

La structure de l'arborescence d'une installation de LaTeX est décrite dans le TDS (TeX Directory Structure[14]. Rappelons-en les grandes lignes. La racine de l'arborescence peut être `/usr/local/lib/texmf/`. Dans la situation qui nous intéresse (une distribution basée sur `web2c`, utilisant `dvips`), ce répertoire contient

- un répertoire `tex` qui contient les fichiers inclus par LaTeX,
- un répertoire `fonts` qui contient les fontes : plus précisément, il contient les sous-répertoires
  - `tfm` pour les métriques
  - `source` pour les sources métafont
  - `truetype` pour les fontes True Type et le fichier `ttfonts.map`
  - `type1` pour les fontes PostScript Type 1
  - `afm` pour les métriques des fontes PostScript Type 1
  - `vf` pour les fontes virtuelles
  - `pk` pour les fontes bitmap (dans certaines distributions, la résolution des bitmaps n'est plus indiquée dans le nom de fichier, `toto10.300pk`, mais dans le nom du répertoire le contenant, `dpi300/toto.pk`)
- un répertoire `metafont` qui contient les fichiers lus par métafont qui ne sont pas des fontes (par exemple, des macros),
- un répertoire `dvips` qui contient les fichiers de configuration de `dvips`
- un répertoire `web2c` qui contient les formats et le fichier de configuration `texmf.cnf` (en particulier, la liste des répertoires à chercher)
- un fichier `ls-R` qui contient la liste des fichiers contenus dans ce répertoire et qui est utilisé par `kpsearch`. on peut le remettre à jour à l'aide de la commande `mktexlsr` (ou `texhash`, selon la distribution).

La recherche des fichiers par TeX, MetaPost, dvips, etc. est réalisée par la bibliothèque `kpsearch`. Pour savoir où elle va chercher un certain fichier, on peut utiliser la commande `kpsewhich`.

```
kpsewhich --progname=ttf2tfm --expand-var='$TTFONTS'
kpsewhich --progname=ttf2tfm --expand-var='$TTF2PKINPUTS'
kpsewhich --progname=ttf2tfm --expand-var='$TTF2TFMINPUTS'
kpsewhich comic.sty
```

Quand on installe une nouvelle fonte (ou un nouveau paquetage), on peut soit le rajouter dans l'arborescence, soit créer une nouvelle arborescence : cela permet de bien séparer les fichiers qui proviennent de la distribution que l'on a choisie (par exemple teTeX) et ceux que l'on a rajoutés soi-même ; en particulier, on peut réinstaller cette distribution (ou même une autre) sans rien changer à ce que l'on avait rajouté soi-même.

Pour ce faire, il suffit d'éditer le fichier `texmf.cnf` (si vous ne savez pas où il se trouve, utilisez la commande `kpsewhich`). Il contient des choses comme

```
TEXMFMAIN = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf
TEXMF     = $TEXMFMAIN
```

On remplace ces lignes par quelque chose du genre (oui, j'ai bien une dizaine d'arborescences)

```
TEXMFMAIN = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf

  TEXMFLOCAL = /usr/local/lib/texmf
  EXTRATEXMF = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf.MOI
  GUTTEXMF = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf.GUT
  HOMETEXMF = $HOME/.texmf
  CJKTEXMF = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf.CJK
  TEXLIVE = /share/riemann/raid/local/pkg/texlive3/texmf/
  TEXLIVEBIS = /share/riemann/raid/local/pkg/texlive3/local/
  OLDCOPY = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf.old1
  OLDCOPYBIS = /share/riemann/raid/users/umr-tge/zoonek/gnu/lib/texmf.old2

  TEXMF = {$HOMETEXMF:!!$EXTRATEXMF:!!$TEXMFMAIN:!!$GUTTEXMF:
    !!$CJKTEXMF:!!$TEXLIVEBIS:!!$OLDCOPY:!!$OLDCOPYBIS:!!$TEXLIVE}
```

14. `ftp://ftp.lip6.fr://pub/TeX/CTAN/tds/misc/`

Si vous n'avez pas le droit d'éditer le fichier `texmf.cnf`, vous pouvez faire une copie du répertoire `$TEXMF`web2c/ dans votre nouvelle arborescence, éditer la copie du fichier `texmf.cnf`. Pour que LaTeX aille chercher ce fichier-là et pas celui par défaut, il suffit de mettre le nom du nouveau répertoire `web2c` dans la variable d'environement `TEXMFCNF` (modifiez les fichiers de configuration de votre shell). Les doubles points d'exclamation indiquent que si le fichier cherché n'est pas dans la base de données crée par la commande `mktexlsr` (ou `texhash`, pour certaines distributions), il ne faut pas chercher dans l'arborescence : on suppose que cette base de données est toujours à jour.

3.5 Quelle est la différence entre `inputenc` et `fontenc` ?

Dans le préambule d'un document, on conseille souvent de mettre

```
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
```

La première ligne demande d'utiliser des fontes avec le codage `T1` (si on voit une fonte comme un tableau de caractères, le codage est juste l'emplacement des caractères). Il s'agit du codage de la fonte.

La seconde ligne indique le codage du texte : une machine Unix et Ouindoze ne codent pas les lettres accentuées de la même manière. Préciser `latin1` revient à dire que l'on a utilisé une machine Unix pour taper le texte : on peut après le compiler sur un Machintosh sans que les lettres accentuées soient corrompues (voir 3.6).

3.6 Comment taper les lettres accentuées « normalement » ?

Tout d'abord, si LaTeX est bien installé, on peut les taper normalement, i.e., « `é` » au lieu de « `\'e` ». Néanmoins, cela risque de poser un problème de portabilité : en effet, les caractères accentués sont codés différemment selon le type de plateforme (Unix, MacOS, WinDaube). Il faut donc dire quel est le codage utilisé. Si on est sur une machine Unix, il suffit de rajouter

```
\usepackage[latin1]{inputenc}
```

dans le préambule. On peut dès lors saisir les caractères accentués normalement. Sur un Mac, on remplacerait `latin1` par `applemac`, sous DOS, par `cp850`, sous Ouindaube, par `ansinew`.

On peut aussi vouloir utiliser plusieurs codages dans un même document : le problème se pose si, par exemple, plusieures personnes écrivent les différentes parties d'un même document, mais sur des machines différentes. On précise alors le codage par la commande `\inputencoding`.

```
\inputencoding{latin1}%
\include{fichier_tape_sous_unix}
\inputencoding{applemac}%
\include{fichier_tape_sous_unix}
```

3.7 Comment taper le oe normalement ?

Cela dépend de l'éditeur de texte que l'on utilise. Pour (X)Emacs sous Unix, je n'ai pas de solution, à part Unicode.

3.8 Qu'est-ce que le codage ?

Si l'on voit une fonte comme une tableau de dessins de caractères, son codage est la correspondance entre le nom d'un caractère et l'emplacement de son dessin.

Il convient de bien distinguer le codage de la fonte et le codage d'entrée (ie, celui dans lequel on a tapé notre texte, par exemple latin1 sur une machine Unix). La question précédente tentait déjà de préciser cette distinction.

On peut se demander pourquoi il y a différents codages. Si on a besoin juste des lettres latines, il semble très naturel d'utiliser des caractères ASCII. Mais si on a besoin d'autres lettres ? Si on a besoin de caractères accentués ? Si on a besoin de certains symboles mathématiques ? Si on a besoin de caractères cyrriliques ? Si on a besoin de caractères grecs ? Pour chaque problème, on peut apporter une solution, c'est-à-dire concevoir un codage adéquat. Mais comme on se limite à 256 caractères par fontes, on ne parvient pas tout mettre dans une même fonte. Pire encore, si différentes personnes tentent de résoudre le même problème, elles vont le résoudre de manières différentes, ie, aboutir à des codages différents pour le même jeu de caractères (il y a par exemple un nombre incroyable de codages pour le russe). La situation se complique encore pour les écritures qui requièrent plus de 256 caractères...

3.9 Quels sont les différents codages d'entrée ?

Voici une liste (non exhaustive) de codages reconnus par LaTeX

ascii Uniquement les caractères ASCII
latin1 Texte européen sur une machine UNIX (il manque le caractère OE)
decmulti À peu près pareil (mais il doit y avoir un OE)
applemac Texte européen sous MacOS
latin2 Europe de l'est
latin3 Europe du sud
latin5 Turc
ansinew Codage de OuinDaube
cp850 Codage de MSDOS

3.10 Quels sont les différents codages de fontes ?

Voici quelques exemples de codages (la liste n'est pas du tout exhaustive).

T1 C'est le codage actuellement utilisé par LaTeX. Il est aussi connu sous le nom de Cork ou `8t`.
OT1 C'est le codage initialement utilisé par TeX. Il posait des problèmes de césure pour les mots accentués et tend à être supplanté par T1. On le préfère parfois à T1 car il reste plein de place libre pour rajouter des caractères dont on aurait besoin.
Tex Base Encoding C'est le codage utilisé par LaTeX pour les fontes PostScript. Il est aussi connu sous le nom de `8r`.
Adobe Codage utilisé par la plupart des fontes PostScript. Il est aussi connu sous le nom de `8a`. Quand on utilise de fontes PostScript, on se ramène généralement du codage `8a` au codage `8r`.
OML Codage mathématique utilisé par TeX (Math Italics)
OMS Codage mathématique utilisé par TeX (Math Symbols)
OMX Codage mathématique utilisé par TeX (Math Large Symbols)
TS1 Codage utilisé pour les symboles en mode texte (`textcomp.sty` et les fontes TC, « Text Companion »)
OT2 Washington University Cyrillic encoding.
T2A Le codage cyrillique utilisé par défaut par `russianb` dabs `babel`. Il y a plein d'autres codages cyrilliques : OT2, LWN, LCY, X2, T2, T2C, T2B, T2A.
OT3 IPA (alphabet phonétique international), sur 7 bits (128 caractères).
T3 IPA, 256 caractères. Voir `tipa.sty`.
OT4 Polonais
T4 Polonais
LGR Un codage grec.
LV1 l'un des codages de Y&Y.
dvips Il y a aussi un codage propre à `dvips`...
U Codage inconnu.
L... Codage local, normalement non diffusé
E... Codage expérimental (comme le codage local, mais il est destiné à être diffusé).

Cette liste est bien évidemment non limitative.

3.11 Est-il vrai que le codage `OT1` pose des problèmes de césure ?

Oui, parfois. Dans le codage `OT1`, les lettres accentuées ne sont pas des caractères mais sont construites par LaTeX à partir d'une lettre et un accent. Le problème, c'est que TeX ne sais pas couper les mots contenant de telles constructions. Pour y remédier, il y a deux solutions :

– Utiliser une fonte utilisant le codage `T1`, par exemple les fontes `EC` (fontes Métafont fabriquées à partir des fontes Computer Modern) ou les fontes `ae` (fontes virtuelles utilisant les fontes Computer Modern : on y a recours quand on est contraint

d'utiliser les fontes CM, par exemple avec pdfTeX qui veut des fontes PostScript – les fontes CM existent en PostScript mais pas les fontes EC) : les lettres accentuées sont alors des caractères comme les autres.
– Utiliser MlTeX (avec les distributions récentes de (La)TeX, il suffit de donner l'option `-mltex` quand on crée un nouveau format). C'est la solution la plus simple.

3.12 Pourquoi les fontes ont-elles des noms aussi illisibles ?

Cela résulte de l'utilisation répandue d'un certain système d'exploitation qui limite le nombre de caractères d'un nom de fichier à 8 caractères. Pour que les fontes aient le même nom sur toutes les machines, on s'efforce d'utiliser ces noms réduits sur toutes les machines. La signification exacte de ces nom est expliquée dans `fontname`[15]. En voici une présentation (incomplète et) simplifiée.

– Première lettre : Nom de l'entreprise qui commercialise la fonte. Voici quelques exemples.

9 inconnu
f petites entreprises, particuliers, public
p Adobe
z le nom de la fonte ne satisfait pas à fontname.

– Deux lettres suivantes : nom de la fonte Voici quelques exemples.

ad Adobe Garamond
ga ITC Garamond (distribué par Adobe) [Il y a encore plein d'autres fontes Garamond.]
tm Times (il y a aussi plein de fontes Times)

– Lettre suivante : le poids. Quelques exemples :

b gras
r normal

– Lettres suivantes : variantes (italique, SC, sans sérif, codage)

i italique
o penché (o comme oblique)
s sans sérif
c SC
w manuscrit
8 + une lettre codage sur 8 bits
8r TeXBase1Encoding
8t CorkEncoding

– Lettre suivante : largeur (généralement omise)

En fait, cela ne suffit bien évidemment pas. On utilise donc un fichier qui permet de passer d'un nom de fonte long, du genre `AGaramond-SemiboldItalic`, à un nom de fonte court, du genre `padsi8a`, utilisable sur des ordinateurs munis de systèmes d'exploitation archaïques.

Ce fichier `fontname.map` est lu par web2c pour traduire le nom de fichier donné dans le source TeX (qui peut donc être arbitrairement long) en un nom de fichier court.

Pourquoi s'embêter avec tout cela ? Si on est sûr de ne jamais utiliser de ces systèmes d'exploitation archaïques n'autorisant pas plus de 8 caractères dans les noms de fichiers, pourquoi ne pas utiliser des noms plus explicites ?

Bonne question. Le plus simple est souvent de ne pas s'en préoccuper. Mais on ne peut alors plus utiliser de logiciels qui permettent de manipuler des fontes en supposant qu'elles sont installées en respectant ces conventions (`vfinst`, `mathkit`, `mathinst`).

Une autre solution consiste à ne pas chercher à comprendre fontname et à utiliser des noms arbitraires, conformes à fontname. Par exemple

9 créateur inconnu

15. `ftp://ftp.lip6.fr/pub/TeX/CTAN/info/fontname/fontname_toc.html`

pi000 le nom de la fonte (les deux lettres « pi » sont conformes à fontinst, elles sont suivies de trois chiffres — cela nous fait dépasser des 8 caractères généreusement accordés par B. Gates, mais ce n'est pas grave)

8r codage utilisé par TeX pour les fontes PS ou TTF

Il ne reste plus qu'à dire de quel type de fonte il s'agit (droite, penchée ou italique ; maigre ou grasse ; sérif, sans sérif, machine à écrire ; petites capitles ou non ; etc.)

Voici quelques exemples (je mets des espaces pour plus de lisibilité)

```
9 pi000 r 8r : Fonte normale
9 pi000 b 8r : Fonte grasse
9 pi000 r s 8r : Fonte normale sans sérif
9 pi000 b s 8r : Fonte grasse sans sérif
9 pi000 r t 8r : Fonte normale Machine à écrire
9 pi000 b t 8r : Fonte grasse Machine à écrire
9 pi000 r c 8r : Fonte normale petites capitales
9 pi000 b c 8r : Fonte grasse petites capitales
9 pi000 r i 8r : Fonte normale italique
9 pi000 b i 8r : Fonte grasse italique
9 pi000 r o 8r : Fonte normale oblique
9 pi000 b o 8r : Fonte grasse oblique
```

Et il en manque plein, mais je ne sais pas si on met « i c » ou « c i » pour des petites capitales italiques (c'est peut-être simplement interdit), etc.

3.13 Pourquoi y a-t-il plein de chiffres dans certains noms de fontes : `ecrm1000.tfm` ?

Cela n'apparait pas dans toutes les fontes, mais juste dans certaines fontes Métafont récentes, par exemple, les fontes EC. La fonte Computer Modern maigre droite était présente en plusieures versions différentes, correspondant à des tailles différentes (`cmr5`, `cmr10`, `cmr17`, etc.). Il s'agissait vraiment de fontes différentes car, pour des raisons de lisibilité, les fontes de petites tailles sont relativement plus larges que des fontes de grande taille. Ainsi, sur la figure suivante, on a utilisé la fonte Computer Modern à 6 point et è 35 points, avec la même graisse.

ABCabc ABCabc

Pour les fontes EC, c'est pareil, mais il y a beaucoup plus de tailles prévues et ces tailles ne sont pas données en points dans le nom des fontes, mais en centièmes de points. Ainsi, la fonte `ecrm1000` est-elle l'équivalent de `cmr10`.

Certains de ces fichiers peuvent d'ailleurs être crés automatiquement par `mktexmf`. À DÉTAILLER

Chapitre 4

NFSS (New Font Selection Scheme)

4.1 Qu'est-ce que NFSS ?

Initialement, les fontes sous (La)TeX étaient considérées comme complètement distinctes les unes des autres : il y avait par exemple une fonte droite, une fonte grasse, une fonte italique, une fonte italique grasse, mais le logiciel ne connaissait pas leurs attributs et n'était pas conscient qu'elles faisaient partie d'une même famille. Ainsi, si on demandait du gras au début d'un paragraphe, puis, pour un mot à l'intérieur du paragraphe, de l'italique, il fallait explicitement demander de l'italique maigre ou de l'italique gras : on ne pouvait pas dire, simplement, « de l'italique avec la même graisse ». En d'autres termes, les changements de fontes n'étaient pas orthogonaux.

Le NFSS [1] (New Font Selection Scheme) permet de spécifier séparément certaines des caractéristiques des fontes : inclinaison (droit, penché, italique, petites capitales, etc.), graisse (normal, gras, gras condensé, gras étendu), taille. Cela permet à peu près toutes les combinaisons (sauf les petites capitales italiques).

4.2 Quelles sont les caractéristiques d'une fonte ?

- Son codage : OT1 (vieilles fontes TeX, dans lesquelles les caractères accentués sont construits à partir de deux caractères, l'accent et la lettre), T1 (fontes TeX plus récentes : les caractères accentués sont directement présents), OML (italique mathématique), OMS (symboles mathématiques), OMX (grands symboles), etc. ;
- Sa famille : cmr (Computer Modern), cmss (Computer Modern Sans Sérif), cmtt (Computer Modern Typewriter), cmm (math italics), cmsy (math symbols), cmex (math large symbols), ptm (Times), phv (Helvetica), pcr (Courrier), etc. ;
- Sa série : médium (m), gras (b), gras étendu (bx), semi-gras (sb), condensé (c) ;
- Sa forme : droit (n), italique (it), penché (sl), petites capitales (sc), etc.

4.3 Comment changer de fonte (commandes de « bas niveau ») ?

Nous avons déjà vu les commandes de « haut niveau » : `\rmfamily`, `\sffamily`, `\ttfamily`, `\mdseries`, `\bfseries`, `\upshape`, `\itshape`, `\slshape`, `\scshape`.

Les commandes de bas niveau sont les suivantes (attention, elles supposent que les fontes sont décrites quelque part, voir 4.6) :

```
\fontencoding{T1}%
\fontfamily{cmr}%
\fontseries{bx}%
\fontshape{n}%
\fontsize{10pt}{12pt}%
\selectfont
```

On commence par définir les caractéristique de la fonte (les deux dimensions sont d'une part la taille des caractères, d'autre par la taille de l'interligne (pour augmenter l'interligne, il est généralement plus juducieux d'utiliser l'extension `setspace`) et on appelle la commande `\selectfont`.

Il y a une commande un peu plus évoluée :

1. `http://www.loria.fr/services/tex/ctan-doc/macros/latex/base/fntguide.dvi`

```
\usefont{T1}{cmr}{bx}{n}
```

Les commandes suivantes contiennent les caractéristiques de la fonte courrante : `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, `\f@size`, `\f@baselineskip`.

Il y en a aussi trois autres, qui contiennent la taille de texte mathématique en taille normale, en indice et en indice d'indices (que l'on peut définir à l'aide de la commande `\mathsize` : ses quatres arguments sont la taille du texte et les trois tailles mathématiques) : `\tf@size`, `\sf@size` et `\ssf@size`.

4.4 Comment avoir des petites capitales italiques ?

La question est problématique, car italique (ou penché) et petites capitales sont tous deux des « formes » de la fonte : on les sélectionne par les commandes `\ifshape`, `\slshape` ou `\scshape`. Il s'ensuit qu'on ne peut pas avoir de petites capitales penchées si l'on s'en tient strictement à NFSS.

Néanmoins, ces fontes existent, par exemple dans la famille EC (si elles n'existaient pas, on pourrait les fabriquer, voir 8.11, 9.30, 9.31, 7.18) : `ecsc*.mf` pour les petites capitales penchées maigres et `ecoc*.mf` pour les grasses. Pour les utiliser, il faut court-circuiter NFSS : on peut le faire salement, en reprenant les commandes TeX (qui ne tiennent pas compte de la taille courrante, ni de la graisse courrante).

```
\font\scsl=ecsc1000 at 12pt
\font\scslbf=ecoc1000 at 12pt
```

Abcd **Abcd**
Abcd ***Abcd***

On peut aussi le faire plus proprement, sans quitter NFSS, en définissant une nouvelle forme de caractères `scsl`.

```
% Copié depuis t1cmr.fd
\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><29.86><35.83>genb*#5}{}}
\DeclareFontFamily{T1}{cmr}{}
\EC@family{T1}{cmr}{m}{n}{ecrm}
\EC@family{T1}{cmr}{m}{sl}{ecsl}
\EC@family{T1}{cmr}{m}{it}{ecti}
\EC@family{T1}{cmr}{m}{sc}{eccc}
\EC@family{T1}{cmr}{bx}{n}{ecbx}
\EC@family{T1}{cmr}{b}{n}{ecrb}
\EC@family{T1}{cmr}{bx}{it}{ecbi}
\EC@family{T1}{cmr}{bx}{sl}{ecbl}
\EC@family{T1}{cmr}{bx}{sc}{ecxc}
\EC@family{T1}{cmr}{m}{ui}{ecui}
% Je rajoute les deux lignes suivantes
\EC@family{T1}{cmr}{m}{scsl}{ecsc}
\EC@family{T1}{cmr}{bx}{scsl}{ecoc}

\let\NFSSscshape\scshape
\let\NFSSitshape\itshape
\let\NFSSslshape\slshape
\def\scshape{%
  \def\tempa{it}%
  \ifx\tempa\f@shape\scslshape\else
  \def\tempa{sl}%
  \ifx\tempa\f@shape\scslshape\else
  \def\tempa{scsl}%
  \ifx\tempa\f@shape\scslshape\else
```

```
  \NFSSscshape
  \fi\fi\fi}
\def\itshape{%
  \def\tempa{sc}%
  \ifx\tempa\f@shape\scslshape\else
  \def\tempa{scsl}%
  \ifx\tempa\f@shape\scslshape\else
  \NFSSitshape
  \fi\fi}
\def\slshape{%
  \def\tempa{sl}%
  \ifx\tempa\f@shape\scslshape\else
  \def\tempa{scsl}%
  \ifx\tempa\f@shape\scslshape\else
  \NFSSslshape
  \fi\fi}
\def\scslshape{\fontshape{scsl}\selectfont}
```

Signalons aussi l'existence d'une extension `scaps.sty` permettant d'utiliser plus proprement les petites capitales.

4.5 Comment utiliser ponctuellement une fonte ?

Sélectionner la fonte à l'aide de la commande `\selectfont` ne suffit pas, car le nom de la fonte pour TeX et le nom des fichiers correspondants n'est pas nécessairement le même (en fait, une même fonte peut avoir une demi-douzaine de noms différents : celui pour TeX, celui pour fontname, celui pour le système de fichier, celui pour l'interpréteur PostScript — je suis sûr que j'en oublier, mais lesquels ?).

Pour utiliser une fonte appelée `toto10` de manière ponctuelle, il suffit de mettre les lignes suivantes dans l'en-tête du fichier et d'utiliser la commande `\texttutu`.

```
\DeclareFontFamily{U}{tutu}{}
\DeclareFontShape{U}{tutu}{m}{n}
        { <5><6> toto5
        <7><8><9> toto7
        <10><10.95><12><14.4><17.28><20.74><24.88> toto10
        }{}
\newcommand{\texttutu}[1]
{{\fontencoding{U}\fontfamily{tutu}\selectfont #1}}
\newcommand{\tutufamily}
{\fontencoding{U}\fontfamily{tutu}\selectfont}
```

La première ligne déclare une nouvelle famille (vide pour l'instant). Cette famille utilisera le codage U (généralement, pour une fonte qui ne comporte que des caractères ASCII ou une fonte qui comporte des symboles, que l'on n'utilisera pas directement). Les codages les plus courrants, pour les fontes TeX, sont T1 et OT1.

Le dernier argument (très souvent vide) contient des commandes à exécuter juste après avoir sélectionné la fonte : il peut par exemple s'agir d'ajustements des dimensions. Nous y reviendrons plus loin (4.7)

La deuxième ligne stipule que pour la fonte `U/tutu/m/n`, seules les tailles 5, 6, 7, 8, 9, 10, 10.95, 12, 14.4, 17.28 et 24.88 points sont disponibles et qu'il faut aller les chercher dans les fichiers `toto5.tfm`, `toto7.tfm` et `toto10.tfm`. Rappelons à cette occasion que les seuls fichiers concernant une fonte dont TeX a besoin sont les métriques `*.tfm`. Il n'a pas besoin du dessin des caractères (c'est `xdvi` ou `dvips` qui s'en chargent). Remarquons aussi que le nom de la fonte pout LaTeX (`tutu`) peut être différent des noms de fichiers (`toto`).

Les dernières lignes définissent une macro `\texttutu` et un environement `tutufamily` permettant d'utiliser notre fonte de manière ponctuelle.

Signalons aussi qu'il existe des commandes pour définir de telles macros.

```
\DeclareFixedFont{\tutu}{U}{tutu}{m}{n}{5}
```

```
\DeclareTextFontCommand{\textrm}{\rmfamily}
```

```
\DeclareTextFontCommand{\texttutu}{\normalfont\itshape}
```

```
\DeclareOldFontCommand{\it}{\itshape}{\mathit}
```

4.6 Comment utiliser ponctuellement une fonte (bis) ?

En fait, l'exemple précédent n'est pas très propre : normalement, la description de la fonte est cachée dans un fichier `*.fd` qui est chargé automatiquement par LaTeX quand il en a besoin.

Dans un fichier `toto.sty`, mettons les lignes suivantes :

```
\newcommand{\texttutu}[1]
{{\fontencoding{U}\fontfamily{tutu}\selectfont #1}}
\newcommand{\tutufamily}
{\fontencoding{U}\fontfamily{tutu}\selectfont}
```

Dans un fichier `ututu.fd`, mettons les lignes suivantes.

```
\DeclareFontFamily{U}{tutu}{}
\DeclareFontShape{U}{tutu}{m}{n}
        { <5><6> toto5
        <7><8><9> toto7
        <10><10.95><12><14.4><17.28><20.74><24.88> toto10
        }{}
```

Rappelons que les fichiers `*.fd` sont nommés de la manière suivante : le codage suivi du nom de la famille suivi de `.fd`, le tout en minuscules.

Nous pouvons dès lors utiliser cette fonte de la manière suivante.

```
\documentclass[12pt,a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{toto}
\begin{document}
Ce texte est tapé avec le codage d'entrée latin1 (car je suis sur
une machine Unix), avec le codage de fonte T1 et les fontes standard
de LaTeX.

\begin{tutufamily}
Maintenant, nous utilisons les fontes << toto >>, avec le codage U.
On remarquera qu'il manque plein de caractères dans ce codage, par
exemple, les lettres accentuées : ce n'est pas un codage adapté aux
textes français.
\end{tutufamily}

Nous retournons aux fontes usuelles de LaTeX, mais nous pouvons mettre
\texttutu{quelques mots} dans notre nouvelle fonte.
\end{document}
```

4.7 Quel est le dernier argument des commandes `\DeclareFontFamily` et `\DeclareFontS`

Le dernier argument de ces commandes est généralement vide, mais pas toujours. On l'utilise pour changer les valeurs par défaut de `\skewchar` (par défaut -1) ou `\hyphenchar` (le caractère utilisé pour effectuer la césure, par défaut le tiret). Ainsi, on peut inhiber la césure

```
\DeclareFontFamily{OT1}{cmtt}{\hyphenchar\font=-1}
```

ou changer le caractère utilisé pour la césure

```
\DeclareFontFamily{OT1}{cmvtt}{\hyphenchar\font45 }
```

La commande `\skewchar` permet de régler la position des accents en mode mathématique : pour les fontes droites, la valeur par défaut (qui consiste à mettre l'accent juste au milieu) convient, par contre, pour les fontes italique, il faut déplacer l'accent sur la droite. Je ne sais pas à quoi correspond le nombre qu'on lui donne. (A FAIRE) Je ne sais pas si on peut mettre autre chose que les instructions `\skewchar` et `\hyphenchar`. (A FAIRE)

```
\DeclareFontFamily{U}{eur}{\skewchar\font'177}
```

4.8 Quelles autres choses peut-on mettre dans un fichier `*.fd` ?

On peut indiquer une taille unique.

```
\DeclareFontShape{U}{toto}{m}{n}{<10> toto}{}
```

On peut indiquer plusieures tailles.

```
\DeclareFontShape{OT1}{cmtt}{m}{n}{
  <5><6><7><8> cmtt8
  <9>          cmtt9
  <10><10.95>  cmtt10
  <12><14.4><17.28><20.74><24.88> cmtt12}{}
```

La taille peut être un intervalle.

```
\DeclareFontShape{U}{euex}{m}{n}{<-8> euex7 <8-> euex10}{}
```

Il est possible de créer automatiquement les noms de fontes à partir des tailles demandées. Les deux exemples suivants sont équivalents.

```
\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 }{}
```

```
\DeclareFontShape{OT1}{cmr}{m}{n}{
  <5>      cmr5
  <6>      cmr6
  <7>      cmr7
  <8>      cmr8
  <9>      cmr9
  <10>     cmr10
  <12>     cmr12
  <10.95>  cmr10
  <14.4>   cmr12
  <17.28><20.74><24.88>  cmr17 }{}
```

La commande `genb` (respectivement `sgenb`) est semblable mais utilise quatre chiffres : les deux exemples suivants sont équivalents.

```
\DeclareFontShape{T1}{cmr}{m}{n}{
  <5><6><7><8><9><10><10.95><12><14.4>
  <17.28><20.74><24.88><29.86><35.83>  genb*ecrm}{}}
```

```
\DeclareFontShape{T1}{cmr}{m}{n}{
  <5>      ecrm0500
  <6>      ecrm0600
  <7>      ecrm0700
  <8>      ecrm0800
  <9>      ecrm0900
  <10>     ecrm1000
  <10.95>  ecrm1095
  <12>     ecrm1200
  <14.4>   ecrm1440
  <17.28>  ecrm1728
  <20.74>  ecrm2074
  <24.88>  ecrm2488
  <29.86>  ecrm2986
  <35.83>  ecrm3583 }{}}
```

Il est possible de dire à LaTeX d'aller chercher les fontes dans d'autre familles. Ces substitutions peuvent être silencieuses (`ssub`) ou non (`sub`).

```
\DeclareFontShape{OT1}{cmtt}{m}{ui}{<->ssub*cmtt/m/it}{}
\DeclareFontShape{OT1}{cmtt}{bx}{n}{<->ssub*cmtt/m/n}{}
\DeclareFontShape{OT1}{cmtt}{bx}{it}{<->ssub*cmtt/m/it}{}
\DeclareFontShape{OT1}{cmtt}{bx}{ui}{<->ssub*cmtt/m/it}{}
```

La commande `fixed` (respectivement `sfixed`) demande de ne pas tenir compte de la taille demandée par l'utilisateur. Ainsi, dans l'exemple suivant, les fontes entre `{U}{euex}{m}{n}{5}` et `{U}{euex}{m}{n}{8}` seront toutes des fontes à 7 points (si la fonte `euex` est à 7 points).

```
\DeclareFontFamily{U}{euex}{}
\DeclareFontShape{U}{euex}{m}{n}{
  <5-8> sfixed * euex7 <8> <9> gen * euex
  <10> <10.95> <12> <14.4> <17.28> <20.74> <24.88> euex10
  }{}
```

La plupart des fontes sont conçues à une taille de 10 points. Dans l'exemple suivant, on demande que les fontes de taille inférieure à 8 (il s'agit de la taille pour l'utilisateur) soient en fait de taille 12.4 (il s'agit de la taille de la fonte, par rapport à la taille à laquelle elle a été conçue), que les fontes de taille 8 à 10 soient de taille 12.8 et que les fontes de taille supérieures soient de taille 12.8

```
\DeclareFontFamily{U}{heb}{}
\DeclareFontShape{U}{heb}{m}{n}{%
   <-8> fixed * [12.4] hclassic
   <8-10> s * [12.8] hclassic
   <10><10.95><12><14.4><17.28><20.74><24.88> s * [12.8] hclassic }
```

Comme le suggère l'exemple précédent, on peut mettre un facteur multiplicatif entre crochets avec n'importe quelle commande (on ne rajoute pas de signe de multiplication `*` pour la commande « vide »).

On peut rajouter soi-même des fonctions à l'aide de `\DeclareSizeFunction{name}{code...}`. Pour de plus amples détails, nous renvoyons à fntguide[2].

2. `ftp://ftp.lip6.fr/pub/TeX/CTAN/macros/latex/doc/fntguide.dvi`

4.9 Comment accéder à un caractère d'une fonte en connaissant juste son numéro ?

La primitive TeX permettant d'obtenir un caractère dans la fonte courrante à partir de son numéro est `\char`. La commande LaTeX correspondante s'appelle `\symbol`, simplement définie comme

```
\def\symbol#1{\char #1\relax}
```

On peut l'utiliser ainsi (les doubles accolades sont nécessaires pour que le changement de fonte soit limité au caractère).

```
\newcommand{\toto}
{{\fontencoding{U}\fontfamily{wasy}\selectfont\symbol{27}}}
```

Il existe néanmoins une manière un peu plus propre d'effectuer cela, à l'aide de la commande `\DeclareTextSymbol` (voir fntguide[3] pour les détails)

```
\DeclareTextSymbolDefault{\texteuro}{TS1}
\DeclareFontEncoding{TS1}{}{}
\DeclareFontSubstitution{TS1}{cmr}{m}{n}
\DeclareTextSymbol{\texteuro}{TS1}{191}
```

Très souvent, les symboles en question seront utilisés en mode mathématique : c'est alors un peu plus compliqué, car LaTeX doit savoir de quel type de symbole il s'agit : une lettre, une relation binaire, un délimiteur gauche, etc. Cette information est nécessaire pour avoir un espacement correct de chaque côté du symbole. Les commandes suivantes peuvent s'utiliser directement et prennent alors le caractère voulu (ou n'importe quoi d'autre) en argument et s'occuppent de ces espaces.

- `\mathalpha` : une lettre ;
- `\mathop` : un opérateur (unaire), comme par exemple `\sin` ou `\hom` ;
- `\mathbin` : un opérateur binaire, comme par exemple `+` ;
- `\mathrel` : une relation binaire, comme par exemple `=`, `\neq` ou `\mid` ;
- `\mathopen` : un délimiteur ouvrant, par exemple `(` ou `\langle` ;
- `\mathclose` : un délimiteur fermant, comme par exemple `)` ou `\rangle` ;
- `\mathpunct` : une ponctuation ;
- `\mathord` : n'importe quoi d'autre, qui sera traité à peu près comme une lettre, par exemple `\hbar`, `\alpha` ou `\bigstar`.

Après avoir déclaré les fontes mathématiques, on définit les symboles que l'on désire à l'aide de la commande `\DeclareMathSymbol`, comme dans l'exemple suivant.

```
\DeclareSymbolFont{stmry}{U}{stmry}{m}{n}
\SetSymbolFont{stmry}{bold}{U}{stmry}{b}{n}
\DeclareMathSymbol\shortleftarrow\mathrel{stmry}{"00}
\DeclareMathSymbol\shortrightarrow\mathrel{stmry}{"01}
```

4.10 Comment changer la fonte par défaut du document ?

Nous avons déjà vu plus haut comment faire s'il y avait déjà un fichier `*.sty` qui chargeait la fonte : nous regarderons donc ici ce que l'on peut faire si l'on est laissé à soi-même. Il s'agit essentiellement de redéfinir une ou plusieures des commandes suivantes (il faut toutefois que LaTeX parvienne à trouver les fichiers `*.fd` et les métriques correspondantes) : `\rmdefault`, `sfdefault`, `ttdefault`, `bfdefault`, `mddefault`, `\itdefault`, `\sldefault`, `\scdefault`, `\updefault`, `\encodingdefault`, `\familydefault`, `\seriesdefault`, `\shapedefault`.

Voire même l'une des commandes suivantes :

```
<tt>\tiny</tt>,
<tt>\scriptsize</tt>,
<tt>\footnotesize</tt>,
<tt>\small</tt>,
<tt>\normalsize</tt>,
```

3. `ftp://ftp.lip6.fr/pub/TeX/CTAN/macros/latex/doc/fntguide.dvi`

```
<tt>\large</tt>,
<tt>\Large</tt>,
<tt>\LARGE</tt>,
<tt>\huge</tt>,
<tt>\Huge</tt>.
```

Par exemple :

```
\renewcommand{\normalsize}{\fontsize{10}{12}\selectfont}
```

Voici un exemple plus compliqué : on cherche à imiter une machine à écrire. Il n'y a donc plus qu'une seule fonte, qu'une seule taille. (Nous verrons plus loin (4.13) comment nous occuper des fontes en mode mathématique.)

```
\def\rmdefault{cmtt}
\def\sfdefault{cmtt}%
\def\ttdefault{cmtt}
\def\bfdefault{bx} % Il n'y a pas de gras, de toute manière
\def\mddefault{m}
\def\itdefault{u}
\def\sldefault{u}
\def\scdefault{u}
\def\updefault{u}
\renewcommand{\emph}{\uline} % From the ulem package
\renewcommand{\textbf}{\uline}
\let\small\normalsize
\let\footnotesize\normalsize
\let\scriptsize\normalsize
\let\tiny\normalsize
\let\large\normalsize
\let\Large\normalsize
\let\LARGE\normalsize
\let\huge\normalsize
\let\Huge\normalsize
\let\HUGE\normalsize
```

4.11 Comment utiliser plusieurs codages de fonte en même temps ?

Cette question n'a pas grand sens : quand on demande à LaTeX d'utiliser une fonte, cela implique un certain codage. Si à un autre endroit du texte on demande une autre fonte, elle impliquera un autre codage. Ces différents codages correspondent à différentes parties du texte (éventuellement incluses les unes dans les autres) : cela ne pose donc aucun problème.

La question peut quand même avoir un sens dans le cas de fontes possédant plusieurs codages. Le premier exemple qui vient à l'esprit est celui des fontes Computer Modern, qui existent sous les codages OT1 et T1. On peut demander explicitement à LaTeX d'utiliser l'un ou l'autre de ces codages, par exemple à l'aide de la commande

```
\usepackage[T1]{fontenc}
```

Le codage `T1` est le plus récent (mais ce n'est pas celui par défaut, pour ces raisons de compatibilité arrière), il contient beaucoup plus de caractères (les caractères accentués) et permet des césures correctes des langues accentuées même dans `mltex`. Il présente deux inconvénients : comme il y a beaucoup plus de caractères, il n'y a plus de place dans la fonte si on veut en rajouter ; d'autre part, il n'y a pas de fontes PostScript (gratuites) Computer Modern avec le codage T1 (on ne peut donc pas directement utiliser `pdftex` avec le codage `T1` ; mais les fontes virtuelles `ae` permettent de contourner ce problème).

On a aussi besoin de plusieurs codages pour une même fonte si cette fonte comporte plus de 256 caractères : ainsi, pour écrire un document en Computer Modern à la fois en français et en russe, on pourra utiliser la même fonte, tantôt avec le codage `T1`, tantôt avec le codage `OT2`. Ainsi, la ligne suivante

```
\usepackage[OT2,T1]{fontenc}
```

demande à LaTeX de lire les fichiers `t1enc.def` et `ot2enc.def` qui définissent les deux codages et dit que le document sera essentiellement en `T1`. La commande `\fontencoding{???}\selectfont` (où il faut remplacer les points d'interrogation `???` par le nom du codage de fonte) permet de changer le codage de la fonte courrante, comme vu en 4.3.

4.12 Comment utiliser une fonte texte en mode mathématique ?

Le plus simple est d'utiliser la commande `\text` définie dans `amsmath`, qui permet de mettre du texte en mode mathématique en respectant la taille courrante de l'écriture (taille des indices et des indices d'indice). La commande `\text` respecte les caractéristiques du texte alentours, par exemple, dans une formule mathématique au milieu d'un texte en gras italique, elle va prendre du gras italique. Ce n'est pas toujours désiré. Si on n'en veut pas, on peut revenir à la fonte par défaut grace à `\text{\normalfont ...}`. Il ne faut pas utiliser la commande `\mbox`, car elle ne respecte pas la taille des indices ou des exposants.

Comme il s'agit d'une fonte texte, on peut utiliser des caractères accentués : ce n'est pas le cas des fontes mathématiques. Si on désire appeler `Homéo(X,Y)` (dans une fonte maigre droite) l'ensemble des homéomorphismes entre des espaces topologiques `X` et `Y`, ou `Groupoïdes` (dans une fonte gothique grasse) la 2-catégorie des groupoïdes, cela ne pose pas de problème. Avec les commandes `DeclareMathOperator` ou `mathfrak`, on ne pouvait pas mettre d'accents...

4.13 Comment utiliser une fonte texte comme si c'était une fonte mathématique ?

D'après la réponse à la question précédente, on voit que ce n'est peut-être pas une très bonne idée : d'une part il est plus simple d'utiliser une fonte texte et la commande `\text`, d'autre part, on perdrait la possibilité d'utiliser des accents.

Toutefois, si on insiste, il suffit de rajouter les deux lignes suivantes, qui définissent une commande `\mathTOTO`.

```
\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
```

4.14 Comment utiliser une fonte texte comme fonte mathématique par défaut ? Comment modifier certains caractères mathématiques ?

On peut aussi vouloir changer la fonte utilisée en mode mathématique *par défaut*. Si l'on essaye les lignes suivantes (qui tentent d'utiliser une fonte sans sérif), tout semble marcher parfaitement.

```
\DeclareSymbolFont{letters}{OT1}{cmss}{m}{n}
\SetSymbolFont{letters}{bold}{OT1}{cmss}{bx}{n}
\DeclareSymbolFont{operators}{OT1}{cmss}{m}{n}
\SetSymbolFont{operators}{bold}{OT1}{cmss}{bx}{n}
```

Soit f une fonction continue sur $[0; 1]$.

Mais essayons maintenant d'utiliser des symboles ou des lettres grecques (ici, un phi).

Soit Œ une fonction continue sur $[0; 1]$.

Le problème vient du fait que la fonte utilise un codage `T1` (ou `T1` ou n'importe quoi d'autre, mais c'est un codage de fonte texte), alors que LaTeX attend un codage du genre `OMX`.

On peut ne changer que les caractères alphanumériques, comme suit. Les autres caractères (lettres grecques, symboles) seront pris dans la fonte mathématique précédente.

```
\DeclareSymbolFont{pletters}{OT1}{cmss}{m}{n}
\SetSymbolFont{pletters}{bold}{OT1}{cmss}{bx}{n}
\DeclareSymbolFont{operators}{OT1}{cmss}{m}{n}
\SetSymbolFont{operators}{bold}{OT1}{cmss}{bx}{n}

\DeclareMathSymbol{a}{\mathalpha}{pletters}{`a}
\DeclareMathSymbol{b}{\mathalpha}{pletters}{`b}
```

```
\DeclareMathSymbol{c}{\mathalpha}{pletters}{`c}
\DeclareMathSymbol{d}{\mathalpha}{pletters}{`d}
\DeclareMathSymbol{e}{\mathalpha}{pletters}{`e}
\DeclareMathSymbol{f}{\mathalpha}{pletters}{`f}
\DeclareMathSymbol{g}{\mathalpha}{pletters}{`g}
\DeclareMathSymbol{h}{\mathalpha}{pletters}{`h}
\DeclareMathSymbol{i}{\mathalpha}{pletters}{`i}
\DeclareMathSymbol{j}{\mathalpha}{pletters}{`j}
\DeclareMathSymbol{k}{\mathalpha}{pletters}{`k}
\DeclareMathSymbol{l}{\mathalpha}{pletters}{`l}
\DeclareMathSymbol{m}{\mathalpha}{pletters}{`m}
\DeclareMathSymbol{n}{\mathalpha}{pletters}{`n}
\DeclareMathSymbol{o}{\mathalpha}{pletters}{`o}
\DeclareMathSymbol{p}{\mathalpha}{pletters}{`p}
\DeclareMathSymbol{q}{\mathalpha}{pletters}{`q}
\DeclareMathSymbol{r}{\mathalpha}{pletters}{`r}
\DeclareMathSymbol{s}{\mathalpha}{pletters}{`s}
\DeclareMathSymbol{t}{\mathalpha}{pletters}{`t}
\DeclareMathSymbol{u}{\mathalpha}{pletters}{`u}
\DeclareMathSymbol{v}{\mathalpha}{pletters}{`v}
\DeclareMathSymbol{w}{\mathalpha}{pletters}{`w}
\DeclareMathSymbol{x}{\mathalpha}{pletters}{`x}
\DeclareMathSymbol{y}{\mathalpha}{pletters}{`y}
\DeclareMathSymbol{z}{\mathalpha}{pletters}{`z}
\DeclareMathSymbol{A}{\mathalpha}{pletters}{`A}
\DeclareMathSymbol{B}{\mathalpha}{pletters}{`B}
\DeclareMathSymbol{C}{\mathalpha}{pletters}{`C}
\DeclareMathSymbol{D}{\mathalpha}{pletters}{`D}
\DeclareMathSymbol{E}{\mathalpha}{pletters}{`E}
\DeclareMathSymbol{F}{\mathalpha}{pletters}{`F}
\DeclareMathSymbol{G}{\mathalpha}{pletters}{`G}
\DeclareMathSymbol{H}{\mathalpha}{pletters}{`H}
\DeclareMathSymbol{I}{\mathalpha}{pletters}{`I}
\DeclareMathSymbol{J}{\mathalpha}{pletters}{`J}
\DeclareMathSymbol{K}{\mathalpha}{pletters}{`K}
\DeclareMathSymbol{L}{\mathalpha}{pletters}{`L}
\DeclareMathSymbol{M}{\mathalpha}{pletters}{`M}
\DeclareMathSymbol{N}{\mathalpha}{pletters}{`N}
\DeclareMathSymbol{O}{\mathalpha}{pletters}{`O}
\DeclareMathSymbol{P}{\mathalpha}{pletters}{`P}
\DeclareMathSymbol{Q}{\mathalpha}{pletters}{`Q}
\DeclareMathSymbol{R}{\mathalpha}{pletters}{`R}
\DeclareMathSymbol{S}{\mathalpha}{pletters}{`S}
\DeclareMathSymbol{T}{\mathalpha}{pletters}{`T}
\DeclareMathSymbol{U}{\mathalpha}{pletters}{`U}
\DeclareMathSymbol{V}{\mathalpha}{pletters}{`V}
\DeclareMathSymbol{W}{\mathalpha}{pletters}{`W}
\DeclareMathSymbol{X}{\mathalpha}{pletters}{`X}
\DeclareMathSymbol{Y}{\mathalpha}{pletters}{`Y}
\DeclareMathSymbol{Z}{\mathalpha}{pletters}{`Z}
\DeclareMathSymbol{0}{\mathalpha}{operators}{`0}
\DeclareMathSymbol{1}{\mathalpha}{operators}{`1}
\DeclareMathSymbol{2}{\mathalpha}{operators}{`2}
\DeclareMathSymbol{3}{\mathalpha}{operators}{`3}
\DeclareMathSymbol{4}{\mathalpha}{operators}{`4}
\DeclareMathSymbol{5}{\mathalpha}{operators}{`5}
\DeclareMathSymbol{6}{\mathalpha}{operators}{`6}
\DeclareMathSymbol{7}{\mathalpha}{operators}{`7}
\DeclareMathSymbol{8}{\mathalpha}{operators}{`8}
```

```
\DeclareMathSymbol{9}{\mathalpha}{operators}{`9}
```

Soit ϕ une fonction continue sur $[0, 1]$.

En modifiant légèrement cet exemple, on peut par exemple demander à LaTeX de mettre toutes les majuscules en mode mathématique droites, et non plus penchées.

```
\DeclareMathSymbol{A}{\mathalpha}{operators}{`A}%
\DeclareMathSymbol{B}{\mathalpha}{operators}{`B}%
\DeclareMathSymbol{C}{\mathalpha}{operators}{`C}%
\DeclareMathSymbol{D}{\mathalpha}{operators}{`D}%
\DeclareMathSymbol{E}{\mathalpha}{operators}{`E}%
\DeclareMathSymbol{F}{\mathalpha}{operators}{`F}%
\DeclareMathSymbol{G}{\mathalpha}{operators}{`G}%
\DeclareMathSymbol{H}{\mathalpha}{operators}{`H}%
\DeclareMathSymbol{I}{\mathalpha}{operators}{`I}%
\DeclareMathSymbol{J}{\mathalpha}{operators}{`J}%
\DeclareMathSymbol{K}{\mathalpha}{operators}{`K}%
\DeclareMathSymbol{L}{\mathalpha}{operators}{`L}%
\DeclareMathSymbol{M}{\mathalpha}{operators}{`M}%
\DeclareMathSymbol{N}{\mathalpha}{operators}{`N}%
\DeclareMathSymbol{O}{\mathalpha}{operators}{`O}%
\DeclareMathSymbol{P}{\mathalpha}{operators}{`P}%
\DeclareMathSymbol{Q}{\mathalpha}{operators}{`Q}%
\DeclareMathSymbol{R}{\mathalpha}{operators}{`R}%
\DeclareMathSymbol{S}{\mathalpha}{operators}{`S}%
\DeclareMathSymbol{T}{\mathalpha}{operators}{`T}%
\DeclareMathSymbol{U}{\mathalpha}{operators}{`U}%
\DeclareMathSymbol{V}{\mathalpha}{operators}{`V}%
\DeclareMathSymbol{W}{\mathalpha}{operators}{`W}%
\DeclareMathSymbol{X}{\mathalpha}{operators}{`X}%
\DeclareMathSymbol{Y}{\mathalpha}{operators}{`Y}%
\DeclareMathSymbol{Z}{\mathalpha}{operators}{`Z}%
```

Soit F une fonction continue sur $[0, 1]$ et n un entier.

En plain TeX, on était contraint d'écrire des choses bien plus illisibles (je mets ces lignes ici à titre purement historique, surtout pas pour qu'elles soient utilisées).

```
% The following is a well-known (dirty?) plain-TeX hack.
\mathcode`A"7041  \mathcode`B"7042  \mathcode`C"7043
\mathcode`D"7044  \mathcode`E"7045  \mathcode`F"7046
\mathcode`G"7047  \mathcode`H"7048  \mathcode`I"7049
\mathcode`J"704A  \mathcode`K"704B  \mathcode`L"704C
\mathcode`M"704D  \mathcode`N"704E  \mathcode`O"704F
\mathcode`P"7050  \mathcode`Q"7051  \mathcode`R"7052
\mathcode`S"7053  \mathcode`T"7054  \mathcode`U"7055
\mathcode`V"7056  \mathcode`W"7057  \mathcode`X"7058
\mathcode`Y"7059  \mathcode`Z"705A
% \mathcode`l"0160
```

Nous verrons un peu plus loin (9.35) comment faire la même chose avec des fontes virtuelles.

Le lecteur désirant d'autres exemples est invité à regarder comment les extensions `pslatex` ou `mathpple` sont programmées.

4.15 Comment utiliser une fonte mathématique différente?

Commençons par quelques révisions : nous avons déjà vu des commandes permettant de changer ponctuellement la fonte mathématique :

```
\mathnormal \mathit \mathbf \mathsf \mathrm \mathtt \mathcal \mathfrak
```

Nous avons aussi déjà vu comment changer la version mathématique, par exemple pour tout mettre en gras.

```
\mathversion{bold}
```

***** A REVOIR ***** (Cette question est très pertinente : si on désire unstaller une nouvelle fonte textuelle, ne comportant pas tous les caractères mathématiques dont on aura besoin, on peut créer des fontes mathématiques virtuelles à partir des caractères mathématiques présents dans la fonte et en allant chercher les caractères manquants dans les fontes COmputer Modern ou Euler.)

(ANCIENNE REPONSE — A EFFACER) La question de l'utilisation d'une autre fonte mathématique ne se pose pas vraiment, car il y en a très peu et elles sont toutes accompagnées d'un fichier `*.sty` permettant de les utiliser. Pour que le problème se pose, il faudrait que l'on écrive soi-même une telle fonte ou alors que l'on modifie soi-même certains des paramètres des fontes CM ou Euler correspondantes pour qu'elles se marrient bien avec une fonte texte que l'on aurait choisie, ou bien que l'on utilise des fontes virtuelles pour modifier une fonte mathématique existante, ou encore que l'on utilise des fontes virtuelles pour récupérer des caractères dans diverses fontes des symboles.

Le lecteur intéressé devrait trouver son bonheur dans les fichiers `euler.sty`, `mathpple` et `pslatex.sty`.

Nous allons quand même essayer d'expliquer comment on procèderait.

A FAIRE PRENDRE UN EXEMPLE : avec des fontes virtuelles (en modifiant tous les caractères d'une fonte mathématique postscript de la même manière) ou en changeant les paramètres des fontes CM.

Imaaginons que nous ayions écrit une nouvelle fonte mathématique, comportant tous les symboles dont on pourrait avoir besoin. Comment dire à LaTeX de l'utiliser? Nous prendrons l'exemple de la fonte `euler`, i.e., nous allons réécrire `euler.sty`.

Commençons par demander à LaTeX d'utiliser notre fonte pour les caractères textuels utilisés en mode mathématique. La première ligne définit un nouvel alphabet `letters` (plus exactement, on le redéfinit). Pour l'instant, il n'est définit que pour la version `normal`. Mais comme la fonte existe à la fois en maigre et en gras, on peut définir une nouvelle version, `bold`.

```
\DeclareSymbolFont{letters}{U}{eur}{m}{n}
\SetSymbolFont{letters}{bold}{U}{eur}{b}{n}
```

Texte X, $\hom X$, $1+1$, $\oplus$, α, Γ
Texte $\boldsymbol{X}$, $\boldsymbol{\hom X}$, $\boldsymbol{1+1}$, $\boldsymbol{\oplus}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Gamma}$

On constate que LaTeX va bien chercher les caractères alphabétiques dans cette fonte, mais pas les chiffres. (Ne me demandez pas pourquoi.) Il suffit donc de redéfinir ces caractères un par un.

```
\DeclareMathSymbol{0}\mathalpha{letters}{"30}
\DeclareMathSymbol{1}\mathalpha{letters}{"31}
\DeclareMathSymbol{2}\mathalpha{letters}{"32}
\DeclareMathSymbol{3}\mathalpha{letters}{"33}
\DeclareMathSymbol{4}\mathalpha{letters}{"34}
\DeclareMathSymbol{5}\mathalpha{letters}{"35}
\DeclareMathSymbol{6}\mathalpha{letters}{"36}
\DeclareMathSymbol{7}\mathalpha{letters}{"37}
\DeclareMathSymbol{8}\mathalpha{letters}{"38}
\DeclareMathSymbol{9}\mathalpha{letters}{"39}
```

Texte X, $\hom X$, $1+1$, $\oplus$, α, Γ
Texte $\boldsymbol{X}$, $\boldsymbol{\hom X}$, $\boldsymbol{1+1}$, $\boldsymbol{\oplus}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Gamma}$

Même remarque pour les caractères grecs : la fonte Euler, que nous aavons choisie pour cet exemple, redéfinit les majuscules (on garde les minuscules de Computer Modern).

```
\DeclareMathSymbol\Gamma  \mathord{letters}{"00}
\DeclareMathSymbol\Delta  \mathord{letters}{"01}
\DeclareMathSymbol\Theta  \mathord{letters}{"02}
\DeclareMathSymbol\Lambda \mathord{letters}{"03}
\DeclareMathSymbol\Xi     \mathord{letters}{"04}
\DeclareMathSymbol\Pi     \mathord{letters}{"05}
\DeclareMathSymbol\Sigma  \mathord{letters}{"06}
\DeclareMathSymbol\Upsilon\mathord{letters}{"07}
\DeclareMathSymbol\Phi    \mathord{letters}{"08}
\DeclareMathSymbol\Psi    \mathord{letters}{"09}
\DeclareMathSymbol\Omega  \mathord{letters}{"0A}
\let\varsigma=\sigma
\let\varrho=\rho
```

Texte X, $\hom X$, $1+1$, $\oplus$, α, Γ
Texte $\boldsymbol{X}$, $\boldsymbol{\hom X}$, $\boldsymbol{1+1}$, $\boldsymbol{\oplus}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Gamma}$

Il reste encore tous les autres symboles : ils sont dans une autre fonte, que l'on charge,

```
\DeclareSymbolFont{EulerFraktur}{U}{euf}{m}{n}
\SetSymbolFont{EulerFraktur}{bold}{U}{euf}{b}{n}
```

et dans laquelle on pioche les symboles, un par un.

```
\DeclareMathSymbol{!}\mathord  {EulerFraktur}{"21}
\DeclareMathSymbol{(}\mathopen {EulerFraktur}{"28}
\DeclareMathSymbol{)}\mathclose{EulerFraktur}{"29}
\DeclareMathSymbol{+}\mathbin  {EulerFraktur}{"2B}
\DeclareMathSymbol{-}\mathbin  {EulerFraktur}{"2D}
\DeclareMathSymbol{=}\mathrel  {EulerFraktur}{"3D}
\DeclareMathSymbol{[}\mathopen {EulerFraktur}{"5B}
\DeclareMathSymbol{]}\mathclose{EulerFraktur}{"5D}
\DeclareMathSymbol{"}\mathord  {EulerFraktur}{"7D}
\DeclareMathSymbol{&}\mathord  {EulerFraktur}{"26}
\DeclareMathSymbol{:}\mathrel  {EulerFraktur}{"3A}
\DeclareMathSymbol{;}\mathpunct{EulerFraktur}{"3B}
\DeclareMathSymbol{?}\mathord  {EulerFraktur}{"3F}
\DeclareMathSymbol{^}\mathord  {EulerFraktur}{"5E}
\DeclareMathSymbol{`}\mathord  {EulerFraktur}{"12}
\DeclareMathDelimiter{(}{EulerFraktur}{"28}{largesymbols}{"00}
\DeclareMathDelimiter{)}{EulerFraktur}{"29}{largesymbols}{"01}
\DeclareMathDelimiter{[}{EulerFraktur}{"5B}{largesymbols}{"02}
\DeclareMathDelimiter{]}{EulerFraktur}{"5D}{largesymbols}{"03}
```

il y a encore d'autres symboles dans une autre fonte.

```
\SetSymbolFont{EulerScript}{bold}{U}{eus}{b}{n}
\DeclareSymbolFontAlphabet\mathscr{EulerScript}
\DeclareMathSymbol\aleph           \mathord{EulerScript}{"40}
\DeclareMathSymbol\Re              \mathord{EulerScript}{"3C}
\DeclareMathSymbol\Im              \mathord{EulerScript}{"3D}
\DeclareMathDelimiter\vert         \mathord{EulerScript}{"6A}{largesymbols}{"0C}
\DeclareMathDelimiter\backslash    \mathord{EulerScript}{"6E}{largesymbols}{"0F}
\DeclareMathSymbol{|}              \mathord{EulerScript}{"6A}
\DeclareMathDelimiter{|}{EulerScript}{"6A}{largesymbols}{"0C}
\DeclareMathSymbol\neg             \mathord{EulerScript}{"3A}
\let\lnot=\neg
```

```
\DeclareMathSymbol\wedge           \mathbin{EulerScript}{"5E}
\let\land=\wedge
\DeclareMathSymbol\vee             \mathbin{EulerScript}{"5F}
\let\lor=\vee
\DeclareMathSymbol\setminus        \mathbin{EulerScript}{"6E}
\DeclareMathSymbol\sim             \mathrel{EulerScript}{"18}
\DeclareMathSymbol\mid             \mathrel{EulerScript}{"6A}
\DeclareMathDelimiter\arrowvert    \mathord{EulerScript}{"6A}{largesymbols}{"3C}
\DeclareMathSymbol\mathsection     \mathord{EulerScript}{"78}
```

Et encore une...

```
\DeclareSymbolFont{EulerExtension}{U}{euex}{m}{n}
\DeclareMathSymbol\coprod          \mathop {EulerExtension}{"60}%
\DeclareMathSymbol\prod            \mathop {EulerExtension}{"51}%
\DeclareMathSymbol\sum             \mathop {EulerExtension}{"50}
\DeclareMathSymbol\intop           \mathop {EulerExtension}{"52}
\DeclareMathSymbol\ointop          \mathop {EulerExtension}{"48}
\DeclareMathSymbol\braceld         \mathord{EulerExtension}{"7A}
\DeclareMathSymbol\bracerd         \mathord{EulerExtension}{"7B}
\DeclareMathSymbol\bracelu         \mathord{EulerExtension}{"7C}
\DeclareMathSymbol\braceru         \mathord{EulerExtension}{"7D}
\DeclareMathSymbol\infty           \mathord{EulerExtension}{"31}
\DeclareMathSymbol\nearrow         \mathrel{EulerExtension}{"25}
\DeclareMathSymbol\searrow         \mathrel{EulerExtension}{"26}
\DeclareMathSymbol\nwarrow         \mathrel{EulerExtension}{"2D}
\DeclareMathSymbol\swarrow         \mathrel{EulerExtension}{"2E}
\DeclareMathSymbol\Leftrightarrow  \mathrel{EulerExtension}{"2C}
\DeclareMathSymbol\Leftarrow       \mathrel{EulerExtension}{"28}
\DeclareMathSymbol\Rightarrow      \mathrel{EulerExtension}{"29}
\DeclareMathSymbol\leftrightarrow  \mathrel{EulerExtension}{"24}
\DeclareMathSymbol\leftarrow       \mathrel{EulerExtension}{"20}
\let\gets=\leftarrow
\DeclareMathSymbol\rightarrow      \mathrel{EulerExtension}{"21}
\let\to=\rightarrow
\DeclareMathDelimiter\uparrow      \mathrel{EulerExtension}{"22}{largesymbols}{"78}
\DeclareMathDelimiter\downarrow    \mathrel{EulerExtension}{"23}{largesymbols}{"79}
\DeclareMathDelimiter\updownarrow  \mathrel{EulerExtension}{"6C}{largesymbols}{"3F}
\DeclareMathDelimiter\Uparrow      \mathrel{EulerExtension}{"2A}{largesymbols}{"7E}
\DeclareMathDelimiter\Downarrow    \mathrel{EulerExtension}{"2B}{largesymbols}{"7F}
\DeclareMathDelimiter\Updownarrow  \mathrel{EulerExtension}{"6D}{largesymbols}{"77}
\DeclareMathSymbol\leftharpoonup   \mathrel{EulerExtension}{"18}
\DeclareMathSymbol\leftharpoondown \mathrel{EulerExtension}{"19}
\DeclareMathSymbol\rightharpoonup  \mathrel{EulerExtension}{"1A}
\DeclareMathSymbol\rightharpoondown\mathrel{EulerExtension}{"1B}
\DeclareMathDelimiter\lbrace       \mathopen{EulerScript}{"66}{EulerExtension}{"08}
\DeclareMathDelimiter\rbrace       \mathclose{EulerScript}{"67}{EulerExtension}{"09}
```

Et il y a encore tous les symboles qui proviennent des fontes Computer Modern

```
\DeclareSymbolFont{cmmigroup}{OML}{cmm}{m}{it}
\SetSymbolFont{cmmigroup}{bold}{OML}{cmm}{b}{it}
\DeclareMathAccent\vec          \mathord{cmmigroup}{"7E}
\DeclareMathSymbol\triangleleft \mathbin{cmmigroup}{"2F}
\DeclareMathSymbol\triangleright\mathbin{cmmigroup}{"2E}
\DeclareMathSymbol\star         \mathbin{cmmigroup}{"3F}
\DeclareMathSymbol\lhook        \mathrel{cmmigroup}{"2C}
\DeclareMathSymbol\rhook        \mathrel{cmmigroup}{"2D}
\DeclareMathSymbol\flat         \mathord{cmmigroup}{"5B}
```

```
\DeclareMathSymbol\natural      \mathord{cmmigroup}{"5C}
\DeclareMathSymbol\sharp        \mathord{cmmigroup}{"5D}
\DeclareMathSymbol\smile        \mathrel{cmmigroup}{"5E}
\DeclareMathSymbol\frown        \mathrel{cmmigroup}{"5F}
```

Voilà : pour ce qui est des symboles, c'est fini. Il ne reste plus qu'à redéfinir la fonte utilisée par les opérateurs (`\cos`, etc.) et le comportement des commandes du genre `\mathbf` : on va juste prendre les mêmes que pour le texte.

```
\DeclareSymbolFont{operators}{T1}\rmdefault\mddefault\updefault
\SetSymbolFont{operators}{bold}{T1}\rmdefault\bfdefault\updefault
\DeclareMathAlphabet\mathbf{T1}\rmdefault\bfdefault\updefault
\DeclareMathAlphabet\mathsf{T1}\sfdefault\mddefault\updefault
\DeclareMathAlphabet\mathit{T1}\rmdefault\mddefault\itdefault
\DeclareMathAlphabet\mathtt{T1}\ttdefault\mddefault\updefault
\SetMathAlphabet\mathsf{bold}{T1}\sfdefault\bfdefault\updefault
\SetMathAlphabet\mathit{bold}{T1}\rmdefault\bfdefault\itdefault
```

```
\DeclareMathAlphabet\mathbf{U}{eur}{b}{n}}
\DeclareSymbolFontAlphabet\mathfrak{EulerFraktur}
```

Pour le gras, on peut quand même prendre le gras mathématique.

```
\DeclareMathAlphabet\mathbf{U}{eur}{b}{n}}
```

Pour les accents mathématiques : la plupart du temps, on va les chercher dans la fonte texte.

```
\DeclareMathAccent\grave\mathalpha{operators}{"00}
\DeclareMathAccent\acute\mathalpha{operators}{"01}
\DeclareMathAccent\tilde\mathalpha{operators}{"03}
\DeclareMathAccent\ddot \mathalpha{operators}{"04}
\DeclareMathAccent\check\mathalpha{operators}{"07}
\DeclareMathAccent\breve\mathalpha{operators}{"08}
\DeclareMathAccent\bar  \mathalpha{operators}{"09}
\DeclareMathAccent\dot  \mathalpha{operators}{"0A}
\DeclareMathAccent\hat  \mathalpha{EulerFraktur}{"5E}
```

De manière surprenante, il y a certains symboles mathématiques qui se trouvent dans la fonte texte.

```
\DeclareMathSymbol\euler@hbar\mathord{operators}{"09}
\def\hbar{{\euler@hbar\mkern-8muh}}
```

C'est à peu près tout.

Texte X, $\hom X$, $1+1$, α, Γ
Texte $\boldsymbol{X}$, hom $\boldsymbol{X}$, $\boldsymbol{1+1}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Gamma}$

Précisons quand même qu'il faut choisir la fonte texte avec soin (ci-dessous, `concrete`, par opposition à `computer modern` ci dessus).

Texte X, $\hom X$, $1+1$, α, Γ
Texte $\boldsymbol{X}$, hom $\boldsymbol{X}$, $\boldsymbol{1+1}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Gamma}$

(REPONSE ENCORE PLUS ANCIENNE — A EFFACER) Tout cela, c'était pour les fontes « normales », pas celles contenant des symboles.

```
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}

\DeclareSymbolFont{operators}   {OT1}{ptmcm}{m}{n}
\DeclareSymbolFont{letters}     {OML}{ptmcm}{m}{it}
```

```
\DeclareSymbolFont{symbols}     {OMS}{pzccm}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{psycm}{m}{n}
\DeclareSymbolFont{bold}        {OT1}{ptm}{bx}{n}
\DeclareSymbolFont{italic}      {OT1}{ptm}{m}{it}
```

Comme précédemment, il y une autre commande qui permet de définir la même chose dans une version.

```
\DeclareSymbolFont{cyrletters}{LWN}{cmr}{m}{n}
\SetSymbolFont{cyrletters}{bold}{LWN}{cmr}{bx}{n}
```

Pour défininir une commande qui permet d'utiliser ces fontes :

```
\DeclareSymbolFontAlphabet{\mathbb}{AMSb}
```

Ensuite, on peut définir des caractères un par un.

```
\DeclareMathSymbol{\omicron}{0}{operators}{`\o}
\DeclareMathSymbol\rightleftharpoons{\mathrel}{AMSa}{"0A}
\DeclareMathSymbol{\varGamma}{\mathord}{letters}{"00}

\begingroup \catcode`\"=12
\DeclareMathDelimiter\ulcorner{\mathopen} {AMSa}{"70}{AMSa}{"70}
\DeclareMathDelimiter\urcorner{\mathclose}{AMSa}{"71}{AMSa}{"71}
\DeclareMathDelimiter{(}{EulerFraktur}{"28}{largesymbols}{"00}
\DeclareMathDelimiter{)}{EulerFraktur}{"29}{largesymbols}{"01}

\ifCorkEncoding
   \DeclareMathAccent\grave\mathalpha{operators}{"00}
   \DeclareMathAccent\acute\mathalpha{operators}{"01}
   \DeclareMathAccent\tilde\mathalpha{operators}{"03}
   \DeclareMathAccent\ddot \mathalpha{operators}{"04}
   \DeclareMathAccent\check\mathalpha{operators}{"07}
   \DeclareMathAccent\breve\mathalpha{operators}{"08}
   \DeclareMathAccent\bar  \mathalpha{operators}{"09}
   \DeclareMathAccent\dot  \mathalpha{operators}{"0A}
\fi
\DeclareMathAccent\hat\mathalpha{EulerFraktur}{"5E}

\DeclareMathRadical{\sqrt}{symbols}{"70}{largesymbols}{#70}
```

A FAIRE : il faudra parler de `\mathbin`, `mathrel`, etc. (mais plus haut...)

Il y a encore une autre chose que l'on peut régler : la taille des caractères en mode mathématique. La ligne suivante demande que quand le texte est en 12 points, le texte mathématique normal (`\displaystyle` et `\textstyle`) soit en 12 point, les indices (`\scriptstyle`) en 9 point et les indices d'indices (`\scriptscriptstyle`) en 7 points.

```
\DeclareMathSizes{12}{12}{9}{7}
```

A FAIRE : exemples (exemple normal ; exemple avec une autre version ; tt.sty)

4.16 Comment (et pourquoi) utiliser différentes versions mathématiques ?

On peut vouloir utiliser plusieures versions mathématiques (i.e., plusieurs jeux de fontes mathématiques) car ils apparaissent à des endroits différents : par exemple dans le texte (version `normal`) et dans les titres, en gras (version `bold`). C'est l'utilisation la plus courrante, et ces deux versions sont déjà définies — il n'y a donc rien d'autre à faire que de les utiliser, à l'aide des commandes `\mathversion{bold}` et `\mathversion{normal}`.

On utilise aussi différentes versions quand on désire comparer, dans un même document, différentes fontes mathématiques : on peut ainsi mettre une page en Mathtimes, une autre en Computer Modern et une dernière en Lucida. Il faut alors créer soi-même de nouvelles versions mathématiques. Mais cette utilisation reste marginale. Voir par exemple Sur la diversité des fontes mathématiques (Thierry Bouche)[4]

A FAIRE : pour que tout le document soit en gras

```
\renewcommand{\seriesdefault}{bx}
\mathversion{bold}
```

A FAIRE : parler de boldsymbol et bm.

4.17 Quelles sont les différentes dimensions présentes dans une fonte ? Comment les modifier ?

- `\fontdimen 1` : inclinaison de l'italique
- `\fontdimen 2` : espace entre les mots
- `\fontdimen 3` : étirement de l'espace entre les mots
- `\fontdimen 4` : contraction de l'espace entre les mots
- `\fontdimen 5` : hauteur du `x` minuscule
- `\fontdimen 6` : quadratin
- `\fontdimen 7` : espace après un point.
- `\fontdimen 8` : hauteur des capitales
- `\fontdimen 9` : hauteur des lettres « qui montent » (comme le `l`)
- `\fontdimen 10` : hauteur des capitales accentuées
- `\fontdimen 11` : profondeur des lettres « qui descentent » (comme le `p`)
- `\fontdimen 12` : hauteur maximale
- `\fontdimen 13` : profondeur maximale
- `\fontdimen 14` : largeur des chiffres
- `\fontdimen 15` : cap_stem ???
- `\fontdimen 16` : espace entre les lignes (`baselineskip`, par exemple 12 points pour une fonte à 10 points)
- `\fontdimen 17` et après : uniquement pour les fontes mathématiques.

On peut modifier les dimensions des fontes mathématiques, par exemple, afin de changer la position des indices ou exposants. (À FAIRE)

4. http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/25-bouche.pdf

Chapitre 5

Au delà des caractères latins

5.1 Comment utiliser l'alphabet phonétique ?

Il suffit d'utiliser l'extension `tipa`.

```
\documentclass[12pt,a4paper]{demo}
\usepackage[T1]{tipa}
\begin{document}
\textipa{/""saI@ns "fIkS@n/}
\textipa{/pjU@\super r/}
\textipa{/b2t/}
\textipa{/""Z@ n@ seI "kwA:/}
\textipa{/l\ae pIs ""l\ae zjUlI/}
\textipa{/P/}% Interruption glottale
\end{document}
```

/ˌsaɪəns ˈfɪkʃən/
/pjʊə^{r}/
/bʌt/
/ˌʒə nə seɪ ˈkwɑː/
/læpɪs ˌlæzjʊlɪ/
/ʔ/

5.2 Comment écrire des partitions musicales ?

Regardez MusixTeX [1] et Lilypond [2].

5.3 Comment écrire quelques mots en russe ? Comment utiliser un caractère russe en mode mathématique ?

C'est bien expliqué dans `cyrguide.dvi`. Ici, nous allons simplement expliquer comment utiliser des symboles russes, de manière ponctuelle, en mode mathématique. Les fontes en question, `wncyr*.mf`, comptent parmi celles de l'AMS : elles sont probablement déjà là. Il y a néanmoins un problème supplémentaire : le codage d'entrée est différent. Heureusement, les fontes viennent avec un fichier décrivant ce codage, `cyracc.def`.

```
\input cyracc.def
\DeclareFontFamily{U}{russian}{}
\DeclareFontShape{U}{russian}{m}{n}
        { <5><6> wncyr5
```

1. `http://www.loria.fr/services/tex/packages.html##musique`
2. `http://www.cs.ruu.nl/people/hanwen/lilypond/index.html`

```
        <7><8><9> wncyr7
        <10><10.95><12><14.4><17.28><20.74><24.88> wncyr10 }{}
\DeclareSymbolFont{Russian}{U}{russian}{m}{n}
\DeclareSymbolFontAlphabet{\mathcyr}{Russian}
\makeatletter
\let\@math@cyr\mathcyr
\renewcommand{\mathcyr}[1]{\@math@cyr{\cyracc #1}}
\makeatother
\newcommand{\ShGr}{\mathcyr{SH}} % Le groupe de Shafarewicz
\newcommand{\textcyr}[1]{%
  {\fontencoding{U}\fontfamily{russian}\selectfont#1}}
```

Notons Ш le groupe de Shafarewicz.

5.4 Comment écrire quelques mots en grec ancien?

Le plus simple est probablement d'utiliser `babel` avec l'option `greek`. Il n'y a aucune raison pour que Babel soit déjà installé pour la langue grecque. Nous allons procéder comme suit.

- Installer les fontes de `fonts/greek/cb/`
- Prendre les fichiers nécessaires de `fonts/greek/package-babel` (c'est à dire `fonts/greek/package-babel/hyphenation/*` et `fonts/greek/package-babel/encodings/*`)
- Dans `macros/latex/required/babel/`, faire compiler, avec `tex`, les fichiers `gr*.ins` (en profiter pour installer la dernière version de Babel, si elle n'est pas déjà là – si elle était déjà là, le fichier de césure était probablement aussi déjà là et il n'était pas nécessaire de le recréer).
- Rajouter une ligne dans le fichier `language.dat`

  ```
  greek          grhyph.tex
  ```

- Recréer les formats. Pour certaines distributions (`tetex`), il suffit de taper

  ```
  fmtutil --all
  ```

  Pour les autres, il suffit d'aller dans le répertoire `web2c` et de taper

  ```
  tex --ini --mltex latex.ltx
  tex --ini --mltex plain
  ```

  (Si vous utilisez d'autres choses que `tex`, par exemple `etex`, `pdftex` ou `omega`, vous êtes sensé savoir recompiler les formats.)
- Vérifier que cela marche en tentant de compiler les fichiers de `fonts/greek/`

5.5 Pourquoi n'y a-t-il pas de béta interne?

Car il n'est pas utilisé outre-atlantique. Si vous en avez vraiment besoin, voici plusieures solutions :

- Ce caractère existe dans les fontes d'Omega.

– Il y a quelques années, j'en ai eu besoin pour le journal mathématique de l'ENS de Cachan[3] et je l'avais dessiné : vous pouvez le récupérer sur la page web[4] du journal.

ἡ ὕβρις

– E. Riga en a dessiné un autre.

```
cmchar "Inner lowercase Greek beta";
beginchar(oct"026",9u#,asc_height#,0);
italcorr .5[bar_height#,x_height#]*slant+.5vair#+.5u#;
adjust_fit(0,0); pickup fine.nib;
numeric light_curve; light_curve=hround .5[stem,curve];
pos1(hair,0); pos2(.5(stem+hair),-90);
 x1-.5hair=hround -.5hair; x2=.5w-u;
 y1=x_height-x_height/3.141592653589793; top y2l=x_height;
pos3(stem,-180); y3=.4x_height; rt x3l=rt x7l=hround(w-1.5u+.5light_curve);
pos4(vair,90); x4=x6-u=.5[x3,x5]; bot y4r=-oo;
pos5(light_curve,0); x5l=u;  y5=.8x_height;
pos6(vair,-90); top y6r=h+oo;
pos7(stem,-180); y7=.65[x_height,h];
pos8(vair,-315); y8=.25[x_height,h]; x8=x1;
 filldraw stroke z1e{x2-x1,3.14159(y2-y1)}...z2e
    & super_arc.e(2,3) & pulled_arc.e(3,4)
   & pulled_arc.e(4,5) & pulled_arc.e(5,6)
   & super_arc.e(6,7) & super_arc.e(7,8);
penlabels(1,2,3,4,5,6,7,8); endchar;
```

ἡ ὕβρις

5.6 Comment écrire quelques mots en japonais ?

Je suggère d'utiliser CJK[5]. D'autres solutions conviennent mieux à un environement entièrement japonais ; mais ce que nous voulons ici, c'est écrire des documents qui mélangent du japonais et des langues occidentales, ou écrire quelques documents en japonais au milieu de plein d'autres en langues européennes. De surcroît, CJK est documenté en anglais, ce qui est souhaitable si la personne qui va l'installer n'est pas celle qui va l'utiliser.

CJK est juste un ensemble de macros pour utiliser les langues asiatiques sous LaTeX. Il ne contient pas de fontes. Pour le japonais, j'ai essayé de deux manières différentes.

– La manière facile, avec des fontes Postscript (que l'on peut donc utiliser à n'importe quelle taille sans aucun problème de pixellation).

  Procéder ainsi : récupérer les fontes Wadalab (on les trouve souvent à côté de Ghostscript[6] et lire `DNP.doc`. Les instructions sont suffisemment détaillées et claires (je signale toutefois que j'ai eu un problème pour appliquer un patch : il avait visiblement été corrompu lors d'un passage DOS/Unix et j'ai dû le réparer avec `recode`).

日本語が書けるよ。

日本語が書けるよ。

3. `http://www.dptmaths.ens-cachan.fr/eleves/Hubris/`
4. `http://www.dptmaths.ens-cachan.fr/eleves/Hubris/Article/exemple.tar.gz`
5. `http://cjk.ffii.org/`
6. `ftp://ftp.lip6.fr/pub/ghost/kanji`

日本語が書けるよ。

– La manière compliquée, utilisant des fontes bitmap (que l'on ne peut donc pas grossir arbitrairement).
  Je copie textuellement ce que j'ai dans mes notes quant à son installation, sans plus d'explications. Ce n'est ni très simple, ni très court. Pour le chinois ou le coréen, c'est pareil, mais la fonte ne s'appelle plus `kanji48.tar.gz`.

```
tar zxvf CJK.tar.gz

cd CJK/4_1.3/
less doc/japanese/japanese.doc
gzip -d **/*.gz
cp -r texinput/ /usr/local/lib/texmf/tex/CJK
cd utils/hbf2gf
make all OS=unix
cp hbf2gf /usr/local/bin/
cp hbf2gf.1 /usr/local/man/man1/

cd /usr/local/lib ; tar zxvf kanji48.tar.gz
less CJK/4_1.3/doc/hbf2gf.doc
which MakeTeXPK
mv /usr/local/lib/texmf/bin/scripts//MakeTeXPK /usr/local/lib/texmf/bin/scripts//MakeTeXPK.french
cd CJK/4_1.3/
cp utils/hbf2gf/MakeTeXPK /usr/local/lib/texmf/bin/scripts/MakeTeXPK
cd /usr/local/lib/texmf/hbf2gf/
TEXMF=/usr/local/lib/texmf/ hbf2gf jsso12.cfg
sh jsso12
cd /usr/local/lib/texmf/fonts/tfm
mv japanese/**/*(.) ./
vi /usr/local/lib/texmf/web2c/texmf.cnf
  Changer la première ligne et rajouter :$pkdir/CJK à PKFONTS.xdvi et PKFONTS

cd CJK/4_1.3/examples
latex JIS.tex
HBF2GFDIR=/usr/local/lib/texmf/hbf2gf/ TEXMF=/usr/local/lib/texmf xdvi JIS.dvi
  On vérifie que ça marche.
  (MakeTeXPK a besoin de la variable HBF2GFDIR pour savoir où se trouvent les
  fichiers de configuration de hbf2gf, il faudrait la rajouter dans les
  divers fichiers de configuration du shell, ~/.export.)
perl -p -i -e 's|\$TEXMF|/usr/local/lib/texmf|' /usr/local/lib/texmf/hbf2gf/*.cfg

rm -f /usr/local/lib/texmf/fonts/tmp/pk/modeless/*
cd CJK/4_1.3/examples
latex JIS.tex
HBF2GFDIR=/usr/local/lib/texmf/hbf2gf/ xdvi JIS.dvi
```

日本語が書けるよ！

5.7 Comment écrire quelques mots en Chinois? en Coréen?

Vous pouvez utiliser CJK, comme suggéré dans la question précédente. Je ne connais pas du tout ces langues, je ne donne pas d'exemple.

5.8 Comment inclure quelques mots dans une autre langue ?

Je ne sais pas.

- Regarder si cette langue est reconnue par `babel`.
- Regarder dans CTAN/languages/ [7]
- Regarder dans CTAN/fonts/ [8]
- S'il s'agit de la langue principale du document, on peut se renseigner auprès du groupe d'utilisateurs de TeX des pays concernés.

A FAIRE : donner quelques exemples marrants (hiéroglyphes, klingon, elfique).

5.9 Qu'est-ce qu'Unicode ? Comment utilise-t-on Omega ?

Les codages (d'entrée) sont très variés : il en faut généralement un par langue, ce qui rend très pénible la création de documents multilingues. Unicode [9] est un jeu de caractères sensé couvrir toutes les langues actuelles.

Néanmoins, Unicode soulève certaines critiques, du même genre que l'absence du oe dans le codage latin1. Ainsi, Unicode ne distingue pas entre des caractères japonais et chinois de même origine. Par contre, il distingue le A majuscule latin du Alpha majuscule...

Unicode n'est qu'un *jeu* de caractères : il y a plusieurs codages possibles, le plus courrant étant UTF-8 (les caractères ASCII sont inchangés, les autres sont codés sur 2 octets, ou plus).

Précisons le vocabulaire

UCS Universal Character Set : jeu de caractères (31 bits) qui vise à couvrir toutes les langues « connues » (y compris les hiéroglyphes ou le Klingon).

ISO 10646 Voir UCS.

BMP Basic Multilingual Plane : Un morceau de UCS (16 bits), comprenant les langues actuellement utilisées.

ISO 10646-1 Voir BMP.

U+0041 Caractère numéro `0041` (en hexadécimal) d'UCS (il s'agit d'un `Latin capital letter A`).

ISO 8859-1 Codage Latin 1, utilisé pour les langues d'Europe de l'Ouest (il manque toutefois les trois caractères oe OE et

JIS X 0221-1995 Unicode tel qu'il est utilisé au Japon

KS X 1005-1:1995 Unicode tel qu'il est utilisé en Corée

GB 13000.1-93 Unicode tel qu'il est utilisé en Chine

Unicode Il y avait initialement deux tentatives de créer un je de caractères exhaustif : Unicode et ISO 10646-1. Elles sont toujours disjointes, mais compatibles. Ainsi, Unicode 3 et `ISO 10646-1:2000` coïncident.

UCS-2 Codage d'UCS sur 2 octets : pour transformer un caractère ASCII en UCS-2, il suffit de rajouter un `0x00` devant (certains logiciels n'aiment pas, car ils utilisent `0x00` pour indiquer la fin des chaînes de caractères).

UCS-4 Codage d'UCS sur 4 octets : pour transformer un caractère ASCII en UCS-2, il suffit de rajouter trois 0x00 devant (certains logiciels n'aiment pas).

UTF-8 Codage d'UCS sur un à six octets (en pratique, ça ne dépasse pas trois).

Omega [10] est une implémentation de TeX qui reconnait naturellement Unicode. On le lance en tapant `omega` (si on utilise Plain TeX) ou `lambda` (si on utilise LaTeX). Les programmes `xdvi` et `dvips` sont remplacés par `oxdvi` et `odvips`. Pour l'utilisateur, il n'y a rien de changé. Ce n'est pas obligatoire, mais on peut taper le texte en Unicode (tous les éditeurs de texte ne reconnaissent pas encore Unicode). Voici un exemple avec du grec (je ne parle pas du tout cette langue — l'exemple n'est donc pas de moi) mais pas Unicode.

```
\documentclass{article}

\usepackage{omega}[1999/06/01 New Omega macros]
\usepackage{epsfig}
```

7. `ftp://ftp.lip6.fr/pub/TeX/CTAN/languages/`
8. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts//`
9. `http://www.cl.cam.ac.uk/~mgk25/unicode.html`
10. `http://www.gutenberg.eu.org/omega/`

```
\background{usenglish}
\load[beta=twoform,
      accents=polytonic,
      numbersix=oldstigma,
      digamma=alt,
      apostrophe=latin
     ]{greek}

\begin{document}

Here's some english text.  The following dinner invitation from Oxyrynchos
is entered with the 8859-7 character set.

\begin{greek}
\begin{quote}
>Fåñùô=á|A FóåA FÍéêåöüñïòA Fäåé-\\A
Fðí=çóáéA Få>éòA FêëåßíçíA Fôï=õ\\A
FêõñßïõA F□́áñÜðéäïòA >FåíA Fô=ù|\\A
\end{quote}
\end{greek}

\end{document}
```

Si vous utilisez Unicode (UTF-8), vous pouvez procéder ainsi (sans jamais avoir à préciser de quelle langue il s'agit).

```
\ocp\TexUTF=inutf8
\InputTranslation currentfile \TexUTF
\documentclass[a4paper,12pt]{article}
\usepackage{omega}
\begin{document}
...
\end{document}
```

Chapitre 6

Métafont

6.1 Qu'est-ce que métafont ?

C'est un langage de description de fontes et un programme, écrit par D. Knuth, pour être utilisé avec (La)TeX. C'est un vrai langage de programmation, dont l'utilisation est décrite dans le Metafont book[1] Vous pouvez aussi regarder la documentation accompagnant MétaPost[2].

Nous avons déjà vu un exemple de fichier Métafont plus haut (3.2)

6.2 Comment utiliser Métafont ? Comment visualiser des fichiers Métafont ?

Il suffit généralement de l'invoquer avec en argument le nom du fichier que l'on désire compiler.

```
mf toto.mf
```

Si cela ne marche pas, vous pouvez essayer

```
mf '&cmbase \mode=localfont; \mag=1.0; input toto'
```

Cette fois-ci, s'il râle encore, tapez `q` puis Entrée (et croisez les doigts).

Généralement, vous n'avez pas un seul fichier Métafont, mais une ribambelle (parfois un par lettre). Il ne faut pas les compiler tous : ils ne sont pas faits pour être compilés de manière isolée, mais pour l'être ensemble. Il y a un (ou plusieurs) fichiers Métafont qui va charger tous les autres : généralement ceux dont le nom se termine par un chiffre (c'est à peu près la même différence qu'entre les fichiers `*.c` et `*.h` si vous programmez en C).

```
for fichier in *[0-9].mf
do
  mf '&cmbase \mode=localfont; \mag=1.0; input '`basename &dollar;fichier`
done
```

Il est possible de visualiser le résultat à l'aide de la commande `gftodvi`.

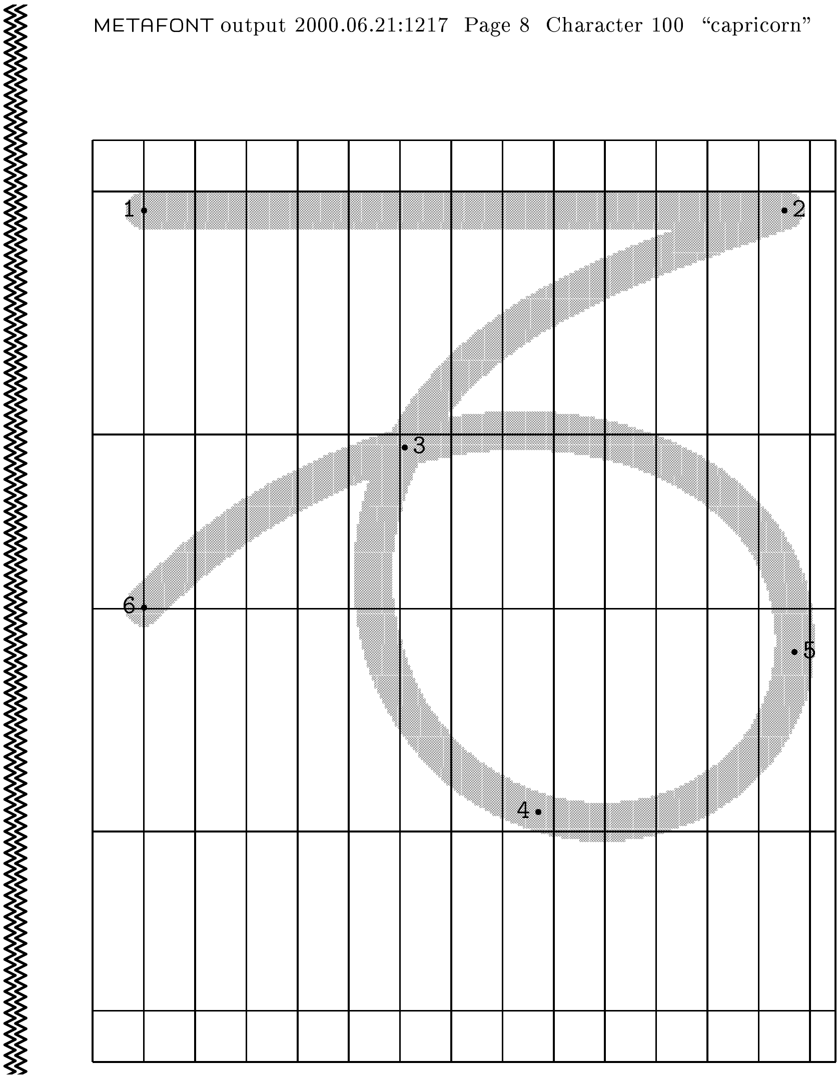
```
for fichier in *gf
do
  gftodvi &dollar;fichier
done
for fichier in *dvi
do
  echo On essaye $fichier
  xdvi &dollar;fichier
done
```

1. `ftp://ftp.lip6.fr/pub/TeX/CTAN/systems/knuth/mf/mfbook.tex.gz`
2. `http://cm.bell-labs.com/who/hobby/MetaPost.html`

Je ne sais pas si tout le monde est comme moi, mais à chaque fois que j'utilise `gftodvi` et que j'essaie de visualiser le résultat, `xdvi` a des problèmes avec la fonte `gray`. Il y a quelques années, il se contentait de ne pas la trouver (j'utilisais visiblement une distribution de LaTeX un peu trop minimaliste). Récemment, j'ai constaté qu'il y avait plusieurs fichiers contenant ce genre de fonte : un pour chaque type d'imprimante. Le fichier `gray.mf` va charger celui pour les imprimantes `cx`, alors que `xdvi` a lancé `metafont` pour une imprimante `ljfour`. Pour que cela marche quand même, on peut procéder ainsi :

```
mf '\mode:=ljfour; mag:=1+0/600; nonstopmode; input graylj'
rename 's/^graylj/gray/' gray*
gftopk *gf
```

Voici le résultat final. L'image au format GIF (celle que vous voyez si vous regardez ce documen en HTML) est très laide et n'est pas un reflet fidèle du résultat de `gftodvi` : normalement, le trait est d'un beau gris et laisse voir les points de contrôle.



6.3 Comment réaliser un catalogue de fontes Métafont ?

La situation est la suivante : vous avez récupéré quelques dizaines de fontes au format Métafont et vous aimeriez en choisir une pour votre prochain document. Pour cela, il faudrait pouvoir les visualiser toutes en même temps. Comment faire ?

La première solution consiste à regarder si le catalogue [3] que l'ai réalisé en 1998 n'est pas devenu trop obsolète.

Une autre solution consiste à utiliser le même script [4] que moi.(Attention, il ne marche pas toujours très bien : restez à côté et surveillez-le.)

3. `http://www.math.jussieu.fr/~zoonek/LaTeX/MF/liste.html`
4. `http://www.math.jussieu.fr/~zoonek/LaTeX/scripts/FONTES.html.pl`

6.4 Comment utiliser une fonte Métafont de manière ponctuelle dans un document LaTeX ?

Après avoir récupéré (ou écrit) une fonte, vous ne disposez que de fichiers Métafont. LaTeX a aussi (surtout) besoin des métriques : si votre version de LaTeX n'est pas trop récente, il faut les créer vous-même. On peut procéder ainsi. C'est Métafont qui crée les métriques en même temps que les fichiers `*.gf`.

```
for fichier in *[0-9].mf
do
  mf '&cmbase \mode=localfont; \mag=1.0; input '`basename &dollar;fichier`
  rm -f *.gf
done
```

Comme précédemment, si métafont râle et s'arrête, tapez `q` puis `entrée`.

Si vous désirez installer ces fontes, vous pouvez mettre les fichiers `*.mf` quelque part dans dans `$TEXMF/fonts/` ou `$TEXMF/mf/` (???) et les métriques `*.tfm` dans `$TEXMF/fonts/tfm/`. (L'arborescence est détaillée dans le fichier `$TEXMF/web2c/texmf.cnf` (voir 3.4) Mais si les fichiers sont dans le répertoire courrant, LaTeX et Métafont vont les trouver sans problème.

Il faut maintenant utiliser ces fontes. Premier cas : il y a une extension, ie, un fichier `*.sty`, qui accompagne la fonte. Il définit des commandes permettant de l'utiliser. Il faut mettre ce fichier, ainsi qu'un éventuel fichier `*.fd`, là où LaTeX saura le trouver, par exemple dans `$TEXMF/tex/latex/local/`.

Mais souvent, ce fichier n'existe pas : il faut le créer (voir 4.6). Un exemple valant mieux qu'un long discours, supposons que nous ayons récupéré des fichiers `toto5.mf`, `toto7.mf`, `toto10.mf`, correspondant respectivement aux dessins des caractères en 5 points, 7 points et 10 points. (Les fichiers *.mf dont le nom ne se termine pas par un chiffre sont généralement des fichiers inclus par les trois fichiers précédents et ne sont pas utilisables seuls.) Les lignes suivantes définissent des commandes `\texttutu` et `\mathtutu` pour utiliser cette fonte en mode texte et en mode mathématique respectivement (voir 4.13).

```
\DeclareFontFamily{U}{tutu}{}
\DeclareFontShape{U}{tutu}{m}{n}
        { <5><6> toto5
          <7><8><9> toto7
          <10><10.95><12><14.4><17.28><20.74><24.88> toto10
        }{}
\DeclareSymbolFont{bahhhhh}{U}{tutu}{m}{n}
\DeclareSymbolFontAlphabet{\mathtutu}{tutu}
\newcommand{\texttutu}[1]
{{\fontencoding{U}\fontfamily{tutu}\selectfont #1}}
```

Si on tient à faire les choses proprement, on peut créer deux fichiers, le premier, `toto.sty`, appelé explicitement par l'utilisateur par la commande `\usepackage`, contenant

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{toto}
\newcommand{\texttutu}[1]
{{\fontencoding{U}\fontfamily{tutu}\selectfont #1}}
\DeclareSymbolFont{tutu}{U}{tutu}{m}{n}
\DeclareSymbolFontAlphabet{\mathtutu}{tutu}
```

Le second, `ututu.fd` (le `u` initial est le codage), appelé automatiquement par LaTeX s'il en a besoin, contenant

```
\ProvidesFile{ututu.fd}
\DeclareFontFamily{U}{tutu}{}
\DeclareFontShape{U}{tutu}{m}{n}
        { <5><6> toto5
        <7><8><9> toto7
        <10><10.95><12><14.4><17.28><20.74><24.88> toto10
        }{}
```

Il convient de placer ces fichiers là où LaTeX saura les trouver, par exemple `$TEXMF/tex/latex/local/`.

6.5 Comment utiliser une fonte MF comme fonte par défaut dans un document LaTeX ?

Il suffit de redéfinir les commandes `\encodingdefault`, `\rmdefault`, `\itdefault`, `\sfdefault` et `\bfdefault` dans le fichier `*.sty` que l'on a créé.

```
\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
%\renewcommand{\bfdefault}{bx}
%\renewcommand{\itdefault}{it}
```

Mais dans cet exemple il n'y a qu'une seule fonte : ce n'est pas suffisant. Il faut au moins des fontes maigre droite, maigre italique, grasse droite et grasse italique.

6.6 Comment utiliser plusieures fontes MF comme fonte par défaut dans un document LaTeX ?

Supposons maintenant que nous ayons des fontes `toto_maigre_droit.mf`, `toto_maigre_italique.mf`, `toto_gras_droit.mf`, `toto_gras_italique.mf` : comment en faire les fontes par défaut du document ?

Dans le fichier `t1toto.fd` :

```
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{   <-> toto_maigre_droit    }{}
\DeclareFontShape{T1}{toto}{m}{it}{  <-> toto_maigre_italique }{}
\DeclareFontShape{T1}{toto}{bx}{n}{  <-> toto_gras_droit      }{}
\DeclareFontShape{T1}{toto}{bx}{it}{ <-> toto_gras_italique   }{}
```

Dans le fichier `toto.sty` :

```
\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{it}
```

6.7 Comment peut-on modifier une fonte Métafont ?

Cela dépend bien évidemment de la fonte. Prenons l'exemple de la fonte `cmr10`. Le début du fichier est une liste de paramètres que l'on peut modifier. Cette liste est vraiment très longue et je n'ai jamais eu le courrage de la modifier. Pour des exemples de fontes Computer Modern Modifiées, vous pouvez regarder les fontes CMBright (le fichier `br10pt.mf`), `cmff10.mf`, ou les fontes accompagnant `mathkit`.

```
font_identifier:="CMR"; font_size 10pt#;

u#:=20/36pt#;      % unit width
width_adj#:=0pt#;    % width adjustment for certain characters
serif_fit#:=0pt#;    % extra sidebar near lowercase serifs
cap_serif_fit#:=5/36pt#;  % extra sidebar near uppercase serifs
letter_fit#:=0pt#;    % extra space added to all sidebars

body_height#:=270/36pt#;  % height of tallest characters
asc_height#:=250/36pt#;    % height of lowercase ascenders
```

```
cap_height#:=246/36pt#;    % height of caps
fig_height#:=232/36pt#;    % height of numerals
x_height#:=155/36pt#;    % height of lowercase without ascenders
math_axis#:=90/36pt#;    % axis of symmetry for math symbols
bar_height#:=87/36pt#;    % height of crossbar in lowercase e
comma_depth#:=70/36pt#;    % depth of comma below baseline
desc_depth#:=70/36pt#;    % depth of lowercase descenders

crisp#:=0pt#;      % diameter of serif corners
tiny#:=8/36pt#;      % diameter of rounded corners
fine#:=7/36pt#;      % diameter of sharply rounded corners
thin_join#:=7/36pt#;    % width of extrafine details
hair#:=9/36pt#;      % lowercase hairline breadth
stem#:=25/36pt#;    % lowercase stem breadth
curve#:=30/36pt#;    % lowercase curve breadth
ess#:=27/36pt#;      % breadth in middle of lowercase s
flare#:=33/36pt#;    % diameter of bulbs or breadth of terminals
dot_size#:=38/36pt#;    % diameter of dots
cap_hair#:=11/36pt#;    % uppercase hairline breadth
cap_stem#:=32/36pt#;    % uppercase stem breadth
cap_curve#:=37/36pt#;    % uppercase curve breadth
cap_ess#:=35/36pt#;    % breadth in middle of uppercase s
rule_thickness#:=.4pt#;    % thickness of lines in math symbols

dish#:=1/36pt#;      % amount erased at top or bottom of serifs
bracket#:=20/36pt#;    % vertical distance from serif base to tangent
jut#:=28/36pt#;      % protrusion of lowercase serifs
cap_jut#:=37/36pt#;    % protrusion of uppercase serifs
beak_jut#:=10/36pt#;    % horizontal protrusion of beak serifs
beak#:=70/36pt#;    % vertical protrusion of beak serifs
vair#:=8/36pt#;      % vertical diameter of hairlines
notch_cut#:=10pt#;    % maximum breadth above or below notches
bar#:=11/36pt#;      % lowercase bar thickness
slab#:=11/36pt#;    % serif and arm thickness
cap_bar#:=11/36pt#;    % uppercase bar thickness
cap_band#:=11/36pt#;    % uppercase thickness above/below lobes
cap_notch_cut#:=10pt#;    % max breadth above/below uppercase notches
serif_drop#:=4/36pt#;    % vertical drop of sloped serifs
stem_corr#:=1/36pt#;    % for small refinements of stem breadth
vair_corr#:=1/36pt#;    % for small refinements of hairline height
apex_corr#:=0pt#;    % extra width at diagonal junctions

o#:=8/36pt#;      % amount of overshoot for curves
apex_o#:=8/36pt#;    % amount of overshoot for diagonal junctions

slant:=0;      % tilt ratio $(\Delta x/\Delta y)$
fudge:=1;      % factor applied to weights of heavy characters
math_spread:=0;      % extra openness of math symbols
superness:=1/sqrt2;    % parameter for superellipses
superpull:=1/6;      % extra openness inside bowls
beak_darkness:=11/30;    % fraction of triangle inside beak serifs
ligs:=2;      % level of ligatures to be included

square_dots:=false;    % should dots be square?
hefty:=false;      % should we try hard not to be overweight?
serifs:=true;      % should serifs and bulbs be attached?
monospace:=false;    % should all characters have the same width?
variant_g:=false;    % should an italic-style g be used?
low_asterisk:=false;    % should the asterisk be centered at the axis?
```

```
math_fitting:=false;    % should math-mode spacing be used?
```

6.8 Peut-on transformer une fonte Métafont en une fonte PostScript ?

Oui, avec `mf2ps`[5] (je n'ai pas regardé comment cela marchait) A FAIRE ou avec `mf2pt3`[6] (un petit script en Perl qui utilise MétaPost).

6.9 Peut-on transformer une fonte Métafont en une fonte PostSCript de type 1 ?

Pas vraiment : les fontes Métafont sont décrites comme le chemin parcourru par un « crayon » de forme ellipsoïdale, alors que les fontes PostScript sont décrites par un *contour* qui est ensuite remplis. Le passage de l'un à l'autre est, parait-il, un problème de géométrie algorithmique non trivial[7]. Néanmoins, Métafog[8] (logiciel commercial) permet cette conversion.

6.10 Peut-on convertir une fonte Métafont en fonte True Type ? Peut-on utiliser les fontes Computer Modern avec d'autres logiciels que TeX ?

Pour la conversion, je ne sais pas trop (regardez du côté de Métafog[9]). Il existe des versions PostScript[10] et True Type[11] des fontes Computer Modern, qui sont donc utilisables par d'autres logiciels.

5. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/mf2pt3`
6. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/mf2pt3`
7. `http://truetex.com/bezint.htm`
8. `http://truetex.com/`
9. `http://truetex.com/`
10. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/cm/ps-type1/bluesky/`
11. `http://truetex.com/um.zip`

Chapitre 7

Fontes PostScript

7.1 Qu'est-ce que le PostScript ?

Le PostScript [1] est un langage de description de page, conçu par Adobe [2] et reconnu par certaines imprimantes. C'est un vrai langage de programmation (avec des variables, éventuellement locales, des fonctions, de la récursivité, des boucles, des conditionnelles, etc.). Il n'est pas nécessaire de le connaître pour utiliser des fontes PostScript, mais nous allons quand même présenter brièvement le langage. Le lecteur « normal » est prié de passer à la question suivante.

Pour que l'ordinateur reconnaisse facilement un fichier PostScript, ses deux premiers caractères doivent être `%!`.

Le caractère `%` est utilisé pour annoncer un commentaire, qui s'étend jusqu'à la fin de la ligne. Cela nous permet de comprendre le comprendre le début de la plupart des fichiers PostScript.

```
%!PS-Adobe-2.0
%%Creator: dvips(k) 5.85 Copyright 1999 Radical Eye Software
%%Title: these.dvi
%%Pages: 126
%%PageOrder: Ascend
%%BoundingBox: 0 0 596 842
%%DocumentFonts: XYATIP10 XYBTIP10
%%EndComments
%DVIPSWebPage: (www.radicaleye.com)
%DVIPSCommandLine: dvips -f these.dvi
%DVIPSParameters: dpi=600, compressed
%DVIPSSource:  TeX output 1999.12.12:2152
```

La première ligne indique le *niveau* (i.e., la version) du langage PostScript utilisée (il y en a trois : 1, 2 et 3 : à chaque fois on ajoute quelques possibilités supplémentaires ; attention à ne pas envoyer un fichier PostScript de niveau 3 à une imprimante qui ne connait que le 2). Les lignes suivantes sont des commantaires indiquant l'origine du fichier ou sa structure (par exemple, la « bounding box », ie, la taille de la page, ou le nombre de pages, pour les logiciels qui voudraient n'en extraire que quelques unes). Les commentaires qui commencent par `%%` répondent à une syntaxe très précise et facilitent la manipulation des fichiers PostScript par divers logiciels.

Pour nos premiers essais en PostScript, nous utiliserons l'interpréteur PostScript `ghostscript`.

Nous pouvons maintenant commencer à programmer un peu en PostScript. C'est un langage à pile : quand on lui donne des données, par exemple, des nombres, il les met sur une pile. Quand on lui donne une commande, qui attend des arguments, il les prend sur la pile. Ainsi, pour calculer « 1 + 1 », on tapera `1 1 add` et le résultat se retrouvera sur la pile. la commande `pstack` demande à `ghostscript` d'afficher la pile. Les opérations élémentaires sont `add`, `sub`, `mul`, `div`, `mod`, `idiv` (division entre entiers), `abs`, `neg`, `ceiling`, `floor`, `round`, `truncate`, `sqrt`, `atan`, `cos`, `sin`, `ln`, `log`, `exp` (deux arguments : la base, puis l'exposant), `rand` (entier aléatoire), `srand`. On remarquera que la notion de pile permet de se débarasser complètement des parenthèses. L'exemple suivant calcule « (1+2)*3 » et « 9*(1+2) ».

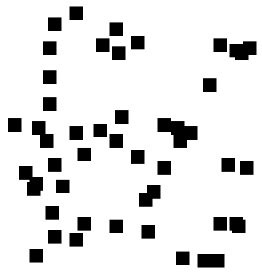
```
GNU Ghostscript 5.10 (1998-12-17)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
```

1. `http://partners.adobe.com/asn/developer/PDFS/TN/PLRM.pdf`
2. `http://partners.adobe.com/asn/developer/technotes.html`

```
GS>
GS> 1 2 add 3 mul pstack
9
GS<1> 1 2 pstack
2
1
9
GS<3> add mul pstack
27
```

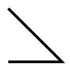
Voici un exemple d'utilisation de nombres au hasard.

```
5438 srand
50 {
  newpath
    rand 100 mod
    rand 100 mod
    moveto
    -5 0 rlineto
    0 -5 rlineto
    5 0 rlineto
    closepath
  fill
} repeat
```



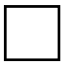
Expliquons maintenant comment dessiner en PostScript. On commence par définir un « chemin », i.e., un morceau de dessin *non encore dessiné*. On peut ensuite tracer de chemin, le colorier, écrire du texte le long de ce chemin, s'en servir comme d'un pochoir, etc. Il importe de bien comprendre que ces deux opérations, la description du chemin et son tracé, sont distinctes. La commande `newpath` commence un nouveau chemin. La commande `moveto` permet de se déplacer sans rien ajouter au chemin. La commande `lineto` permet de se déplacer en ajoutant un segment au chemin courrant. Ces deux commandes prennent deux coordonnées en argument (i.e., sur la pile). La commande `closepath` referme le chemin courrant (c'est nécessaire si on veut un chemin fermé, par exemple pour colorier la zône entourrée par le chemin). Enfin, la commande `stroke` trace le chemin alors que la commande `fill` remplit la surface qu'il délimite.

```
newpath
  0 0 moveto
  100 0 lineto
  0 100 lineto
stroke
```

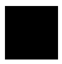


```
newpath
  0 0 moveto
  100 0 lineto
  100 100 lineto
  0 100 lineto
```

```
  closepath
stroke
```

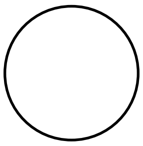


```
newpath
  0 0 moveto
  100 0 lineto
  100 100 lineto
  0 100 lineto
  closepath
fill
```



La commande `arc` permet de tracer des cercles ou des arcs de cercle. Ses deux premiers arguments définissent le centre du cercle, le suivant, son rayon et les deux derniers les angles (en degrés) entre lesquels l'arc sera tracé, dans le sens trigonométrique. Si jamais il y a un point courrant, l'interpréteur tracera aussi un segment de ce point vers le premier point de l'arc — mais si on trace l'arc juste après la commande `newpath`, il n'y a pas de point courrant.

```
400 400 translate
newpath
0 0 100 0 360 arc stroke
```

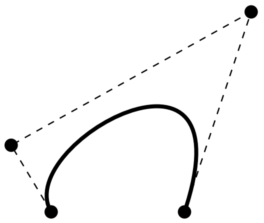


La commande `arcn` trace des arcs de cercle dans le sens des aiguilles d'une montre.

La commande `curveto` prend six arguments et trace des courbes de Bézier : les quatre points de contrôle sont le point courrant et les trois points dont les coordonnées sont prises sur la pile.

```
gsave
  50 50 translate
  /A {0 0} def
  /B {100 0} def
  /C {-30 50} def
  /D {150 150} def
  /point {gsave newpath 5 0 360 arc closepath fill grestore} def
  A point
  B point
  C point
  D point
  gsave
    [5 5] 0 setdash
    newpath
      A moveto C lineto D lineto B lineto
    stroke
  grestore
  3 setlinewidth
  newpath
    A moveto C D B curveto
```

```
  stroke
grestore
```



Pour utiliser une fonte, on commence par la commande `findfont`, qui attend un nom de fonte sur la pile et qui renvoie une structure de donnée (on parle de *dictionnaire*) décrivant cette fonte (en particulier les métriques et le code PostScript pour dessiner chaque caractère). On remarquera que le nom de la fonte est précédé d'un slash (c'est comme cela que l'on indique le *nom* des variables ou des fonctions, par opposition à leur valeur). La commande `scalefont` transforme cette structure de donnée pour que la fonte soit à la taille désirée. Finalement, la commande `setfont` prend cette structure de donnée et fait de la fonte qu'elle décrit la fonte courrante : on peut maintenant l'utiliser. Les chaînes de caractères sont entre parenthèses (et non pas entre guillemets, comme dans beaucoup de langages de programmation). On utilise la commande `show` pour dessiner les caractères.

```
/Times-Roman findfont 150 scalefont setfont
newpath
  10 100 moveto (ABCabc)
show
```

ABCabc

La plupart du temps (i.e., pour les fontes de type 1), la commande `show` se contente d'appeler les commandes `charpath` pour construire un chemin, le contour des caractères, puis `fill` pour le remplir. (La commande `charpath` attend un autre argument, qui sera toujours `false`.)

```
/Times-Roman findfont 150 scalefont setfont
newpath
  10 100 moveto (ABCabc) false charpath
fill
```

ABCabc

On peut alors utiliser ce chemin à d'autres fins : ici, nous ne traçons que le contour des caractères. Nous verrons d'autres exemples plus loin.

```
/Times-Roman findfont 150 scalefont setfont
newpath
  10 100 moveto (ABCabc) false charpath
stroke
```

ABCabc

L'état graphique désigne l'ensemble des caractéristiques du prochain élément qui sera dessiné : le chemin courrant, la position courrante, le niveau de gris (ou la couleur), le type de pointillés, l'orientation (plus généralement, la matrice de passage entre les coordonnées de la page et celles de l'utilisateur), la fonte courrante, le pochoir courrant, etc. La commande `translate` permet de déplacer le dessin (elle s'utilise avant que le chemin ne soit défini).

```
/Times-Roman findfont 150 scalefont setfont
10 100 translate
```

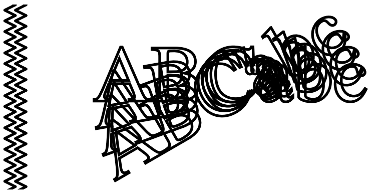
```
0 0 moveto (ABCabc) show
0 150 translate
0 0 moveto (ABCD) show
```

De même, la commande `rotate` permet d'effectuer une rotation.

```
/Times-Roman findfont 150 scalefont setfont
10 100 translate
0 0 moveto (ABCabc) false charpath stroke
10 rotate
0 0 moveto (ABCabc) false charpath stroke
10 rotate
0 0 moveto (ABCabc) false charpath stroke
10 rotate
0 0 moveto (ABCabc) false charpath stroke
```

Il s'agit d'une rotation par rapport à l'origine : si on veut effectuer la rotation autour d'un autre point, il convient de commencer par une translation.

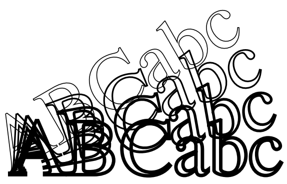
```
/Times-Roman findfont 150 scalefont setfont
50 300 translate
0 0 moveto (ABCabc) false charpath stroke
300 0 translate 10 rotate -300 0 translate
0 0 moveto (ABCabc) false charpath stroke
300 0 translate 10 rotate -300 0 translate
0 0 moveto (ABCabc) false charpath stroke
300 0 translate 10 rotate -300 0 translate
0 0 moveto (ABCabc) false charpath stroke
```



La commande `setlinewidth` permet de modifier l'épaisseur des traits.

```
/Times-Roman findfont 150 scalefont setfont
10 100 translate
4 setlinewidth
0 0 moveto (ABCabc) false charpath stroke
10 rotate
3 setlinewidth
0 0 moveto (ABCabc) false charpath stroke
```

```
10 rotate
2 setlinewidth
0 0 moveto (ABCabc) false charpath stroke
10 rotate
1 setlinewidth
0 0 moveto (ABCabc) false charpath stroke
```

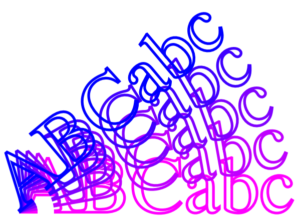


La commande **setgray** permet de définir le niveau de gris, entre `0` (noir) et `1` (blanc).

```
%!PS-Adobe-2.0
/Times-Roman findfont 150 scalefont setfont
newpath
  0.8 setgray
  10 100 moveto (ABCabc)
show
newpath
  0 setgray
  20 110 moveto (ABCabc)
show
```

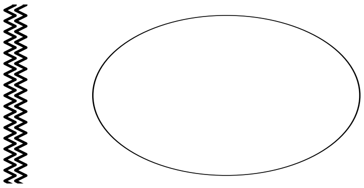
La commande **setrgbcolor** permet de choisir la couleur : les trois paramètres varient entre 0 et 1. Par exemple, blanc = `1 1 1`, noir = `0 0 0`, jaune = `1 1 0`, cyan = `0 1 1`, magenta = `1 0 1`.

```
/Times-Roman findfont 150 scalefont setfont
10 100 translate
1 0 1 setrgbcolor
0 0 moveto (ABCabc) false charpath stroke
10 rotate
.75 0 1 setrgbcolor
0 0 moveto (ABCabc) false charpath stroke
10 rotate
.5 0 1 setrgbcolor
0 0 moveto (ABCabc) false charpath stroke
10 rotate
.25 0 1 setrgbcolor
0 0 moveto (ABCabc) false charpath stroke
10 rotate
0 0 1 setrgbcolor
0 0 moveto (ABCabc) false charpath stroke
```



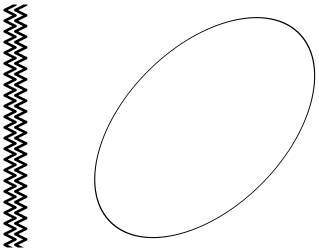
La commande `scale` permet d'étirer (ou de contracter) les axes de coordonnées. Cela permet par exemple de tracer des ellipses.

```
400 400 translate
1 0.6 scale
newpath
0 0 100 0 360 arc
stroke
```



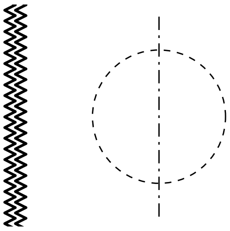
Il est possible de combiner les commandes `scale` et `rotate`, par exemple pour obtenir des ellipses orientées dans un sens quelconque. On fera attention à l'ordre de ces deux opérateurs.

```
400 400 translate
45 rotate
1 0.6 scale
newpath
0 0 100 0 360 arc
stroke
```



La commande `setdash` permet de définir le type de pointillés : le premier argument est un tableau, indiquant la longueur du trait, puis la longueur du vide, puis la longueur du vide, etc. et le second argument est toujours `0` (voir le manuel pour les détails).

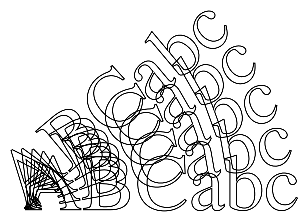
```
200 500 translate
newpath
  [5 5] 0 setdash
  0 0 50 0 360 arc
stroke
newpath
  [10 4 2 4] 0 setdash
  0 -75 moveto
  0 75 lineto
stroke
```



Il y a d'autres éléments dans l'état graphique : `setlinecap`, `setlinejoin`, `setmiterlimit`, bien expliqués dans le manuel.

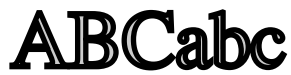
Les commandes `gsave` et `grestore` permettent de sauvegarder l'état graphique courrant (en le plaçant sur une pile) et de le récupérer. Cela permet par exemple de changer les coordonnées à grands coups de `translate`, `rotate` ou `scale` sans que cela ait de conséquence sur le reste du dessin. En d'autres termes, les modifications effectuées à l'état graphique entre `gsave` et `grestore` sont locales.

```
%!PS-Adobe-2.0
gsave
50 100 translate
0 0 moveto
/Times-Roman findfont 150 scalefont setfont
  gsave
    (ABCabc) false charpath stroke
  grestore
  gsave
    10 rotate
    (ABCabc) false charpath stroke
  grestore
  gsave
    20 rotate
    (ABCabc) false charpath stroke
  grestore
  gsave
    30 rotate
    (ABCabc) false charpath stroke
  grestore
  gsave
    40 rotate
    (ABCabc) false charpath stroke
  grestore
grestore
```



On a aussi recours à ces commandes quand on veut utiliser plusieures fois un même chemin.

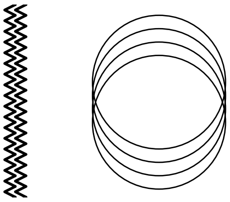
```
%!PS-Adobe-2.0
/Times-Roman findfont 150 scalefont setfont
gsave
  newpath
  10 100 moveto (ABCabc) false charpath
  gsave 0.8 setgray fill grestore
  2 setlinewidth stroke
grestore
```



On définit une fonction PostScript de la manière suivante : `/toto { ... } def`, où `toto` est le nom de la fonction (quand on veut donner à l'interpréteur le *nom* d'une fonction, on le fait précéder d'un slash) et les pointillés contiennent le corps de la fonction. Voici un exemple de fonction qui trace un cercle de rayon 50.

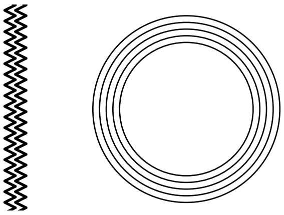
```
%!PS-Adobe-2.0
/cercle {
  newpath
```

```
    0 0 50 0 360 arc
  stroke
} def
gsave
  200 500 translate
  cercle
  0 10 translate cercle
  0 10 translate cercle
  0 10 translate cercle
grestore
```



Les éventuels paramètres sont passés par l'intermédiaire de la pile. la fonction suivante trace des cercles de rayon donné.

```
%!PS-Adobe-2.0
/cercle {
  newpath
    0 exch 0 exch 0 360 arc
  stroke
} def
gsave
  200 500 translate
  50 cercle
  55 cercle
  60 cercle
  65 cercle
  70 cercle
grestore
```



La commande `bind def` est l'équivalent PostScript du `\edef` de TeX : elle remplace toutes les fonctions appelées par la fonction que l'on définit par leur définition, de manière à accélérer l'exécution et à court-circuiter d'éventuelles redéfinitions.

L'une des manières d'écrire des fonctions qui prennent des paramètres consiste à savoir manipuler les éléments sur la pile. La commande `pop` permet de se débarasser du premier élément de la pile. La commande `dup` permet de le dupliquer (pour les cas où on en aurait besoin à plusieures reprises). La commande `exch` permet d'échanger les deux premiers éléments de la pile. La commande `index` prend un argument, n, et duplique le $(n+1)$ième élément de la pile.

```
GNU Ghostscript 5.10 (1998-12-17)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
GS>    5 4 3 2 1 0
GS<6>  5 index
GS<7>  pstack
5
0
1
```

```
2
3
4
5
```

La commande `1 roll` prend un argument (en plus du `1` que j'ai déjà mis), n, et va effectuer une rotation sur les n premiers éléments de la pile, de manière que le 2ième élément se retrouve en premier et que le premier se retrouve à la $n$ième place.

```
GNU Ghostscript 5.10 (1998-12-17)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
GS>    5 4 3 2 1
GS<5>  pstack
1
2
3
4
5
GS<5>  3 1 roll
GS<5>  pstack
2
3
1
4
5
GS<5>
```

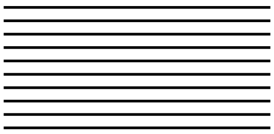
La commande `clear` efface tout le contenu de la pile et la commande `count` compte le nombre total d'éléments de la pile : ces commandes ne sont pas vraiment utilisables, car il est toujours possible qu'une autre partie du programme ait mis des données sur la pile. Pour rendre les choses un peu plus locales, la commande `mark` met une marque sur la pile et les commande `cleartomark` et `counttomark` n'affectent que la partie de la pile jusqu'à la marque. Nous reparlerons des marques quand nous rencontrerons des tableaux ou des dictionnaires (ie, des tables de hachage).

```
GNU Ghostscript 5.10 (1998-12-17)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
GS>     3 2 1 mark 5 4 3 2 1 pstack
1
2
3
4
5
-mark-
1
2
3
GS<9>
GS<9>   counttomark
GS<10>  pstack
5
1
2
3
4
5
-mark-
1
2
3
```

```
GS<10>  cleartomark
GS<3>   pstack
1
2
3
```

La commande `repeat` répète une procédure un nombre de fois fixé.

```
%!PS-Adobe-2.0
gsave
  10 {
    0 5 translate
    newpath
      0 0 moveto
      100 0 lineto
    stroke
  } repeat
grestore
```

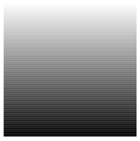


La commande `loop` est une boucle infinie dont on sort par la commande `exit`.

```
%!PS
% Ne pas envoyer ce fichier à une imprimante !!!
% Il n'y a aucun rapport entre la taille d'un fichier
% et le nombre de pages qu'il contient ou la quantité
% d'encre qu'il va utiliser.
{0 0 10000 0 360 arc fill showpage} loop
```

EXEMPLE avec `exit`. La commande `for` est une boucle plus évoluée : elle attend sur la pile, la valeur de départ du compteur, sont incrémentation et sa valeur finale. À chaque exécution de la boucle, on se retrouve avec la valeur du paramètre sur la pile (si on n'en a pas besoin, on peut s'en débarasser avec `pop` — mais alors, la commande `repeat` aurait peut-être été préférable).

```
%!PS-Adobe-2.0
gsave
  10 10 translate
  0 .01 1 {
    setgray
    newpath
      0 0 moveto
      100 0 lineto
    stroke
    0 1 translate
  } for
grestore
```



```
%!PS
gsave
```

```
  /Times-Roman findfont 150 scalefont setfont
  30 10 translate
  0 1 20 {
    gsave
      dup dup neg exch translate 0 0 moveto
      (ABCabc) false charpath 20 div 1 sub neg setgray fill
    grestore
  } for
grestore
```

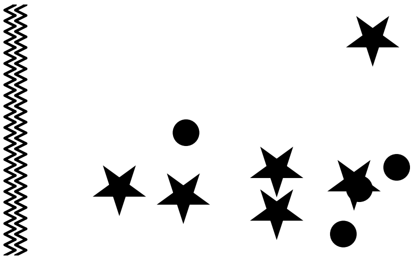

```
%!PS
gsave
  200 200 translate
  /hauteur 600 def
  /x hauteur 255 div def
  /bouboule {
    gsave
      .1 .1 scale
      300 0 moveto 0 0 300 0 360 arc
      gsave clip
        x 1 add setlinewidth
        hauteur .2 mul dup translate
        0 1 255 {
          gsave
            255 div dup 1 sub neg setgray
            hauteur .8 mul mul dup 0 moveto
            0 exch 0 exch 0 360 arc
            stroke
          grestore
        } for
      grestore 5 setlinewidth stroke
    grestore
  } bind def
  6572 srand
  /h { rand 2 30 {2 mul} repeat div 200 mul } def
  0 1 100 { pop
    gsave h h translate bouboule grestore
  } for
grestore
```

Passons maintenant aux conditionnelles : la commande `if` attend un booléen et un bloc sur la pile ; la commande `ifelse` attend un booléen et deux blocs sur la pile. Les différentes opérations booléennes sont `eq` (égalité), `ne` (non égalité), `lt` (inférieur à), `gt` (supérieur à), `le` (inférieur ou égal à), `ge` (supérieur ou égal à), `and`, `or`, `not`, `xor`, `true`, `false`.

```
%!PS
7438 srand
/point  { rand 100 mod rand 100 mod } def
/cercle { newpath point 5 0 360 arc fill } def
/etoile {
  gsave newpath
    point translate
    0 0 moveto
```

```
    5 { 144 rotate 20 0 translate 0 0 lineto } repeat
    fill
  grestore } def

50 50 translate
10 {
  rand 2 mod 0 eq
  {cercle} {etoile}
  ifelse
} repeat
```



Il est possible de définir des *tableaux* et de les manipuler avec les commandes `[`, `]`, `array`, `length`, `get`, `put`, `forall`, etc.

```
GNU Ghostscript 5.10 (1998-12-17)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
GS>    [ 0 1 2 3 4 5 ] pstack
[0 1 2 3 4 5]
GS<1>  /t exch def
GS>    t length pstack pop
6
GS>    t 3 get pstack pop
3
GS>    t 3 5 put
GS>    t pstack
[0 1 2 5 4 5]
```

Les *dictionnaires* sont une autre structure de donnée très utile : ce sont des tables de hachage. On définit un nouveau dictionnaire en donnant sa taille (nombre d'entrées) à la commande `dict`. Le dictionnaire vide est alors sur la pile. Comme on en aura besoin par la suite, on peut le mettre dans une variable (c'est juste un pointeur). Pour mettre des choses dedans, on va l'ouvrir, avec la commande `begin` : il disparait alors de la pile (et s'en va sur une autre pile, celle des dictionnaires). on peut alors rajouter des entrées dedans, à l'aide de la commande `def`. Quand on en a fini avec, on le ferme, à l'aide de la commande `end`.

```
/mondico 8 dict def
mondico begin
  /a 10       def
  /b 1.4      def
  /c (ABCabc) def
end
```

Les commandes `<<` et `>>` sont un raccourci pour créer des dictionnaires.

```
<<
  /a 10
  /b 1.4
  /c (ABCabc)
>>
/mondico exch def
```

On pourrait définir la commande `<<` comme étant `mark` et la commande `>>` comme suit.

```
mark
  /a 1
  /b 2
% On compte le nombre d'éléments
counttomark 2 idiv dup
% On crée un dictionnaire, on l'ouvre
% et on en laisse une copie sur la pile
dict dup begin exch
% On définit les variables, en prenant soin de
% ne pas toucher à la copie du dictionnaire
% qui est en haut de la pile.
{ 3 1 roll def } repeat
% On ferme le dictionnaire
end
% On enlève la marque
exch pop
```

Il est possible d'extraire ou de modifier le contenu d'un dictionnaire sans l'ouvrir, à l'aide des commandes `get` et `put`

```
GNU Ghostscript 5.10 (1998-12-17)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
GS>    <<
GS<1>    /a 10
GS<3>    /b 1.4
GS<5>    /c (ABCabc)
GS<7>  >>
GS<1>  pstack
-dict-
GS<1>  /mondico exch def
GS>    mondico /a get pstack
10
GS<1>  mondico /a 11 put
GS<1>  mondico /a get pstack
11
10
```

Le manuel mentionne d'autres commandes pour manipuler des dictionnaires : `load`, `undef`, `known`, `copy` (attention : les dictionnaires sont en fait des pointeurs, il ne faut donc pas les copier « naïvement »), `forall`, etc.

Les dictionnaires permettent aussi de créer des variables locales. Par exemple, si vous voulez utiliser des variables qui ont un nom très simple, du genre `/a`, `/b`, alors que des variables de même nom sont utilisées dans une autre partie du programme, vous pouvez créer un dictionnaire qui contiendra les variables locales de cette partie du programme : pour d'autre langages de programmation, on parlerait d'espaces de noms. Par exemple, `dvips` fait cela automatiquement quand il inclut des fichiers PostScript — car ils sont parfois mal programmés...

Il est aussi possible de manipuler les chaines de caractères, des fichiers, mais nous n'en parlerons pas (voir le manuel).

La commande `clip` prend le chemin courrant et l'utilise comme *pochoir* : lors des tracés suivants, seules les parties du dessin qui sont dans le pochoir seront tracées.

```
%!PS
gsave
  /Times-Roman findfont 150 scalefont setfont
  10 10 translate
  gsave 0 300 translate
    0 0 moveto (ABCabc) false charpath clip newpath
    0 3 200 {gsave 0 exch translate 0 0 moveto 600 0 lineto stroke grestore} for
  grestore
grestore
```

Il y a quelquad temps, en cherchant à « affiner » une fonte grasse, j'ai pensé utiliser ce genre de chose (la manipulation équivalente marche très bien pour épaissir des caractères), mais ce n'était pas une bonne idée :

```
%!PS
gsave
  /Times-Roman findfont 150 scalefont setfont
  10 10 translate
  0 0 moveto (ABCabc) show
  0 150 translate
  0 0 moveto (ABCabc) false charpath clip
  6 0 translate 0 0 moveto (ABCabc) false charpath clip
  -3 2 translate 0 0 moveto (ABCabc) false charpath clip
  0 0 moveto (ABCabc) show
grestore
```

À partir du niveau 2, les pochoirs peuvent être un peu plus compliqués. Je donne juste un exemple et je renvoie au manuel pour les détails.

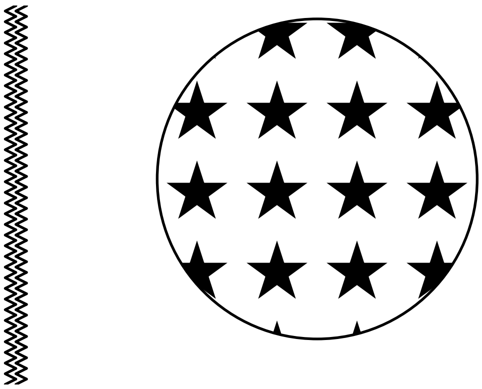
```
%!PS-Adobe-2.0
% color patterns
<<
  /PaintType   1
  /PatternType 1
  /TilingType  1
  /BBox        [ 0 0 60 60 ]
  /XStep       60
  /YStep       60
  /star {
    gsave
      0 12 moveto
      4 { 144 rotate 0 12 lineto } repeat
      closepath fill
    grestore
  }
  /PaintProc {
    begin
      0.3 setgray
      15 15 translate star
      30 30 translate star
      0.7 setgray
      -30   0 translate star
       30 -30 translate star
    end
  }
>>
matrix makepattern /Star4 exch def

120 120 184 120 4 copy
/Pattern setcolorspace Star4 setcolor
%Star4 setpattern
```

```
rectfill
0 setgray
rectstroke
/Times-Roman 270 selectfont
160 100 translate
0.9 setgray      0 0 moveto (A) show
Star4 setpattern 0 0 moveto (A) show
```

```
%!PS-Adobe-2.0
% colorless patterns
<<
  /PaintType   2
  /PatternType 1
  /TilingType  1
  /BBox        [ -12 -12 12 12 ]
  /XStep       30
  /YStep       30
  /PaintProc {
    pop
    0 12 moveto
    4 { 144 rotate 0 12 lineto } repeat
    closepath fill
  }
>>
matrix makepattern /Star exch def

140 110 170 100 4 copy
0.9 setgray rectfill
[/Pattern /DeviceGray] setcolorspace
1 Star setcolor rectfill
225 185 60 0 360 arc
0 Star setpattern gsave fill grestore
0 setgray stroke
```



***** A FAIRE : Images bitmap *****

7.2 Qu'est-ce qu'une fonte PostScript ?

Une fonte PostScript [3] est un dictionnaire (au sens PostScript) contenant des procédures permettant de dessiner les différents caractères.

***** A FAIRE : donner une liste de quelques unes des entrées de ce dictionnaire. *****

3. http://partners.adobe.com/asn/developer/PDFS/TN/T1_SPEC.PDF

Il y a plusieurs types de fontes PostScript :

Type 1 La plupart des fontes PostScript sont de type 1. Elles comportent au plus 256 caractères. Les procédures PostScript qui décrivent les glyphes ne sont pas quelconques : elles ne peuvent utiliser qu'une partie du langage PostScript, de manière que l'on puisse être sûr que le caractère sera toujours tracé de la même manière, sans qu'il utilise des variables extérieures, sans qu'il produise d'erreur, sans qu'il entraine des effets de bord. De surcroît, ces prodédures sont codées (un peu comme les chemins utilisateurs peuvent l'être). Dernier point, les procédures qui décrivent les glyphes le définissent que leur contour, mais ne le remplissent pas.

Type 3 Les fontes de type 3 sont comme les fontes de type 1 sans toutes ces restrictions : les procédures tracent vraiment les caractères, elles ne sont pas codées et peuvent contenir absolument n'importe quoi (on supposera quand même qu'elles ne provoquent pas d'erreurs, mais on n'en est pas sûr). Les fontes bitmap utilisées par TeX sont de type 3

Type 0 Les fontes de type 0 sont des fontes composites : il ne s'agit pas d'une unique fonte mais d'un ensemble de fontes. Cela permet de dépasser la limite des 256 caractères par fonte et ainsi d'imprimer facilement du texte dans des langues idéographiques. Pour de plus amples renseignements, voir CJK (5.6).

Type 42 C'est une fonte True Type légèrement modifiée pour être reconnue par l'interpréteur PostScript.

Type 2 Fonte CFF (Compact Font Format)

Types 9, 10, 11 32 CID-keyed fonts

Type 14 Fonte Chameleon

A FAIRE : donner des exemples de fontes.

7.3 Comment créer soi-même une fonte PostScript de type 3?

Voici un exemple de fonte PostScript de type 3 (tiré du manuel).

```
%!
% Type 3 font
8 dict begin
  /FontType   3 def
  /FontMatrix [.001 0 0 .001 0 0] def
  /FontBox    [0 0 1000 1000] def
  /Encoding   256 array def
    0 1 255 { Encoding exch /.notdef put } for
    Encoding 97 /Square   put
    Encoding 98 /Triangle put
  /CharProcs  3 dict def
  CharProcs begin
    /.notdef {} def
    /Square { 0 0 moveto 750 0 lineto 750 750 lineto
              0 750 lineto closepath fill } bind def
    /Triangle { 0 0 moveto 375 750 lineto 750 0 lineto
                closepath fill } bind def
  end
  /BuildGlyph { 1000 0         % largeur
                0 0 750 750    % cadre
                setcachedevice
                exch /CharProcs get exch
                2 copy known not {pop /.notdef} if
                get exec
             } bind def
  /BuildChar { 1 index /Encoding get exch def % ???
               1 index /BuildGlyph get exec
             } bind def
  currentdict
end
/ExampleFont exch definefont
pop % le dictionnaire est sur la pile
/ExampleFont findfont 12 scalefont setfont
36 52 moveto (ababa) show
```

***** J'ai oublié de parler des métriques *****

7.4 Comment créer soi-même une fonte PostScript de type 1 ?

***** A FAIRE *****

7.5 Comment avoir les contours d'une fonte de type 1 ?

Il suffit de modifier l'entrée `PaintType` de son dictionnaire (mettre 2 à la place de 0) et de rajouter une entrée `StrokeWidth`. L'essai naïf suivant ne marche pas, car le dictionnaire que l'on essaie de modifier n'est accessible qu'en lecture. C'est le cas de toutes les fontes, même celles que l'on aurait définies soi-mêmes : tant qu'on n'a pas transformé le dictionnaire en fonte à l'aide de la commande `definefont`, on peut en faire ce que l'on veut, mais après, tout est figé.

```
/Times-Roman findfont
dup /StrokeWidth 1 put
dup /PaintType 2 put
150 scalefont
setfont
newpath
  10 100 moveto (ABCabc)
show
```

On va donc copier le dictionnaire définissant la fonte, le modifier et en faire une fonte.

```
/Times-Roman findfont
dup length 1 add dict begin
  {  1 index /FID ne
     {def}
     {pop pop} ifelse
  } bind forall

  /StrokeWidth 1 def
  /PaintType 2 def

  currentdict
end
/myfont exch definefont
pop

/myfont findfont 30 scalefont
setfont
newpath
  10 100 moveto (ABCabc)
show
```

7.6 Comment transformer une fonte de type 1 en une fonte de type 3 ?

***** A FAIRE ***** Je suggère de procéder ainsi : partir d'une fonte `*.pfa`, la décoder à l'aide de `t1disasm`,

```
t1disasm toto.pfa > toto.t1
```

Changer la valeur de `FontType`

```
perl -p -i -e 's/FontType\s+3/FontType 2/g' toto.t1
```

Rajouter les commandes `BuildChar` (qui attent un dictionnaire et un nom de caractère sur la pile et dessine ce caractère) et `BuildGlyph` (idem, mais avec le numéro du caractère).

```
perl -p -e -i 'print "
  /BuildChar {
    1 index /Encoding get exch get
    1 index /BuildGlyph get exec
  } bind def
  /BuildGlyph { 1000 0
                0 0 750 750
                setcachedevice
                exch /CharStrings get exch
                2 copy known not {pop /.notdef} if
                get exec
              } bind def
" if m/FontType/' toto.t1
```

Ensuite, je constate que ça ne parche pas. *********** Autre idée : dans les divers fichiers de GhostScript, on trouve ceci.

```
(%Type1BuildChar) cvn   % <font> <code> %Type1BuildChar -
 { 1 index /Encoding get 1 index get .type1build
 } bind def
%Type1BuildGlyph) cvn  % <font> <name> %Type1BuildGlyph -
 { dup .type1build
 } bind def
/.type1build            % <font> <code|name> <name> .type1build -
 { 2 index begin
    dup CharStrings exch .knownget not
     { 2 copy eq { exch pop /.notdef exch } if
       QUIET not
        { (Substituting .notdef for ) print = flush }
        { pop }
       ifelse
       /.notdef CharStrings /.notdef get
     } if
   end .type1execchar
 } bind def
```

Mais je crais que `.type1execchar` soit propre à GhostScript. ******* A COMPRENDRE

7.7 Comment changer le codage d'une fonte PostScript ?

On procède comme précédemment, avec la variable `Encoding` de la fonte. L'exemple suivant intervertit les lettres A et B.

```
clear
/Times-Roman findfont
dup length 1 add dict begin
  {  1 index /FID ne
     {def}
     {pop pop} ifelse
  } bind forall
  /Encoding
  [
    Encoding
    {
```

```
      dup /A eq
      {pop /B}
      {
        dup /B eq
        {pop /A}
        if
      }
      ifelse
    }
    forall
  ]
  def
  currentdict
end
/myfont exch definefont
pop

/myfont findfont 30 scalefont
setfont
newpath
  10 100 moveto (ABCabc)
show
```

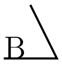
BACabc

Généralement, on se contente de reprendre un codage déjà existant (voir 7.14).

7.8 Comment mettre des commandes PostScript dans un fichier LaTeX ?

On peut par exemple utiliser PSTricks. (Normalement, on n'utilise pas PSTricks ainsi : il y a déjà plein de commandes déjà définies qui permettent de faire à peu près tout ce que l'on veut, plus simplement et plus sûrement que si on devait programmer soi-même en PostScript.)

```
\documentclass{minimal}
\usepackage{pstricks}
\begin{document}
A
\pscustom{\code{
  gsave
  newpath
  0 0 moveto
  20 0 lineto
  10 20 lineto
  stroke
  grestore
}}
B
\end{document}
```

A B 

On peut aussi utiliser la commande `\special`, mais ce n'est pas portable.

```
\documentclass{minimal}
\begin{document}
A
```

```
\special{ps:
  gsave
  newpath
  0 0 moveto
  100 0 lineto
  10 100 lineto
  stroke
  grestore
}
B
\end{document}
```

A B

```
\documentclass{minimal}
\begin{document}
A
\special{ps:
  gsave
  currentpoint translate
  newpath
  0 0 moveto
  100 0 lineto
  10 100 lineto
  stroke
  grestore
}
B
\end{document}
```

A B

7.9 Comment visualiser une fonte PostScript ?

Bonne question. J'imagine deux solutions : soit utiliser directement ghostscript pour visualiser la fonte, soit passer par LaTeX pour créer un fichier PostScript l'utilisant puis utiliser ghostscript pour le visualiser. Regardons la première solution (pour la deuxième, voir la question suivante).

Si Ghostscript connait déjà la fonte, on peut simplement lui dire

```
%!
/Helvetica findfont 72 scalefont setfont
72 72 moveto
(Hello, world!) show
showpage
```

Mais bien évidemment, les situations intéressantes sont celles où il ne connait pas la fonte. Il faut donc l'inclure dans le fichier. Pour cela, nous avons besoin de connaître le nom PostScript de la fonte (qu'n'a rien à voir avec le nom du fichier) : il est indiqué dans le fichier, dans une ligne du genre

```
/FontName /Toto def
```

Notre fonte s'appelle donc `Toto` (la majuscule est importante). Pour visualiser la fonte, on devrait pouvoir procéder ainsi :

```
(toto.pfa) run
/Toto findfont 72 scalefont setfont
72 72 moveto
```

```
(Hello, world!) show
showpage
```

Il y a aussi un programme en PostScript, qui vient avec Ghostscript et qui fait cela. On lance Ghostscript avec le nom de la fonte et le nom du programme,

```
gs Utopia-Regular.pfa prfont.ps
```

Ensuite, on tape les commandes suivantes au prompt de Ghostscript (ici encore, `Utopia-Regular` est le nom PostScript de la fonte, qui n'est peut-être pas le même que celui du fichier).

```
/Utopia-Regular DoFont
```

On constate que cela marche très avec les fontes `*.pfa` mais pas du tout avec des `*.pfb` : par contre, il suffit de convertir les fichiers `*.pfb` en `*.pfa` à l'aide de `pfb2pfa`.

```
for i in *.pfb
do
  pfb2pfa $i
done
```

On peut automatiser cela de la manière suivante.

```
for i in *.pfa
do
  name=`grep FontName $i | awk '{print $2}'`
  gs $i prfont.ps <(echo $name DoFont)
done
```

Si on préfère GhostView, pour pouvoir par exemple zoomer sur certaines parties de la page (avec `prfont.ps`, il vaut mieux éviter GhostView), on peut automatiser les choses ainsi.

```
for i in *.pfa
do
  name=`grep FontName $i | awk '{print $2}'`
  echo \($i\) run                                     > $i.ps
  echo $name findfont 72 scalefont setfont           >> $i.ps
  echo "72 144 moveto (Hello, world!) show showpage" >> $i.1.ps
  echo '***' $i
  ghostview $i.ps
done
```

7.10 Comment réaliser un catalogue de fontes PostScript ?

On peut par exemple procéder ainsi (je suppose que le codage des fontes est celui d'Adobe : avec des fontes T1 ou autres, le résultat sera bizarre). [Question : le codage n'est-il pas indiqué dans la fonte elle-même ? S'il n'est pas indiqué, il n'y a pas trop d'espoir de parvenir à un script qui permette de faire les choses automatiquement, par contre, si le codage est précisé, cela ne devrait pas poser de problème.]

```
#! perl -w

# BUG : Il vaut mieux que les noms de fichiers soient en minuscules.
#       De même, il est préférable q'ils ne contiennent pas de
#       caractères que LaTeX est suceptible de ne pas aimer.
#       On peut utiliser la commande rename :
#           rename 'y/A-Z/a-z/' *.pfa *.pfb *.afm *.pfm *.ps

# Liste des fichiers *.pfa et *.pfb
```

```
opendir(DIR,"./") || die "Cannot opendir ./ : $!";
foreach (readdir DIR) {
  push @fontes, $_ if m/\.(pfa|pfb)$/i;
}
closedir DIR;

open(HTML,">Catalogue.html") || die "Cannot open Catalogue for writing : $!";
print HTML '<HTML><HEAD><TITLE>Catalogue de fontes PostScript</TITLE></HEAD><BODY>';
print HTML '<H1>Catalogue de fontes PostScript</H1><P><HR>';

foreach $fonte (@fontes) {
  print STDERR "\n*** Processing $fonte\n";
  print HTML "$fonte\n";
  $base = $fonte;
  $base =~ s/\.(pfa|pfb)$//i;
  $afm = $base .".afm";
  $pfm = $base .".pfm";
  $tex = $base .".tex";
  $dvi = $base .".dvi";
  $tex = $base .".tex";
  $ps  = $base .".ps";
  $gif = $base .".gif";

  $aux = $base .".aux";
  $log = $base .".log";
  $vf  = $base .".vf";
  $vpl = $base .".vpl";

  # Création des métriques et du fichier psfonts.map
  if( -f $afm ){
    # Les métriques sont là
  } else {
    # Pas des métriques : on prend juste la taille des caractères
    print HTML "(pas de métriques)\n";
    print STDERR "Warning: missing metrics file\n";
    system "gs -dNODISPLAY -q -- pf2afm.ps $fonte";
  }

  $psfonts = `afm2tfm $base -v $base -t ec.enc`;
  chomp($psfonts);
  system "vptovf $base";
  open(PSFONTS,">psfonts.map") || die "Cannot open psfonts.map for writing : $!";
  print PSFONTS "$psfonts \"ECEncoding ReEncodeFont\" <ec.enc <$fonte\n";
  close PSFONTS;

  # Création du fichier LaTeX

  system "cp Template.tex $tex";
  system "perl -p -i -e 's/toto/$base/g' $tex";
  system "latex $tex";
#  system "xdvi $dvi";
  system "dvips -E -f $dvi -o $ps";
  system "pstogif -out $gif -density 288 $ps"; # 288 = 4*72

  unlink $tex, $dvi, $ps, $aux, $log, $vf, $vpl;

  print HTML "<P> <IMG SRC=\"$gif\"> <P> <HR>\n";
}
```

```
print HTML "Automatically generated on ". `date`;
print HTML '</BODY>';
close(HTML);
```

Le fichier `Template.tex` est le même que celui que nous avons utilisé pour réaliser un catalogue de fontes TTF.

Voici quelques exemples de résultats : free fonts (gimp) [4], sharefonts (gimp) [5], [6].

7.11 Y a-t-il des versions PostScript des fontes Computer Modern?

Oui, ce sont celles utilisées par défaut par `pdftex`. Si on veut les utiliser avec TeX, il suffit de demander à `dvips` de lire le fichier `cm.map` (voir 7.12). Toutefois, ces fontes n'existent qu'avec le codage `OT1` : si on préfère le codage `T1`, on peut utiliser les fontes virtuelles `ae`.

7.12 Comment dire à dvips quelles sont les fontes PostScript ?

La liste des correspondances entre les noms de fontes utilisés par LaTeX, le nom des fontes PostSCript et le nom des fichiers les contenant est stockée dans des fichiers `*.map`. Le fichier `psfonts.map` contient la liste des fontes PostScript standard. Si on crée un fichier `psfonts.map` dans le répertoire courrant, c'est lui qui sera lu et pas celui par défaut (c'est ce que l'on peut faire si on veut essayer uen fonte sans vraiment l'installer). Pour rajouter des fontes à cette liste, il suffit de modifier le fichier de configuration général de `dvips`, `config.ps` ou le fichier de configuration propre à chaque utilisateur, par défaut `~/.dvipsrc`. Par exemple, pour demander à `dvips` d'utiliser les fontes PostSCript mentionnées dans le fichier `toto.map`, il suffit de rajouter la ligne suivante à la fin de son `~/.dvipsrc`.

```
p +toto.map
```

7.13 Comment utiliser une fonte PostScript de manière ponctuelle dans un document LaTeX ?

Tout d'abord, ne nous préoccupons pas du codage des fontes. (Si vous voulez tout de suite la bonne réponse, passez à la question suivante.)

Prenons une fonte PostScript : deux fichiers, `toto.afm` et `toto.pfb`. (Si ce n'est pas le cas, vous risquez d'avoir des problèmes : Sur PC, on trouve parfois un fichier `*.pfm` à la place, que l'on peut convertir grace à pfm2afm [7] (je n'ai pas essayé). Si vous n'avez rien, on peut en obtenir grace à la commande pf2afm [8] (c'est un programme en PostScript), mais le résultat sera de pauvre qualité : tout ce que l'on peut faire, c'est prétendre que la taille des caractères est exactement celle de la plus petite boite qui les contient ; pour des caractères très penchés, l'effet est desastreux (j'ai déjà essayé))

LaTeX a besoin des métriques au format `*.tfm` :

```
afm2tfm toto
```

et dvips a besoin de savoir où trouver la fontes, rajouter la ligne suivante dans psfonts.map :

```
toto toto <toto.pfa
```

En fait, ce n'est pas exactement cette ligne-là : le premier `toto` est le nom de la fonte pour LaTeX, le second est le nom postscript de la fonte (qui est précisé dans le fichier postscript, ici, ce pourrait être `Toto`) et le dernier est le nom du fichier contenant la fonte. Le premier et le dernier sont connus et la commande `afm2tfm` peut nous donner le second.

On peu automatiser cela :

```
rm -f psfonts.map
for i in *.pfb
do
```

4. `http://www.math.jussieu.fr/~zoonek/LaTeX/PFB/gimp/freefont/`
5. `http://www.math.jussieu.fr/~zoonek/LaTeX/PFB/gimp/sharefont/`
6. `http://www.math.jussieu.fr/~zoonek/LaTeX/PFB/essai/Catalogue.html`
7. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/pfm2afm`
8. `ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/pf2afm`

```
  j=‘echo $i | sed ’s/\..*//’‘
  f=‘afm2tfm $j‘
  echo $f \<$j.pfb >> psfonts.map
done
```

On peut ensuite essayer de l'utiliser comme d'habitude :

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{  <-> bg3r }{}
\DeclareFontShape{T1}{toto}{m}{it}{ <-> sub * toto/m/n }{}
\DeclareFontShape{T1}{toto}{bx}{n}{ <-> sub * toto/m/n}{}
\DeclareFontShape{T1}{toto}{bx}{it}{<-> sub * toto/m/n}{}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{it}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\begin{document}

Dans ce document, la fonte par défaut est une fonte PostScript.
Il n’y a pas de \textbf{gras}, pas d’\textit{italique} et pas
de \textbf{\textit{gras italique}}.

\end{document}
```

On n'a pas du tout tenu compte du codage de la fonte : les caractères accentués ne passent pas du tout.

Remarquons que `xdvi` est normalement capable, tout seul, de créer les fontes bitmap (`*pk`) pour visualiser un document contenant des fontes postscript (pourvu, bien sûr, qu'il trouve le fichier `psfonts.map`). Il lance la commande `mktexpk`,

```
mktexpk --mfmode ljfour --bdpi 600 --mag magstep\(1.0\) --dpi 720 bg3r
```

qui a son tour lance `gsftopk`

```
gsftopk bg3r 720
```

(à la place de `gsftopk`, on peut aussi utiliser `ps2pk`). Pour accélérer les choses (i.e., pour éviter qu'il ne commence par chercher partout s'il n'y a pas déjà un fichier `*.pk` quelque part, puis un fichier `*.mf` qui permettrait de le créer, puis un fichier `*.hbf` qui permettrait de le créer, puis un fichier `*.ttf` qui permettrait de le créer — ce qui peut prendre du temps), on peut lancer cette commande soi-même.

7.14 Comment utiliser une fonte PostScript de manière ponctuelle dans un document LaTeX (en tenant compte du codage) ?

(Suite de la question précédente)

Cette fois-ci, il va falloir utiliser la commande `afm2tfm` de manière plus pertinente : on peut lui demander de changer le codage. [Question : pourquoi ne demande-t-on pas à LaTeX d'utiliser le codage de la fonte ? À cause des problèmes de césure pour les lettres accentuées ?]

La documentation de `afm2tfm` est dans le manuel de `dvips`.

```
afm2tfm bg3r -v bg3r -t ec.enc
vptovf bg3r
```

La première ligne crée un fichier `*.tfm` (avec le mauvais codage) ainsi qu'un fichier `*.vpl`, qui contient une description humainement lisible de la fonte virtuelle. La deuxième ligne transforme le fichier `*.vpl` en des fichiers *.tfm (avec le bon codage) et en une fonte virtuelle `*.vf` compréhensible par l'ordinateur.

QUESTION : parfois, je met des choses du genre « t1.enc », parfois « ec.enc » : quelle est la diffférence ? [Réponse : aucune, sauf que EC n'est normalement pas un codage mais une fonte utilisant le codage Cork, aussi connus sous le nom de T1 ou 8t. Il faudrait remplacer tous les `ec.enc` par des `t1.enc`.]

Il faut aussi modifier le fichier psfonts.map, en lui disant qu'il faut changer le codage :

```
bg3r AmericanGaramondBT-Roman  "ECEncoding ReEncodeFont" <ec.enc <bg3r.pfa
```

On automatise cela comme précédemment.

```
rm -f psfonts.map
for i in *.pfb
do
  j=`echo $i | sed 's/\..*//'`
  f=`afm2tfm $j -v $j -t ec.enc`
  vptovf $j
  echo $f '"ECEncoding ReEncodeFont" <ec.enc' \<$j.pfb >> psfonts.map
done
```

Sur le même exemple que tout-à-l'heure, on constate que cela marche.

7.15 Comment utiliser plusieures fontes PostScript de manière ponctuelle dans un document LaTeX ?

Cette fois, il ne s'agit pas d'une unique fonte PostScript, mais plusieures, une fonte normale, une fonte grasse, une fonte italique et une fonte italique grasse. Par exemple des fichiers `bg3r.pfb`, `bg3b.pfb`, `bg3ri.pfb` et `bg3bi.pfb`, avec les métriques `*.afm` correspondantes. On procéde comme dans la question précédente, pour créer les fontes virtuelles et on peut les utiliser ainsi.

(Il manque un exemple. Prendre le même que pour les fontes MF.)

7.16 Comment utiliser une fonte PostScript comme fonte par défaut dans un document LaTeX ?

Comme pour la question précédente.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{   <-> bg3r  }{}
\DeclareFontShape{T1}{toto}{m}{it}{  <-> bg3ri }{}
\DeclareFontShape{T1}{toto}{bx}{n}{  <-> bg3b  }{}
\DeclareFontShape{T1}{toto}{bx}{it}{ <-> bg3bi }{}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{it}
```

```
\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\begin{document}

Dans ce document, la fonte par défaut est une fonte PostScript.
Il y a pas du \textbf{gras}, de l'\textit{italique} et
du \textbf{\textit{gras italique}}.

\end{document}
```

Pour plus de détails sur l'utilisation des fontes PostScript sous LaTeX, on peut consulter la page de????[9].

7.17 Comment utiliser une fonte PostScript de manière ponctuelle dans un document LaTeX (vieille version de la même question — à effacer) ?

Pour certaines fontes, il y a déjà une extension toute faite : c'est le cas des fontes Times ou Helvetica évoqués plus haut. Sinon, c'est un peu plus compliqué.

Attention : je n'ai pas encore essayé ce qui suit.

Supposons que vous ayez une fonte postscript, ie, deux fichier appelés par exemple `toto.afm` et `toto.pfa`.

Si vous n'avez pas de fichier `*.afm`, ça ne va pas marcher : ce sont les métriques, ie, la taille des caractères. Sur PC, on trouve parfois un fichier `*.pfm` à la place, que l'on peut convertir grace à pfm2afm[10] (je n'ai pas essayé). Si vous n'avez rien, on peut en obtenir grace à la commande pf2afm[11], mais le résultat sera de pauvre qualité : tout ce que l'on peut faire, c'est prétendre que la taille des caractères est exactement celle de la plus petite boite qui les contient ; pour des caractères très penchés, l'effet est desastreux (j'ai déjà essayé).

Si vous avez un fichier `*.pfb` à la place d'un fichier `*.pfa`, ce n'est pas grave, ils contiennent exactement la même chose, sous des formats légèrement différents. On peut passer de l'un à l'autre à l'aide des commandes pfb2pfa[12] ou pfb2pfa[13] (je sais, elles ont toutes deux le même nom). (je n'ai jamais essayé.)

Tapons maintenant

```
afm2tfm toto toto -v toto -t T1.enc
vp2vf toto toto
```

et rajoutons la ligne suivante dans le fichier `psfonts.map`

```
toto toto "T1Encoding ReEncodeFont" <T1.enc <toto.pfa
```

On peut dès lors déclarer la fonte de la manière suivante (on peut mettre les lignes suivantes dans un fichier `t1toto.fd`)

```
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{ <-> toto }{}
```

On peut remplacer `m` par `bx` pour avoir du gras étendu ou `n` par `it` pour avoir de l'italique. Il peut aussi y avoir plusieurs lignes de ce type.

```
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{  <-> toto   }{}
\DeclareFontShape{T1}{toto}{m}{it}{ <-> totoit }{}
```

9. http://www-theorie.physik.unizh.ch/~ichbin/texfonts/
10. ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/pfm2afm
11. ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/pf2afm
12. ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/ps2pk/ps2pk15/misc/pfb2pfa.c.gz
13. ftp://ftp.lip6.fr/pub/TeX/CTAN/fonts/utilities/ps2mf/pfb2pfa.c.gz

On peut utiliser notre fonte de deux manières différentes : soit comme la fonte principale du document,

```
\renewcommand{\encodingdefault}{T1}
\renewcommend{\rmdefault}{toto}
```

soit de manière ponctuelle, tant dans le texte qu'en mode mathématique

```
\newcommand{\texttoto}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathtoto}{toto}
```

Voici le même exemple, complet (non, testé).

```
\documentclass[a4paper,10pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{ <-> toto }{}
\newcommand{\texttoto}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathtoto}{toto}

\begin{document}
ABC \texttoto{ABC} $A \neq \mathtoto{A}$
\end{document}
```

Il est peut-être possible de visualiser le résultat avec xdvi, mais je ne sais pas comment. Il s'agit soit de patcher le script MakeTeXPK (qui ne s'appelle d'ailleurs plus ainsi, il n'y a plus que des minuscules et certaines lettres ont disparu), pour que les fichiers `*.pk` nécessaires soient crés au vol, soit de les créer préalablement, à la main. Le problème est donc le suivant : comment créer des fichiers `*.pk` à partir de fichiers `*.pfa` ? gsftopk ? Oui.

7.18 Comment pencher une fonte PostScript ?

Rajouter l'option `-s .3` (avec éventuellement un autre chiffre — c'est la tangente de l'angle d'inclinaison désiré) quand on lance `afm2tfm`.

```
afm2tfm toto.afm -t ec.enc -s .3 -v toto.vpl REC_toto.tfm
vptovf toto.vpl
```

Dans le fichier `psfonts.map`, on rajoute une ligne du genre

```
REC_toto Toto " .3 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <toto.pfb
```

***** RAJOUTER UN EXEMPLE *****

7.19 Comment élargir ou condenser une fonte PostSCript ?

Rajouter l'option `-e 1.5` (avec éventuellement un autre chiffre — c'est le facteur d'élargissement) quand on lance `afm2tfm`.

```
afm2tfm toto.afm -t ec.enc -e 1.5 -v toto.vpl REC_toto.tfm
vptovf toto.vpl
```

Dans le fichier `psfonts.map`, on rajoute une ligne du genre

```
REC_toto Toto " 2.5 ExtendFont TeXBase1Encoding ReEncodeFont" <8r.enc <toto.pfb
```

***** RAJOUTER UN EXEMPLE *****

7.20 Comment récupérer une fonte PostScript utilisée dans un fichier PDF ?

Une manière simple de procéder consiste à le convertir préalablement en PostScript, à l'aide de Ghostview (`pdf2ps`) ou d'Acrobat.

```
for i in *pdf
do
  pdf2ps $i $i.ps
done
rename 's/.pdf.ps$/.ps/' *.pdf.ps
```

Vous constaterez probablement (comme moi) que les fihciers PostScript que l'on obtient sont très étranges et que les suggestions de la question suivante inefficaces. ***** A DETAILLER *****

7.21 Comment récupérer une fonte PostScript utilisée dans un fichier PostScript ?

***** A FAIRE ***** Avant tout, il convient de rappeler que si les fontes sont commerciales, c'est bien évidemment illégal et que les fontes présentes dans un fichier PostScript sont souvent incomplètes et ne comportent que les caractères effectivement utilisés dans le fichier.

Je suppose que le fichier est bien structuré, avec des commentaires indiquant le début des fontes. On le coupe alors en morceaux :

```
csplit -k weinstein.ps '/^\%\!FontType1/' '/^\%\!FontType1/' '/^\%\!FontType1/' '/^\%\!FontType1/' '/^\%\!FontT
rm xx00
```

On coupe ensuite juste après la commande `cleartomark` (car on constate que les fichiers `*.pfa` se terminent ainsi).

```
rename 's/$/.pfa/' xx[0-9][0-9]
for i in *pfa
do
  perl -n -i -e 'BEGIN{$ok=(0==0)} print if $ok; $ok=(0==1) if m/cleartomark/;'
$i
done
```

```
for i in xx*.pfa
do
  name=`grep '^/FontName' $i | awk '{print $2}' | perl -p -e 's|/||'`
  mv $i $name.pfa
done
```

Vous avez maintenant vos nouvelles fontes PostScript. Il peut y avoir un problème avec les métriques (qui sont absentes) et peut-être aussi avec le codage.

Chapitre 8

Fontes True Type (TTF)

8.1 Qu'est-ce qu'une fonte True Type ?

Voir le manuel de référence[1], chez Adobe, ou la documentation de MiniMou[2].

8.2 Comment créer une fonte True Type ? Comment modifier une fonte True Type existante ?

Après avoir lu le manuel de référence, vous pouvez vous tourner vers ttf_edit[3] (logiciel commercial) ou certains produits MiniMou[4].

8.3 Comment visualiser une fonte True Type ?

Je suppose que FreeType[5] a été installé :

```
tar zxvf freetype.tar.gz
cd freetype
./configure --prefix=$HOME/gnu --bindir=$HOME/gnu/bin/‘uname‘ --libdir=$HOME/gnu
/lib/‘uname‘
make
make install
cd contrib/ttf2pk
./configure --prefix=$HOME/gnu --bindir=$HOME/gnu/bin/‘uname‘ --libdir=$HOME/gnu
/lib/‘uname‘
make
make install
```

(Il y a une erreur dans les lignes précédentes : il faut demander à `ttf2pk` d'utiliser `kpathsea`. Pour vérifier que, contrairement à moi, vous l'avez bien fait, lancez `ttf2tfm -version`).

Pour pouvoir utiliser les fontes TTF de manière presque transparente, sans avoir à lancer soi-même `ttf2pk`, on peut modifier le script `mktexpk` (j'utilise web2c 7.2), à l'aide du patch qui se trouve dans `freetype-current/contrib/ttf2pk/scripts/web2c-7.2/`. (Sur certaines distributions récentes, ce n'est pas nécessaire, car cela a déjà été fait.)

```
cd freetype-current/contrib/ttf2pk/scripts/web2c-7.2/
patch =mktexpk mktexpk.diff
```

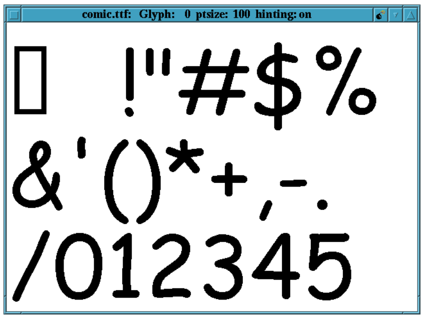
Néanmoins, dans les exemples qui suivent, je créerai les fichiers bitmap explicitement.

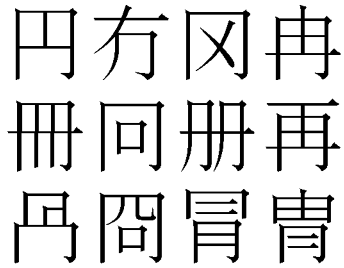
On peut maintenant visualiser des fontes TrueType à l'aide de la commande `ftview`.

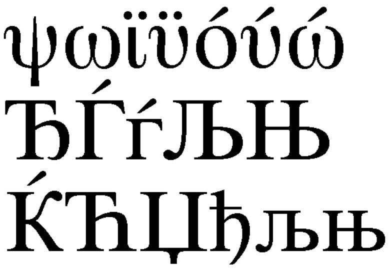
1. http://fonts.apple.com/TTRefMan/index.html
2. http://www.microsoft.com/typography
3. http://truetex.com/ttf_edit.htm
4. ftp://ftp.microsoft.com/developr/drg/TrueType
5. http://www.freetype.org/

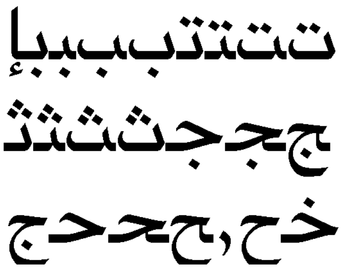
```
ftview 100 toto.ttf
```

Utilisez les touches `o` et `i` pour passer à la ligne suivante ou précédente et la touche `q` pour quitter le programme.









ざしじす
ずせぜそ
ぞただち

넷넸넹녀
녁년녈념
녑녔녕녘

8.4 Comment réaliser un catalogue de fontes True Type?

La situation est la suivante : vous avez récupéré quelques dizaines de fontes TTF et vous aimeriez pouvoir les comparer les unes aux autres. On peut utiliser le script suivant (à lancer dans le répertoire contenant les fontes) qui crée des images GIF pour chacune des fontes et un fichier HTML les reliant. Il ne tient pas compte des fontes ne comportant pas de lettres accentuées et indique les fontes « sans kerning », donc a priori de moindre qualité.

```
#! perl -w

use strict;
use constant TRUE  => 0==0;
use constant FALSE => 0==1;

# Liste des fichiers ttf
opendir(DIR, ".");
my @liste = grep /\.ttf$/, readdir(DIR);
closedir(DIR);

open(HTML,">Catalogue.html") || die "Cannot open Catalogue.html for writing";
print HTML "<HTML><HEAD><TITLE>Catalogue de fontes TTF</TITLE></HEAD><BODY>";
foreach my $ttf (@liste){
  my $base = $ttf;
  $base =~ s/\.ttf$//i;
  my $tex  = "$base.tex";
  my $dvi  = "$base.dvi";
  my $ps   = "$base.ps";
  my $gif  = "$base.gif";

  print "*** PROCESSING $ttf\n";

  open(A, "ttf2pt1 -A $ttf - 2>&1 |") || die "Cannot run ttf2pt1";
  my $kerning = TRUE;
  my $accents = FALSE;
  while(<A>){
    $kerning = FALSE if m/No Kerning data/;
    $accents = TRUE  if m/; N OE ;/;
  }
  close(A);
```

```
  # S'il n'y a pas de lettres accentuées, on oublie la fonte
  next unless $accents;

  system "ttf2tfm $ttf -q -T T1-WGL4.enc -v $base.vpl REC_$base.tfm"; next if $? != 0;
  system "ttf2tfm $ttf -q -T T1-WGL4.enc -v $base.vpl REC_$base.tfm | tail -1 >> ttfonts.map";
  next if $? != 0;
  system "vptovf $base.vpl $base.vf $base.tfm";
  system "ttf2pk -q REC_$base 720";

  system "cp template.tex $tex";
  system "perl -p -i -e 's/toto/$base/g' $tex";
  system "latex $tex";
  system "xdvi $dvi";
  system "dvips -E -f $dvi -o $ps";
  system "pstogif -out $gif -density 288 $ps"; # 288 = 4*72

  unlink "$base.log", "$base.tfm", "$base.tex", "$base.aux", "$base.dvi";
  unlink "$base.ps", "$base.aux";
  unlink "$base.100pk", "$base.360pk", "$base.600pk", "$base.720pk";

  print HTML "$ttf ";
  print HTML "(no kerning) " unless $kerning;
  print HTML "<P> <IMG SRC=\"$gif\"> <P> <HR>\n";
}

print HTML "<tt>Automatically generated on ". `date` ."</TT>\n";
print HTML "</BODY></HTML>\n";
close(HTML);
```

Le fichier `Template.tex` est le suivant. Vous pouvez remplacer le texte par ce que vous voulez. Ici, je commence par écrire « Hom » avec les fontes Computer Modern avant de passer à la fonte à tester, afin de mieux pouvoir les comparer.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}
{ <-> toto }{}
\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\parindent=0pt
\pagestyle{empty}
\begin{document}
\obeylines
Hom \TOTO Hom ABCabc0123\oe ff ffi ffl fi \'Z éèçàùïëü
ZOONEKYND Vincent
Veuillez agréer l'expression de mes sentiments les meilleurs.
\end{document}
```

Si on n'a pas la commande `pstogif` sous la main (ce doit être un morceau de GhostScript ou de LaTeX2html), on peut convertir une image PS en une image GIF de la manière suivante (remplacer 72 par n*72, avec n suffisemment grand pour avoir des images GIF lisibles).

```
my ($bbx,$bby,$bbw,$bbh);
open(PS,"$tmp.ps");
while (<PS>) {
  if (/^%%BoundingBox:\s+(-?\d+)\s+(-?\d+)\s+(-?\d+)\s+(-?\d+)/) {
    $bbx = 0-$1;    $bby = 0-$2;
    $bbw = $3+$bbx;    $bbh = $4+$bby;
    last if /^%%EndComments/;
  }
}
close(PS);

open(GS, "|gs -q -dNOPAUSE -dNO_PAUSE -sDEVICE=ppmraw -g${bbw}x${bbh} -r72 -sOutputFile=$tmp.ppm");
print GS "$bbx $bby translate\n";
print GS "($tmp.ps) run\n";
print GS "showpage\n";
print GS "quit\n";
close(GS);
```

J'ai mis un exemple de résultat sur le Web [6].

8.5 Comment utiliser une fonte True Type de manière ponctuelle dans un document LaTeX ?

Si vous voulez tout de suite une bonne réponse, passez à la question suivante. Nous allons procéder très lentement, en commettant quelques erreurs, que nous corrigerons plus loin.

On part d'un fichier `toto.ttf`. Pour se mettre en confiance, on peut regarder ce qu'il contient (ce n'est pas nécessaire, mais néanmoins rassurant).

```
ftdump toto.ttf
```

Le résultat nous dit entre autres quel sont les codages utilisés (eh oui, il y a généralement plusieurs codages dans une fonte True Type), par exemple comme suit.

```
character map encodings
-----------------------------------------------
There are 3 encodings:

encoding  0: Apple Unicode
encoding  1: Apple Roman
encoding  2: Windows Unicode
```

Si on voit marqué `Windows Unicode`, ça va. (Cela revient à donner des paramètres `-P 3 -E 1` à `ttf2tfm`, mais ce sont les valeurs par défaut.) Sinon, regarder dans le fichier `ttf2pk.doc` pour plus d'information.

Toujours pour se rassurer, on peut regarder la forme des lettres, à l'aide de la commande `ftview`

```
ftview 100 toto.ttf
```

Créons maintenant les métriques et le fichier de configuration `ttf2pk`. Je suppose ici que la fonte contient en fait des caractères latins (pas d'idéogrammes asiatiques).

```
ttf2tfm toto.ttf -T EC.enc | tail -1 > ttfonts.map
```

Il faut préciser le codage utilisé (sinon, le résultat peut être étrange...). La commande `tail` récupère la dernière ligne, qui contient quelques informations sur la fonte nécessaires à `ttf2pk`, pour la mettre dans le fichier de configuration de `ttf2pk`.

On crée maintenant le fichier `toto.300pk`.

6. `http://www.math.jussieu.fr/~zoonek/LaTeX/TTF/Catalogue.html`

```
ttf2pk toto 300
```

Pour se rassurer, on peut vérifier que les caractères sont bien là et les visualiser.

```
pk2gf toto.300pk
gftodvi toto.300gf
xdvi toto.dvi
```

On essaie maintenant de les utiliser...

```
\documentclass[a4paper,10pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
   \DeclareFontShape{T1}{toto}{m}{n}
           { <-> toto }{}
   \DeclareSymbolFont{toto}{T1}{toto}{m}{n}
   \DeclareSymbolFontAlphabet{\mathtoto}{toto}
   \newcommand{\texttoto}[1]
   {{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}

\begin{document}
ABC \texttoto{ABC}
\end{document}
```

ABC **ABCœéèçù**

Cela semble correct : on a même les lettres accentuées. Mais il n'y a pas de « kerning ». Dans beaucoup de fontes True Type, il n'y a pas ces informations : c'est le cas de l'exemple ci-dessus (c'est une fonte Microsoft). Mais dans l'exemple ci-dessous, ces informations sont présentes : dans la première ligne, LaTeX utilise les informations de kerning dont il dispose, mais pas dans la seconde.

AVAL
AVAL

On ne voit aucune différence : ces informations se sont perdues quelque part en chemin... Dans la question suivante, nous verrons comment les récupérer.

Il y a en fait un autre problème, beaucoup plus visible : il manque certains caractères, certaines ligatures, plus précisément. Dans l'exemple suivant, on devrait voir un trait d'union, un tiret moyen et un tiret long ; à la fin, on devrait dicerner des différences entre les ligatures fi ou fl et ces même lettres bien séparées.

ABCœéèçù - -- --- << >> fi fi fl fl

8.6 Comment utiliser une fonte True Type de manière ponctuelle dans un document LaTeX ?

Nous allons maintenant voir comment utiliser correctement des fontes True Type, sans perdre d'information, comme par exemple le kerning de l'exemple précédent.

On demande à `ttf2tfm` de créer des fontes virtuelles, en utilisant le codage `T1-WGL4` (qui correspond à une traduction Unicode–>TTF (???)). Ces fontes virtuelles sont crées sous forme de fichiers `*.vpl`, qu'il faut encore traduire à l'aide de `vptovf`. On peut automatiser les choses de la manière suivante.

```
for i in *ttf
do
  j=‘echo $i | sed ’s/\.ttf$//’‘
  ttf2tfm $i -q -T T1-WGL4.enc -v EC_$j.vpl REC_$j.tfm >> ttfonts.map
  vptovf EC_$j.vpl EC_$j.vf EC_$j.tfm
done
```

On peut dès lors les utiliser, comme d'habitude.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{ <-> EC_toto }{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}
\begin{document}
\huge\TOTO         AVAL

               {A}{V}{A}{L}
\end{document}
```

Pour créer les fichiers `*.pk` :

```
for i in ‘sed ’s/ .*//’ ttfonts.map‘
do
  ttf2pk $i 1496
done
```

Voici le résultat.

AVAL
AVAL

On remarque aussi que nos histoires de ligatures sont résolues.

ABCœéèçù - - — « » fi fi fl fl

8.7 Comment utiliser plusieures fontes True Type de manière ponctuelle dans un document LaTeX ?

C'est exactement comme pour les fontes Métafont. Voici un exemple. Je suppose que l'on a des fichiers `galirdn.ttf`, `galirdi.ttf`, `galirdb.ttf` et `galirdbi.ttf`

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{   <-> galirdn }{}
\DeclareFontShape{T1}{toto}{m}{it}{  <-> galirdi }{}
```

```
\DeclareFontShape{T1}{toto}{bx}{n}{  <-> galirdb }{}
\DeclareFontShape{T1}{toto}{bx}{it}{ <-> galirdbi }{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\begin{document}

\textTOTO{ABCabc \textbf{ABCabc} \textit{ABCabc} \textbf{\textit{ABCabc}}}

\end{document}
```

Pour créer les métriques et les fichiers bitmap, on peut procéder ainsi.

```
for i in *ttf
do
  j=`echo $i | sed 's/\.ttf$//'`
  ttf2tfm $i -q -T T1-WGL4.enc -v $j.vpl REC_$j.tfm >> ttfonts.map
  vptovf $j.vpl $j.vf $j.tfm
done

for i in `sed 's/ .*//' ttfonts.map`
do
  ttf2pk -q $i 720
done
```

Voici le résultat.

ABCabc **ABCabc**
ABCabc ***ABCabc***

8.8 Comment utiliser plusieures fontes True Type comme fontes par défaut dans un document LaTeX ?

C'est comme dans la question précédente, mais il faut rajouter les lignes suivantes.

```
\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
%\renewcommand{\sfdefault}{pss}
\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{it}
```

On remarquera que dans cet exemple, il n'y a pas de fonte sans sérif : on peut par exemple prendre Helvetica (il conviendrait de vérifier que ces fontes se marrient bien et, le cas échéant, de les corriger, par exemple à l'aide de fontes virtuelles), en rajoutant la ligne

```
\usepackage{helvet}
```

Cette ligne est écrite à l'aide d'une fonte True Type. Il y a de l'*italique*, du **gras** et du***gras italique***.

8.9 Comment pencher une fonte True Type?

Il suffit donner l'option `-s 0.25` (ou toute autre valeur) à `ttf2tfm` et de le préciser dans le fichier `ttfonts.map`, de la manière suivante.

```
  REC_comic           comic.ttf            Encoding=T1-WGL4.enc
  REC_comic_slanted   comic.ttf Slant=0.25 Encoding=T1-WGL4.enc
```

On peut automatiser tout cela.

```
for i in *ttf
do
  j=`echo $i | sed 's/\.ttf$//'`
  ttf2tfm $i -q -T T1-WGL4.enc -v $j.vpl REC_$j.tfm >> ttfonts.map
  ttf2tfm $i -q -T T1-WGL4.enc -s .25 -v slant_$j.vpl REC_slant_$j.tfm >> ttfonts.map
  vptovf $j.vpl $j.vf $j.tfm
  vptovf slant_$j.vpl slant_$j.vf slant_$j.tfm
done

for i in `sed 's/ .*//' ttfonts.map`
do
  ttf2pk -q $i 720
done
```

Essayons enfin de les utiliser.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{  <-> comic }{}
\DeclareFontShape{T1}{toto}{m}{sl}{ <-> slant_comic }{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
%\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{sl}

\begin{document}

ABCabc \textbf{ABCabc} \textit{ABCabc} \textbf{\textit{ABCabc}}

\end{document}
```

ABCabc ABCabc
ABCabc ABCabc

Comme toutes les fontes que nous avons demandées ne sont pas là, LaTeX a été amené à effectuer certaines substitutions. Par exemple, en bas à gauche, nous voulions du gras italique. Comme il n'y en a pas, il a commencé par chercher du gras droit. Comme il n'y en a pas non plus, il a pris du maigre droit.

À l'aide de la commande `sub` (et pas `ssub` : la différence, c'est que les messages d'erreur s'affichent), nous pouvons contrôler ces substitutions, en demandant à LaTeX de remplacer le gras italique par de l'italique (et non plus du gras).

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{   <-> comic           }{}
\DeclareFontShape{T1}{toto}{m}{sl}{  <-> slant_comic     }{}
\DeclareFontShape{T1}{toto}{bx}{n}{  <-> sub * toto/m/n  }{}
\DeclareFontShape{T1}{toto}{bx}{sl}{ <-> sub * toto/m/sl }{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
%\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{sl}

\begin{document}

ABCabc \textbf{ABCabc} \textit{ABCabc} \textbf{\textit{ABCabc}}

\end{document}
```

ABCabc ABCabc
ABCabc ABCabc

8.10 Comment condenser ou élargir une fonte True Type ?

Il suffit donner l'option `-e 0.7` (ou toute autre valeur) à `ttf2tfm` et de le préciser dans le fichier `ttfonts.map`, de la manière suivante.

```
  REC_comic           comic.ttf              Encoding=T1-WGL4.enc
  REC_condensed_comic comic.ttf Extend=0.7  Encoding=T1-WGL4.enc
  REC_wide_comic      comic.ttf Extend=1.5  Encoding=T1-WGL4.enc
```

On peut automatiser tout cela.

```
for i in *ttf
do
  j=‘echo $i | sed ’s/\.ttf$//’‘
  ttf2tfm $i -q -T T1-WGL4.enc -v $j.vpl REC_$j.tfm >> ttfonts.map
  ttf2tfm $i -q -T T1-WGL4.enc -e .7 -v condensed_$j.vpl REC_condensed_$j.tfm >> ttfonts.map
  ttf2tfm $i -q -T T1-WGL4.enc -e .7 -v wide_$j.vpl REC_wide_$j.tfm >> ttfonts.map
  vptovf $j.vpl $j.vf $j.tfm
  vptovf condensed_$j.vpl condensed_$j.vf condensed_$j.tfm
  vptovf wide_$j.vpl wide_$j.vf wide_$j.tfm
done
```

```
for i in `sed 's/ .*//' ttfonts.map`
do
  ttf2pk -q $i 720
done
```

Essayons enfin de l'utiliser.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{  <-> comic }{}
\DeclareFontShape{T1}{toto}{m}{sl}{ <-> condensed_comic }{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
%\renewcommand{\bfdefault}{bx}
\renewcommand{\itdefault}{sl}

\begin{document}

ABCabc \textbf{ABCabc} \textit{ABCabc} \textbf{\textit{ABCabc}}

\end{document}
```

ABCabc

ABCabc

ABCabc

8.11 Comment avoir des petites capitales à l'aide d'une fonte TTF ?

Il s'agit ici d'obtenir de fausses petites capitales : on prend simplement des capitales plus petites. Cette solution n'est pas entièrement satisfaisante, car l'épaisseur des traits, l'espace entre les lettres, le gris de la fonte vont correspondre avec ceux d'un corps inférieur et risquent de mal se marrier avec ceux de la fonte de départ. Nous verrons plus loin comment tenter de remédier à ce problème.

Pour obtenir ces fausses petites capitales, il suffit d'utiliser l'option `-V` de `ttf2tfm` à la place de `-v`. On peut automatiser cela de la manière suivante.

```
for i in *ttf
do
  j=`echo $i | sed 's/\.ttf$//'`
  ttf2tfm $i -q -T T1-WGL4.enc -v $j.vpl REC_$j.tfm >> ttfonts.map
  ttf2tfm $i -q -T T1-WGL4.enc -V SC_$j.vpl REC_SC_$j.tfm >> ttfonts.map
  vptovf $j.vpl $j.vf $j.tfm
  vptovf SC_$j.vpl SC_$j.vf SC_$j.tfm
done
```

```
for i in `sed 's/ .*//' ttfonts.map`
do
  ttf2pk -q $i 720
done
```

On ne voit rien de changé dans le fichier `ttfonts.map` :

```
REC_comic      comic.ttf  Encoding=T1-WGL4.enc
REC_SC_comic   comic.ttf  Encoding=T1-WGL4.enc
```

On peut ensuite les utiliser de la manière suivante.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{  <-> comic      }{}
\DeclareFontShape{T1}{toto}{m}{sc}{ <-> SC_comic  }{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\pagestyle{empty}
\begin{document}
\huge
\TOTO    ABCabc
\scshape ABCabc
\end{document}
```

Voici le résultat.

ABCabc ABCabc

8.12 Comment épaissir une fonte TTF?

Une première solution consiste à l'étendre, horizontalement (8.10).

Les noms de catégorie seront écrits dans une fonte gothique (par exemple $\mathfrak{Ens}$ pour la catégorie des ensembles, $\mathfrak{Sh}\,X$ pour la catégorie des faisceaux étales sur X) et les 2-catégories par une fonte gothique grasse (par exemple **Cat** pour la 2-catégorie des catégories ou **Groupoïdes** pour celle des groupoïdes).

Nous n'utiliserons pas de fonte gothique italique (*Ens*) ou gothique italique grasse (***Groupoïdes***).

Une autre solution consiste à utiliser fontinst (9.17).

8.13 Comment automatiser les réponses à toutes les questions précédentes?

Pour une fonte isolée, si on choisit d'utiliser une version étendue à la place du gras, on peut procéder ainsi.

```
#!/bin/sh
```

```
rm ttfonts.map

run () {
  echo -- "$*" 2>&1
  $*
}

doit () {
  name=$1
  shift
  base=`echo $name | sed 's/\.ttf$//'`
  prefix=$1
  shift
  echo "-- ttf2tfm $base.ttf -q -T T1-WGL4.enc $* ${prefix}_$base.vpl REC_${prefix}_$base.tfm >> ttfonts.map"
  ttf2tfm $base.ttf -q -T T1-WGL4.enc $* ${prefix}_$base.vpl REC_${prefix}_$base.tfm >> ttfonts.map
  echo "-- vptovf ${prefix}_$base.vpl ${prefix}_$base.vf ${prefix}_$base.tfm"
  vptovf ${prefix}_$base.vpl ${prefix}_$base.vf ${prefix}_$base.tfm
}

for i in *ttf
do
  doit $i normal -v
  doit $i condensed -e .7 -v
  doit $i wide -e 2 -v
  doit $i slant -s .25 -v
  doit $i slant_condensed -s .25 -e .7 -v
  doit $i slant_wide -s .25 -e 2 -v
  doit $i SC -V
  doit $i SC_condensed -e .7 -V
  doit $i SC_wide -e 2 -V
  doit $i SC_slant -s .25 -V
  doit $i SC_slant_condensed -s .25 -e .7 -V
  doit $i SC_slant_wide -s .25 -e 2 -V

  j=`echo $i | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
  cp Template.sty $j.sty
  perl -p -i -e "s/toto/$j/g" $j.sty
  cp Sample.tex $j.tex
  perl -p -i -e "s/toto/$j/g" $j.tex
done

for i in `sed 's/ .*//' ttfonts.map`
do
  for dpi in 600 720 1244 1496 1194 576
  do
    ttf2pk -q $i $dpi
  done
done

for i in *ttf
do
  j=`echo $i | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
  latex $j
done
```

Le fichier `Template.sty` est le suivant.

```
\ProvidesFile{toto.sty}
```

```
\ProvidesPackage{toto}
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{    <-> normal_toto           }{}
\DeclareFontShape{T1}{toto}{m}{sl}{   <-> slant_toto     }{}
\DeclareFontShape{T1}{toto}{m}{sc}{   <-> SC_toto       }{}
\DeclareFontShape{T1}{toto}{m}{slsc}{ <-> SC_slant_toto  }{}

\DeclareFontShape{T1}{toto}{c}{n}{    <-> condensed_toto           }{}
\DeclareFontShape{T1}{toto}{c}{sl}{   <-> slant_condensed_toto     }{}
\DeclareFontShape{T1}{toto}{c}{sc}{   <-> SC_condensed_toto        }{}
\DeclareFontShape{T1}{toto}{c}{slsc}{ <-> SC_slant_condensed_toto  }{}

\DeclareFontShape{T1}{toto}{x}{n}{    <-> wide_toto           }{}
\DeclareFontShape{T1}{toto}{x}{sl}{   <-> slant_wide_toto     }{}
\DeclareFontShape{T1}{toto}{x}{sc}{   <-> SC_wide_toto        }{}
\DeclareFontShape{T1}{toto}{x}{slsc}{ <-> SC_slant_wide_toto  }{}

\def\encodingdefault{T1}
\def\familydefault{toto}
\def\itdefault{sl}
\def\bfdefault{x}

\def\xseries{\fontseries{x}\selectfont}
\def\cseries{\fontseries{c}\selectfont}

\ifx\undefined\NFSSscshape
  \let\NFSSscshape\scshape
  \let\NFSSitshape\itshape
  \let\NFSSslshape\slshape
  \def\scshape{%
    \def\tempa{it}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{sl}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSscshape
    \fi\fi\fi}
  \def\itshape{%
    \def\tempa{sc}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSitshape
    \fi\fi}
  \def\slshape{%
    \def\tempa{sl}%
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSslshape
    \fi\fi}
  \def\scslshape{\fontshape{slsc}\selectfont}
  \let\slscshape\scslshape
  \def\textscsl#1{{\scslshape\relax #1}}
  \let\textslsc\textscsl
\fi
\endinput
```

Le fichier `Sample.tex` est le suivant.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{toto}
\def\test#1#2{{\fontseries{#1}\fontshape{#2}\selectfont ABCabc}}
\begin{document}

\test{m}{n}
\test{m}{sl}
\test{m}{sc}
\test{m}{slsc}

\test{c}{n}
\test{c}{sl}
\test{c}{sc}
\test{c}{slsc}

\test{x}{n}
\test{x}{sl}
\test{x}{sc}
\test{x}{slsc}

Normal
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}

{\bfseries
Gras
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\cseries
Condensed
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

\end{document}
```

Pour une fonte composée de deux variantes, maigre et grasse, on peut procéder ainsi.

```
#!/bin/sh

# On a deux fontes : une maigre et une grasse.
# Elles sont données en argument.

maigre=$1
grasse=$2

echo Processing fonts $1 and $2 1>&2

rm ttfonts.map

run () {
  echo -- "$*" 2>&1
  $*
}
```

```
doit () {
  name=$1
  shift
  base=`echo $name | sed 's/\.ttf$//'`
  prefix=$1
  shift
  echo "-- ttf2tfm $base.ttf -q -T T1-WGL4.enc $* ${prefix}_$base.vpl REC_${prefix}_$base.tfm >> ttfonts.map"
  ttf2tfm $base.ttf -q -T T1-WGL4.enc $* ${prefix}_$base.vpl REC_${prefix}_$base.tfm >> ttfonts.map
  echo "-- vptovf ${prefix}_$base.vpl ${prefix}_$base.vf ${prefix}_$base.tfm"
  vptovf ${prefix}_$base.vpl ${prefix}_$base.vf ${prefix}_$base.tfm
}

for i in $maigre $grasse
do
  doit $i normal -v
  doit $i condensed -e .7 -v
  doit $i wide -e 2 -v
  doit $i slant -s .25 -v
  doit $i slant_condensed -s .25 -e .7 -v
  doit $i slant_wide -s .25 -e 2 -v
  doit $i SC -V
  doit $i SC_condensed -e .7 -V
  doit $i SC_wide -e 2 -V
  doit $i SC_slant -s .25 -V
  doit $i SC_slant_condensed -s .25 -e .7 -V
  doit $i SC_slant_wide -s .25 -e 2 -V
done

for i in `sed 's/ .*//' ttfonts.map`
do
  for dpi in 600 720 1244 1496 1194 576
  do
    ttf2pk -q $i $dpi
  done
done

i=`echo $maigre | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
j=`echo $grasse | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
cp Template.sty $i.sty
perl -p -i -e "s/toto/$i/g" $i.sty
perl -p -i -e "s/tutu/$j/g" $i.sty
cp Sample.tex $i.tex
perl -p -i -e "s/toto/$i/g" $i.tex
latex $i.tex
```

Le fichier `Template.sty` est le suivant.

```
   \ProvidesFile{toto.sty}
\ProvidesPackage{toto}
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{    <-> normal_toto    }{}
\DeclareFontShape{T1}{toto}{m}{sl}{   <-> slant_toto     }{}
\DeclareFontShape{T1}{toto}{m}{sc}{   <-> SC_toto        }{}
\DeclareFontShape{T1}{toto}{m}{slsc}{ <-> SC_slant_toto  }{}

\DeclareFontShape{T1}{toto}{c}{n}{    <-> condensed_toto           }{}
\DeclareFontShape{T1}{toto}{c}{sl}{   <-> slant_condensed_toto     }{}
\DeclareFontShape{T1}{toto}{c}{sc}{   <-> SC_condensed_toto        }{}
```

```
\DeclareFontShape{T1}{toto}{c}{slsc}{ <-> SC_slant_condensed_toto  }{}

\DeclareFontShape{T1}{toto}{x}{n}{    <-> wide_toto           }{}
\DeclareFontShape{T1}{toto}{x}{sl}{   <-> slant_wide_toto     }{}
\DeclareFontShape{T1}{toto}{x}{sc}{   <-> SC_wide_toto        }{}
\DeclareFontShape{T1}{toto}{x}{slsc}{ <-> SC_slant_wide_toto  }{}

\DeclareFontShape{T1}{toto}{b}{n}{    <-> normal_tutu    }{}
\DeclareFontShape{T1}{toto}{b}{sl}{   <-> slant_tutu     }{}
\DeclareFontShape{T1}{toto}{b}{sc}{   <-> SC_tutu        }{}
\DeclareFontShape{T1}{toto}{b}{slsc}{ <-> SC_slant_tutu  }{}

\DeclareFontShape{T1}{toto}{bc}{n}{    <-> condensed_tutu           }{}
\DeclareFontShape{T1}{toto}{bc}{sl}{   <-> slant_condensed_tutu     }{}
\DeclareFontShape{T1}{toto}{bc}{sc}{   <-> SC_condensed_tutu        }{}
\DeclareFontShape{T1}{toto}{bc}{slsc}{ <-> SC_slant_condensed_tutu  }{}

\DeclareFontShape{T1}{toto}{bx}{n}{    <-> wide_tutu           }{}
\DeclareFontShape{T1}{toto}{bx}{sl}{   <-> slant_wide_tutu     }{}
\DeclareFontShape{T1}{toto}{bx}{sc}{   <-> SC_wide_tutu        }{}
\DeclareFontShape{T1}{toto}{bx}{slsc}{ <-> SC_slant_wide_tutu  }{}

\def\encodingdefault{T1}
\def\familydefault{toto}
\def\itdefault{sl}
\def\bfdefault{b}

\def\xseries{\fontseries{x}\selectfont}
\def\cseries{\fontseries{c}\selectfont}
\def\bxseries{\fontseries{bx}\selectfont}
\def\bcseries{\fontseries{bc}\selectfont}

 \def\textx#1{{\xseries #1}}
\def\textc#1{{\cseries #1}}
\def\textbx#1{{\bxseries #1}}
\def\textbc#1{{\bcseries #1}}

\ifx\undefined\NFSSscshape
  \let\NFSSscshape\scshape
  \let\NFSSitshape\itshape
  \let\NFSSslshape\slshape
  \def\scshape{%
    \def\tempa{it}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{sl}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSscshape
    \fi\fi\fi}
  \def\itshape{%
    \def\tempa{sc}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSitshape
    \fi\fi}
  \def\slshape{%
    \def\tempa{sl}%
```

```
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSslshape
    \fi\fi}
  \def\scslshape{\fontshape{slsc}\selectfont}
  \let\slscshape\scslshape
  \def\textscsl#1{{\scslshape\relax #1}}
  \let\textslsc\textscsl
\fi
\endinput
```

Le fichier `Sample.tex` est le suivant.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{toto}
\def\test#1#2{{\fontseries{#1}\fontshape{#2}\selectfont ABCabc}}
\begin{document}

\test{m}{n}
\test{m}{sl}
\test{m}{sc}
\test{m}{slsc}

\test{c}{n}
\test{c}{sl}
\test{c}{sc}
\test{c}{slsc}

\test{x}{n}
\test{x}{sl}
\test{x}{sc}
\test{x}{slsc}

\test{b}{n}
\test{b}{sl}
\test{b}{sc}
\test{b}{slsc}

\test{bx}{n}
\test{bx}{sl}
\test{bx}{sc}
\test{bx}{slsc}

\test{bc}{n}
\test{bc}{sl}
\test{bc}{sc}
\test{bc}{slsc}

Normal
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}

{\xseries
Extended
\textit{Italique}
\textsc{Petites capitales}
```

```
\textslsc{Petites capitales penchées}}

{\cseries
Condensed
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\bfseries
Bold
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\bxseries
Bold Extended
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\bcseries
Bold Condensed
\textit{Italique}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

\end{document}
```

Pour une fonte un peu plus complète, avec maigre, gras, italique et gras italique, on peut procéder ainsi.

```
#!/bin/sh

# On a quatre fontes : maigre droit, gras droit, maigre italique et
# gras italique.
# Elles sont données en argument.

if [ $# -eq 4 ]
then
  echo Processing fonts $1, $2, $3, $4 1>&2
else
  echo I need 4 arguments, not $#
  echo usage: ./doit galirdn.ttf galirdb.ttf galirdi.ttf galirdbi.ttf
  exit 1
fi

maigre=$1
grasse=$2
ital=$3
italgras=$4

rm -f ttfonts.map

run () {
  echo -- "$*" 2>&1
  $*
}

doit () {
  name=$1
```

```
  shift
  base=`echo $name | sed 's/\.ttf$//'`
  prefix=$1
  shift
  echo "-- ttf2tfm $base.ttf -q -T T1-WGL4.enc $* ${prefix}_$base.vpl REC_${prefix}_$base.tfm >
> ttfonts.map"
  ttf2tfm $base.ttf -q -T T1-WGL4.enc $* ${prefix}_$base.vpl REC_${prefix}_$base.tfm >> ttfonts
.map
  echo "-- vptovf ${prefix}_$base.vpl ${prefix}_$base.vf ${prefix}_$base.tfm"
  vptovf ${prefix}_$base.vpl ${prefix}_$base.vf ${prefix}_$base.tfm
}

for i in $maigre $grasse $ital $italgras
do
  doit $i normal -v
  doit $i condensed -e .7 -v
  doit $i wide -e 2 -v
  doit $i slant -s .25 -v
  doit $i slant_condensed -s .25 -e .7 -v
  doit $i slant_wide -s .25 -e 2 -v
  doit $i SC -V
  doit $i SC_condensed -e .7 -V
  doit $i SC_wide -e 2 -V
  doit $i SC_slant -s .25 -V
  doit $i SC_slant_condensed -s .25 -e .7 -V
  doit $i SC_slant_wide -s .25 -e 2 -V
done

for i in `sed 's/ .*//' ttfonts.map`
do
  for dpi in 600 720 1244 1496 1194 576
  do
    ttf2pk -q $i $dpi
  done
done

i=`echo $maigre   | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
j=`echo $grasse   | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
k=`echo $ital     | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
l=`echo $italgras | perl -p -e 's/\.ttf$//;s/[^A-Za-z]//g'`
cp Template.sty $i.sty
perl -p -i -e "s/toto/$i/g" $i.sty
perl -p -i -e "s/tutu/$j/g" $i.sty
perl -p -i -e "s/ITALGRAS/$l/g" $i.sty
perl -p -i -e "s/ITAL/$k/g" $i.sty

cp Sample.tex $i.tex
perl -p -i -e "s/toto/$i/g" $i.tex
latex $i.tex
```

Le fichier `Template.sty` est le suivant.

```
\ProvidesFile{toto.sty}
\ProvidesPackage{toto}
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{    <-> normal_toto    }{}
\DeclareFontShape{T1}{toto}{m}{sl}{   <-> slant_toto     }{}
\DeclareFontShape{T1}{toto}{m}{sc}{   <-> SC_toto        }{}
\DeclareFontShape{T1}{toto}{m}{slsc}{ <-> SC_slant_toto  }{}
```

```
\DeclareFontShape{T1}{toto}{c}{n}{    <-> condensed_toto           }{}
\DeclareFontShape{T1}{toto}{c}{sl}{   <-> slant_condensed_toto     }{}
\DeclareFontShape{T1}{toto}{c}{sc}{   <-> SC_condensed_toto        }{}
\DeclareFontShape{T1}{toto}{c}{slsc}{ <-> SC_slant_condensed_toto  }{}

\DeclareFontShape{T1}{toto}{x}{n}{    <-> wide_toto           }{}
\DeclareFontShape{T1}{toto}{x}{sl}{   <-> slant_wide_toto     }{}
\DeclareFontShape{T1}{toto}{x}{sc}{   <-> SC_wide_toto        }{}
\DeclareFontShape{T1}{toto}{x}{slsc}{ <-> SC_slant_wide_toto  }{}

\DeclareFontShape{T1}{toto}{b}{n}{    <-> normal_tutu    }{}
\DeclareFontShape{T1}{toto}{b}{sl}{   <-> slant_tutu     }{}
\DeclareFontShape{T1}{toto}{b}{sc}{   <-> SC_tutu        }{}
\DeclareFontShape{T1}{toto}{b}{slsc}{ <-> SC_slant_tutu  }{}

\DeclareFontShape{T1}{toto}{bc}{n}{    <-> condensed_tutu           }{}
\DeclareFontShape{T1}{toto}{bc}{sl}{   <-> slant_condensed_tutu     }{}
\DeclareFontShape{T1}{toto}{bc}{sc}{   <-> SC_condensed_tutu        }{}
\DeclareFontShape{T1}{toto}{bc}{slsc}{ <-> SC_slant_condensed_tutu  }{}

\DeclareFontShape{T1}{toto}{bx}{n}{    <-> wide_tutu           }{}
\DeclareFontShape{T1}{toto}{bx}{sl}{   <-> slant_wide_tutu     }{}
\DeclareFontShape{T1}{toto}{bx}{sc}{   <-> SC_wide_tutu        }{}
\DeclareFontShape{T1}{toto}{bx}{slsc}{ <-> SC_slant_wide_tutu  }{}

\DeclareFontShape{T1}{toto}{m}{it}{    <-> normal_ITAL    }{}
\DeclareFontShape{T1}{toto}{c}{it}{    <-> condensed_ITAL }{}
\DeclareFontShape{T1}{toto}{x}{it}{    <-> wide_ITAL      }{}
\DeclareFontShape{T1}{toto}{b}{it}{    <-> normal_ITALGRAS    }{}
\DeclareFontShape{T1}{toto}{bc}{it}{   <-> condensed_ITALGRAS }{}
\DeclareFontShape{T1}{toto}{bx}{it}{   <-> wide_ITALGRAS      }{}

\def\encodingdefault{T1}
\def\familydefault{toto}
\def\itdefault{it}
\def\bfdefault{b}

\def\xseries{\fontseries{x}\selectfont}
\def\cseries{\fontseries{c}\selectfont}
\def\bcseries{\fontseries{bc}\selectfont}

\def\textx#1{{\xseries #1}}
\def\textc#1{{\cseries #1}}
\def\textbx#1{{\bxseries #1}}
\def\textbc#1{{\bcseries #1}}

\ifx\undefined\NFSSscshape
  \let\NFSSscshape\scshape
  \let\NFSSitshape\itshape
  \let\NFSSslshape\slshape
  \def\scshape{%
    \def\tempa{it}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{sl}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSscshape
```

```
    \fi\fi\fi}
  \def\itshape{%
    \def\tempa{sc}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSitshape
    \fi\fi}
  \def\slshape{%
    \def\tempa{sl}%
    \ifx\tempa\f@shape\scslshape\else
    \def\tempa{scsl}%
    \ifx\tempa\f@shape\scslshape\else
    \NFSSslshape
    \fi\fi}
  \def\scslshape{\fontshape{slsc}\selectfont}
  \let\slscshape\scslshape
  \def\textscsl#1{{\scslshape\relax #1}}
  \let\textslsc\textscsl
\fi
\endinput
```

Le fichier `Sample.tex` est le suivant.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{toto}
\def\test#1#2{{\fontseries{#1}\fontshape{#2}\selectfont ABCabc}}
\begin{document}

\test{m}{n}
\test{m}{sl}
\test{m}{it}
\test{m}{sc}
\test{m}{slsc}

\test{c}{n}
\test{c}{it}
\test{c}{sl}
\test{c}{sc}
\test{c}{slsc}

\test{x}{n}
\test{x}{it}
\test{x}{sl}
\test{x}{sc}
\test{x}{slsc}

\test{b}{n}
\test{b}{it}
\test{b}{sl}
\test{b}{sc}
\test{b}{slsc}

\test{bx}{n}
\test{bx}{it}
\test{bx}{sl}
\test{bx}{sc}
\test{bx}{slsc}
```

```
\test{bc}{n}
\test{bc}{it}
\test{bc}{sl}
\test{bc}{sc}
\test{bc}{slsc}

Normal
\textit{Italique}
\textsl{Penché}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}

{\xseries
Extended
\textit{Italique}
\textsl{Penché}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\cseries
Condensed
\textit{Italique}
\textsl{Penché}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\bfseries
Bold
\textit{Italique}
\textsl{Penché}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\bxseries
Bold Extended
\textit{Italique}
\textsl{Penché}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

{\bcseries
Bold Condensed
\textit{Italique}
\textsl{Penché}
\textsc{Petites capitales}
\textslsc{Petites capitales penchées}}

\end{document}
```

Il y a deux problèmes dans ce que nous venons de voir : d'une part, il manque toujours certains caractères, comme le W majuscule avec un point dessus, d'autre part, quand il n'y a pas de fonte grasse, on aimerait pouvoir en bricoler une en « épaississant » les caracatères et pas en les dilatant comme nous l'avons fait. Nous allons voir que les fontes virtuelles eprmettent ce genre de chose. Il y a un autre problème, plus grave : l'absence de fonte mathématique. Il y a aussi un problème avec ces scripts : ils mettent tout dans le répertoire courrant. Idéalement, il faudrait mettre les fichiers dans $TEXMF/fonts/{truetype:vf:tfm}/ et $TEXMF/tex/generic/, modifier le fichier $TEXMF/fonts/truetype/ttfonts.map et relancer texhash.

8.14 Comment faire pour que LaTeX/xdvi/dvips soit capable de calculer les fontes bitmaps tout seul ?

Il suffit d'installer FreeType correctement, en lui demandant d'utiliser kpathsea.

```
./configure --prefix=$HOME/gnu/`uname`/ --with-kpathsea-dir=/usr/lib
```

Les fontes doivent être installées correctement :

- `$TEXMF/fonts/truetype` : les fontes `*.ttf` et le fichier `ttfonts.map`
- `$TEXMF/fonts/tfm` : les métriques `*.tfm`
- `$TEXMF/fonts/vf` : les fontes virtuelles `*.vf`
- `$TEXMF/tex/generic` ou `$TEXMF/tex/latex` : les fichiers `*.sty` et `*.fd`.

Effacer cette question et mettre la réponse plus haut.

Remarque : mettre la même question pour les fontes PostScript. On peut avoir plusieurs fichiers `psfonts.map` : comment ?

Mettre cette question plus haut, avec la TDS.

8.15 Y a-t-il un moyen plus simple d'utiliser des fontes TrueType ? Est-il possible d'utiliser des fontes True Type avec pdfTeX ?

Oui. Une première solution consiste à transformer les fontes True Type en fontes PostScript, par exemple à l'aide de ttf2pt1 [7]. On utilise ici la commande `ttf2pt1` avec l'option `-b` pour qu'elle produise un fichier `*.pfb`. (Je crois qu'il faut une option du genre `-l adobestd` ou `-l latin1`, mais vus les problèmes qui apparaitrons, ce n'est pas grave.)

```
ttf2pt1 -b toto.ttf toto
```

On peut automatiser cela de la manière suivante.

```
for i in *.ttf
do
  j=`echo $i | sed 's/\.ttf$//'`
  run ttf2pt1 -b $j.ttf $j
done
```

Les fontes s'utilisent alors comme d'habitude. On crée les fontes virtuelles et les métriques.

```
rm -f psfonts.map
for i in *.pfb
do
  j=`echo $i | sed 's/\.pfb$//'`
  font=`afm2tfm $j -v $j -t ec.enc`
  vptovf $j
  echo $font \"ECEncoding ReEncodeFont\" \<ec.enc \<$i >> psfonts.map
done
```

Et le fichier LaTeX peut ressembler à cela.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{   <-> comic }{}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}
```

7. `http://www.netspace.net.au/~mheath/ttf2pt1/`

```
%\renewcommand{\bfdefault}{bx}
%\renewcommand{\itdefault}{it}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\pagestyle{empty}

\begin{document}

Une fonte True Type convertie en PostScript...

\end{document}
```

Le problème, c'est que la fonte PostScript ainsi créée est souvent corrompue : c'est pour cette raison que je ne montre pas de résultat (à l'écran, ça passe, mais l'imprimante refuse).

Une autre solution, beaucoup plus satisfaisante, consiste à utiliser pdfTeX — le format PDF reconnait les fontes TrueType. On commence exactement comme pour `ttf2pk`.

```
ttf2tfm comic.ttf -q -T T1-WGL4.enc -v ECcomic.vpl RECcomic.tfm >> ttfonts.map
vptovf ECcomic.vpl ECcomic.vf ECcomic.tfm
```

On lance ensuite `ttf2afm` et `afm2tfm` et on met le résultat dans `pdftex.map` (ou dans un fichier `*.map` que vous chargerez depuis `pdftex.cfg`.

```
  ttf2afm -e T1-WGL4.enc -o RECcomic.afm comic.ttf
  afm2tfm RECcomic.afm -T T1-WGL4.enc |
    perl -p -e "s/(<T1-WGL4.enc)/ <comic.ttf \$1/"  >> psfonts.map
```

On peut automatiser cela de la manière suivante. (Je ne rajoute pas de lignes pour obtenir des variantes penchées ou des petites capitales, car je constate que cela ne marche pas : avec GhostView, la fonte modifiée reste inchangée, avec AcroRead, elle n'est pas reconnue)

```
#!/bin/sh

cp ‘locate /pftex.map | head -1‘ .
rm -f -- ttfonts.map

for i in *ttf
do
  j=‘echo $i | sed ’s/\.ttf$//’‘
  echo ’*** processing font’ $j
  ttf2tfm $i -q -T T1-WGL4.enc -v EC$j.vpl REC$j.tfm >> ttfonts.map
  vptovf EC$j.vpl EC$j.vf EC$j.tfm
  rm -f -- *.vpl

  ttf2afm -e T1-WGL4.enc -o REC$j.afm $j.ttf
  afm2tfm REC$j.afm -T T1-WGL4.enc |
    perl -p -e "s/(<T1-WGL4.enc)/ <$j.ttf \$1/"  >> psfonts.map
done
```

On utilise alors les fontes comme d'habitude.

```
\documentclass{demo}
\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{<->ECcomic}{}
```

```
\def\rmdefault{toto}
\def\encodingdefault{T1}
\begin{document}
  Ce texte est tapé avec une fonte TrueType...
\end{document}
```

8.16 Comment obtenir les caractères manquants ?

A FAIRE

Il s'agit généralement de certaines lettres accentuées (le Z majuscule accent aigu – cela dit, je ne sais pas du tout à quoi il pourrait servir...), qui ne sont pas présentes dans la fonte, ou de certaines ligatures (ct, st), qui sont présentes dans la fonte (mais visiblement pas dans le codage).

Une idée pour obtenir les caractères manquants est d'utiliser des fontes virtuelles, avec `fontinst` et `latin.mtx`, qui va par exemple construire les lettres accentuées manquantes à l'aide des accents (s'il y en a) et des lettres non accentuées.

[Donner un exemple, modifier le script plus haut qui créait un catalogue de fontes TTF.]

On peut constater que dans la fonte `comic.ttf`, il y a un tiret moyen, des ligatures fl et fi : comment les obtenir ?

Chapitre 9

Fontes virtuelles

Avertissement : Je ne manipule pas encore très bien les fontes virtuelles, il y a probablement des erreurs et des manques dans ce qui suit. En particulier, je ne me suis jamais préoccupé des métriques des fontes que je modifiais (par exemple, quand on fait du faux gras, il faut éloigner un peu les caractères). Je ne parle pas non plus de l'utilisation de fontes PostScript (et de leurs métriques, `*.afm`). Je ne parle pas non plus des différents programmes qui existent sous fontinst, par exemple `latin1.mtx` ou `textcomp.mtx`. Je ne parle pas non plus des fichiers de codage `*.etx`.

9.1 Qu'est-ce qu'une fonte virtuelle ? Que fait-on avec ?

C'est une fonte qui n'existe pas vraiment, qui est construite à partir d'autres fontes, auxquelles on prend certains caractères, après d'éventuelles modifications.

Les fontes virtuelles premettent de

- changer le codage d'une fonte
- construire une fonte à partir d'autres fontes, par exemple, prendre les majuscules dans une fontes, les minuscules dans une autres, les chiffres dans une troisième.
- construire des caractères qui ne sont pas là : par exemple, rajouter un accent sur une lettre, enlever un point sur un i ou un j, souligner ou barrer certains caractères, etc.
- construire des fontes qui ne sont pas là : s'il n'y a pas de gras ou pas de petites capitales, on peut les imiter à l'aide de fontes virtuelles (mais c'est du bricolage).

Les fontes virtuelles ne sont pas reconnues par TeX (qui ne connait que des métrique) mais au niveau des programmes qui lisent les fichiers `*.dvi` tels `dvips` ou `xdvi`. Les fontes virtuelles sont présentes sous forme de métriques `*.tfm` et de fichiers `*.vf` qui indiquent comment fabriquer les caractères. Ces fichiers `*.vf` ne sont pas humainement lisibles, on peut les décoder à l'aide de `vftovpl`, qui les transforme en `*.vpl`, comme nous l'avons déjà vu plus haut (3.2).

9.2 Y a-t-il un moyen SIMPLE d'utiliser des fontes virtuelles ?

Non. On peut utiliser directement les fichiers `*.vpl` (c'est très compliqué) ou passer par `fontinst` (c'est compliqué) mais ce n'est jamais simple.

9.3 Qu'est-ce que fontinst ?

C'est un programme en TeX permettant de manipuler, modifier, créer des fontes virtuelles. C'est ce dont nous allons parler à partir de maintenant.

9.4 Comment utilise-t-on fontinst avec des fontes Métafont ?

Tout d'abord Fontinst a besoin de connaître les métriques (de manière plus générale la description) des fontes qu'il va manipuler. Il les attend sous la forme de fichiers `*.afm` ou `*.pl` créés à partir des métriques `*.tfm` par la commande `tftopl`.

```
for i in *.tfm
do
  j=`echo $i | sed 's/\.tfm$//'`
```

```
  tftopl $i > $j.pl
done
```

Il lui faut ensuite un fichier TeX (car fontinst est programmé en TeX), qui dise ce qu'il faut faire avec ces fontes.

```
\input fontinst.sty
\installfonts
...
\endinstallfonts
\bye
```

On met les choses intéressantes dans les pointillés :

```
\installfont{ecorm0500}{ecrm0500,tcrm0500,dostretch}{T19}{T1}{cmor}{m}{n}{<5>}
\installfont{ecorm0600}{ecrm0600,tcrm0600,dostretch}{T19}{T1}{cmor}{m}{n}{<6>}
```

La première ligne définit la fonte `ecorm0500` à partir de `ecrm0600`, `tcrm0600` et `dostretch`. Chacun de ces trois fichiers (il peut y en avoir un nombre quelconque) correspond à une fonte (les deux premiers) ou à des opérations à effectuer sur une fonte (le dernier, `dostretch.mtx`). La programmation avec `fontinst` se résume donc souvent à écrire des fichiers `*.mtx`. Quand un caractère est défini plusieures fois, seule sa première définition est retenue : ainsi, les caractères éventuellement absents de la première fonte sont pris dans les fontes suivantes. Les deux arguments suivants correspondent au codage : généralement, nous mettrons `T1` des deux côtés. L'argument suivant est la famille à laquelle appartient la fonte (je ne sais pas du tout à quoi peut servir cette information). Les trois derniers arguments sont la graisse de la fonte (`m`, `b`, `bx`, etc.), sa forme (`n`, `sl`, `it`, `sc`, etc.) et sa taille.

Dès lors, il ne reste plus qu'à compiler ce fichier et à construire les fontes virtuelles `*.vf` et leurs métriques `*.tfm`.

```
tex toto.tex

for i in *.vpl
do
  vptovf $i
done
```

On peut ensuite utilise les fontes comme d'habitude. Par exemple, si la nouvelle fonte s'appelle `toto`, on peut procéder ainsi.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{ <-> toto }{}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}

\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\pagestyle{empty}

\begin{document}

Voici la nouvelle fonte.

ABCDEFGHIJKLMNOPQRSTUVWXYZ.
abcdefghijklmnopqrstuvwxyz.

\end{document}
```

9.5 Comment utilise-t-on fontinst avec des fontes PostScript ?

C'est comme pour les fontes MétaFont, à la seule différence que les métriques ne sont pas attendues sous la forme de fichiers `*.pl` provenant de `*.tfm`, mais directement sous la forme de `*.afm`.

9.6 Comment utilise-t-on fontinst avec des fontes True Type ?

Si on utilise `ttf2pk`, on peut procéder comme pour les fonte Métafont, si on les a traduites en PostScript, on procède comme pour les fontes PostScript. Dans le permier cas, il arrive que le fichier `*.pl` obtenu à partir des métriques contienne la ligne

```
(CODINGSCHEME T1ENCODING)
```

que `fontinst` ne reconnait pas (il m'est déjà arrivé de trouver `ECENCODING` à la place, qui ne marchait pas non plus). Il suffit de la remplacer par ce qu'il faut.

```
perl -p -i -e 's/(EC|T1)ENCODING/EXTENDED TEX FONT ENCODING - LATIN/' *.pl
```

Une autre solution consiste à définir `ECENCODING` et `T1ENCODING`, par les commandes suivantes.

```
\declareencoding{ECENCODING}{T1}
\declareencoding{T1ENCODING}{T1}
```

9.7 Comment changer le codage d'une fonte ? Comment pencher une fonte ?

Je reprends cet exemple de la documentation de `fontinst`.

```
\input fontinst.sty
\installfonts
\transformfont{toto}{
  \reencodefont{8r}{
  \slantfont{167}{
  \fromafm{ptmr8a}}}}
\endinstallfonts
\bye
```

Cela va juste créer un fichier `toto26.pl`, que l'on peut transformer en `toto.tfm` grace à `pltotf`. Il faut ensuite écrire (soi-même) le fichier `psfonts.map`.

```
toto26 Times-Roman ".167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc
```

Dans la suite, nous n'utiliserons pas la commande `\trandformfont`, nous passerons toujours par des fichiers `*.etx` ou `*.mtx`.

9.8 Exemples simples avec fontinst

`fontinst` part d'une ou plusieures fontes (décrites par des fichiers `*.pl` ou `*.afm`) et leur applique différentes transformations, décrites dans des fichiers `*.mtx`. La syntaxe d'un fichier `*.mtx` est la suivante.

```
\relax
  Quelques commentaires
  (qui seront ignorés)
\metrics
...
\endmetrics
```

Les choses intéressantes sont dans les points de suspension. L'exemple suivant efface le caractère `zero`, puis définit le caractère `zero` comme étant un `a` à la taille de 1000 milièmes (rappelons que quand un caractère est défini plusieures fois, c'est la première qui compte : si on avait oublié d'effacer de `zero`, sa redéfinition aurait été sans effet) et enfin redéfinit le caractère `one` comme étant `b` à la taille 1.5.

```
\relax
\metrics
  \unsetglyph{zero}
  \setglyph{zero}
    \glyph{a}{1000}
  \endsetglyph
  \resetglyph{one}
    \glyph{b}{1500}
  \endresetglyph
\endmetrics
```

ab23456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ.
abcdefghijklmnopqrstuvwxyz.

L'exemple suivant (qui n'est pas de moi) crée un nouveau caractère `@` à partir d'un `O` majuscule et d'un `a`.

```
\relax
\metrics
  \unsetglyph{at}
  \setglyph{at}
    \push
      \moveup{120}
      \movert{160}
      \glyph{a}{1000}
    \pop
    \glyph{O}{1000}
  \endsetglyph
\endmetrics
```

@

9.9 Comment intervertir deux caractères dans une fonte ?

Nous allons intervertir les caractères `A` et `B` de la fonte `ecrm1000`.

```
\input fontinst.sty
\installfonts
\installfont{toto}
           {ecrm1000,intervertir}
           {T1}{T1}{fcm}{m}{n}{10}
\endinstallfonts
\bye
```

Voici le fichier `intervertir.mtx`.

```
\relax
\metrics
  \setglyph{newA}
    \glyph{B}{1000}
  \endsetglyph

  \resetglyph{B}
```

```
    \glyph{A}{1000}
  \endresetglyph

  \resetglyph{A}
    \glyph{newA}{1000}
  \endresetglyph
\endmetrics
```

BACDEFGHIJKLMNOPQRSTUVWXYZ.

Remarque : n'y a-t-il pas une autre manière d'intervertir deux caractères d'une même fonte, en restant sous TeX ?

9.10 Comment aller chercher un caractère dans une autre fonte ?

Il suffit de charger la première fonte, d'effacer le caractère dont on ne veut plus, et enfin de charger la fonte dans laquelle se trouve la nouvelle version du caractère. Nous allons par exemple remplacer le A de la fonte `ecrm1000` par celui de la fonte `ectt1000`. Le fichier `toto.mtx` est le suivant.

```
\relax
\metrics
  \unsetglyph{A}
\endmetrics
```

Voici le fichier `toto.tex`.

```
\input fontinst.sty
\installfonts
\installfont{toto}
            {ecrm1000,toto,ectt1000}
            {T1}{T1}{toto}{m}{n}{10}
\endinstallfonts
\bye
```

ABCDEFGHIJKLMNOPQRSTUVWXYZ.

9.11 Comment modifier les métriques d'un caractère ?

Nous avons déjà vu (9.13) que les différentes choses que l'on rajoutait dans un caractère contribuaient à ses dimensions, à moins que l'on utilise les commandes `\push` et `\pop`. Si cela ne suffit pas, les commandes `\resetdepth`, `\resetwidth`, `\resetitalic` et `\resetheight` permettent de changer la taille d'un caractère. Il y a un exemple dans `dotlessj.mtx` (voir 9.25).

9.12 Quelles sont les dimensions d'une fonte ? Comment les changer ?

Les dimensions sont (parmi) les suivantes : `ascender`, `capheight`, `descender`, `xheight`, `italicslant`, `minimumkern`, `monowidth`, `underlinethickness`, `designsize`, `interword`, `acccapheight`, `maxheight`, `maxdepth`, `digitwidth`, `capstem`, `baselineskip`, `interword`. Dans le cas des fontes mathématiques, on peut aussi accéder aux dimensions `fontdimen(??)`, où il faut remplacer `??` par le numéro de la dimension désirée. On peut changer ces dimensions à l'aide de la commande `\resetint`.

```
\resetint{interword}{\scale{\int{interword}}{240}}
```

***** DONNER UN EXEMPLE *****

9.13 Comment ajouter un trait ?

La commande `\glyphrule` permet de traver un trait, ses arguments sont la largeur et la hauteur. Il faut préalablement s'être placé au bon endroit à l'aide des commande `\moveup` (déplacement vers le haut) et `\movert` (déplacement vers la droite). L'exemple suivant souligne le caractère A. Les unités sont des milièmes.

```
\relax
\metrics
  \resetglyph{A}
  \push
    \moveup{-150}
    \glyphrule{\width{A}}{40}
  \pop
  \glyph{A}{1000}
  \endresetglyph
\endmetrics
```

ABCABC

Les commandes `\push` et `\pop` permettent de ne pas changer l'état graphique (en particulier la position courrante). Ces sont les analogues des commandes PostScript `gsave` et `grestore`. Si on ne les utilisait pas, le résultat serait le suivant.

_ABC_ABC

On peut compliquer l'exemple précédent en encadrant le caractère au lieu de le souligner. La commande `\add` permet d'effectuer une addition de dimensions. On remarquera qu'il faut faire attention à la largeur du caractère (pour l'augmenter, on a simplement rajouté `\movert{150}` au début et à la fin) et à l'épaisseur des traits (certaines des barres ont une longueur de 340 pour cette raison). Pour généraliser cela à d'autres caractères, il faudrait aussi tenir compte de leur profondeur.

```
\relax
\metrics
  \resetglyph{A}
  \movert{150}
  % Barre du bas
  \push
    \moveup{-150}
    \movert{-150}
    \glyphrule{\add{\width{A}}{300}}{40}
  \pop
  % Barre du haut
  \push
    \moveup{150}
    \movert{-150}
    \moveup{\height{A}}
    \glyphrule{\add{\width{A}}{340}}{40}
  \pop
  % Barre de gauche
  \push
    \movert{-150}
    \moveup{-150}
    \glyphrule{40}{\add{\height{A}}{300}}
  \pop
  % Barre de droite
  \push
    \movert{150}
    \moveup{-150}
    \movert{\width{A}}
    \glyphrule{40}{\add{\height{A}}{340}}
  \pop
```

```
  \glyph{A}{1000}
  \movert{150}
  \endresetglyph
\endmetrics
```

ABCABC

9.14 Comment ajouter des instructions PostScript ?

Lors de la définition d'un caractère, on peut rajouter des instructions PostScript grace à la commande `\glyphspecial`. Essyons par exemple de mettre un caractère en rouge.

```
\relax
  Mettre le A en rouge
\metrics
\resetglyph{A}
  \glyphspecial{ps: currentrgbcolor 255 0 0 setrgbcolor}
  \glyph{A}{1000}
  \glyphspecial{ps: 0 0 0 setrgbcolor}
\endsetglyph
\endmetrics
```

Voici la nouvelle fonte.
ABCDEFGHIJKLMNOPQRSTUVWXYZ. abcdefghijklmnopqrstuvwxyz.

9.15 Peut-on ajouter n'importe quelle instruction PostScript ?

Oui, mais certains problèmes peuvent se poser. En voici un. Essayons par exemple de faire tourner les lettres de 30 degres.

```
\relax
\metrics
\resetglyph{A}
  \glyphspecial{ps: gsave 30 rotate}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
\endsetglyph
\endmetrics
```

BC

On constate que la rotation ne s'effectue pas par rapport au point en bas à gauche du caractère, comme on s'y serait attendu. Regardons où se trouve `currentpoint`

```
\relax
\metrics
\resetglyph{A}
  \glyphspecial{ps: gsave currentpoint
    gsave newpath 5 0 360 arc closepath fill grestore}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
\endsetglyph
\endmetrics
```

ABC

Il est bien en bas à gauche du caractère. Regardons où se trouve le point de coordonnées (0,0).

```
\relax
\metrics
\resetglyph{A}
  \glyphspecial{ps: gsave 0 0 translate}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
\endsetglyph
\endmetrics
```

ABC

Il est en haut à gauche de la feuille. On peut donc à nouveau essayer la rotation.

```
\relax
\metrics
\resetglyph{A}
  \glyphspecial{ps: gsave
    currentpoint
    currentpoint translate
        30 rotate
    neg exch neg exch translate}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
\endsetglyph
\endmetrics
```

ABC

Voici maintenant une rotation par rapport au centre du caractère.

```
\relax
\metrics
\resetglyph{A}
  \movert{\scale{500}{\width{A}}}
  \moveup{\scale{500}{\height{A}}}
  \glyphspecial{ps: gsave
    currentpoint
      currentpoint translate
      30 rotate
    neg exch neg exch translate
  }
  \movert{\scale{-500}{\width{A}}}
  \moveup{\scale{-500}{\height{A}}}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
\endsetglyph
\endmetrics
```

ABC

On peut aussi s'amuser étirer les caractères.

```
\relax
\metrics
\resetglyph{A}
```

```
  \glyphspecial{ps: gsave
    currentpoint
    currentpoint translate
        2 1 scale
    neg exch neg exch translate}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
  \movert{\width{A}}
\endsetglyph
\endmetrics
```

ABC

On peut aussi s'amuser à les pencher (voir aussi 9.7).

```
\relax
\metrics
\resetglyph{A}
  \glyphspecial{ps: gsave
    currentpoint
    currentpoint translate
        [ 1 0 -.5 1  0 0 ] concat
    neg exch neg exch translate}
  \glyph{A}{1000}
  \glyphspecial{ps: grestore}
\endsetglyph
\endmetrics
```

*A*BC

```
\relax
\metrics
\resetglyph{A}
  \push
    \glyphspecial{ps: gsave
      .8 .8 .8 setrgbcolor
      currentpoint
      currentpoint translate
          [ 1 0 -.5 1  0 0 ] concat
      neg exch neg exch translate}
    \glyph{A}{1000}
    \glyphspecial{ps: grestore}
  \pop
  \glyph{A}{1000}
\endsetglyph
\endmetrics
```

ABC

9.16 Peut-on ajouter n'importe quelle instruction PostScript ?

Voici un autre exemple de problème : le corps d'une boucle PostScript est entouré d'accolades, or ces caractères sont interprétés particulièrement par LaTeX. Une manière de les utiliser consiste à changer le catcode des accolades.

```
\relax
\metrics
```

```
\begingroup
  \catcode`\{=11
  \catcode`\}=11
  \catcode`\(=1
  \catcode`\)=2
  \global\def\doit#1(%
  \ifisglyph(#1)\then
    \resetglyph(#1)
      \glyphspecial(ps: gsave 0 1 10 { )
      \glyphspecial(ps: 1 1 translate
                        10 div setgray
                    )
      \push \glyph(#1)(1000) \pop
      \glyphspecial(ps: } for grestore )
      \glyph(#1)(1000)
    \endresetglyph
  \fi)
  \catcode`\{=1
  \catcode`\}=2
  \catcode`\(=12
  \catcode`\)=12
\endgroup
\input caracteres
\endmetrics
```

Les caractères de cette fonte sont accompagnés d'une
« légère » ombre...

Une autre manière de se débarasser de toutes ces histoires de catcode est d'utiliser ActiveTeX.

9.17 Comment avoir des caractères gras quand il n'y en a pas ?

***** A FAIRE ***** Il suffit d'utiliser la question précédente.

9.18 Comment ajouter un caractère ?

***** A FAIRE *****

9.19 Comment ajouter une ligature ?

C'est là qu'interviennent les fichiers `*.etx`. Comme le codage `T1` est déjà bien remplis, nous allons partir du codage `OT1` dans lequel il y a beaucoup plus de place libre. Nous allons reprendre le fichier `ot1.etx` et modifier les choses suivantes (attention à l'endroit où on le met !). À l'endroit où la lettre `c` est définie :

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

Et à la fin du fichier :

```
\setslot{ct}
  \comment{added ligature `ct'}
\endsetslot
```

Le fichier `*.mtx` contient la définition du nouveau caractère.

```
\relax
\metrics
\setglyph{ct}
  \push \moveup{-150}\glyphrule{\add{150}{\width{c}}}{40} \pop
  \glyph{c}{1000}
  \push \moveup{-150}\glyphrule{\width{t}}{40} \pop
  \glyph{t}{1000}
\endsetglyph
  \setleftkerning{ct}{c}{1000}
  \setrightkerning{ct}{t}{1000}
\endmetrics
```

On appelle `fontinst` avec notre nouveau codage.

```
\input fontinst.sty
\installfonts
\installfont{toto16}
            {cmr10,16}
            {OT1_16}{OT1}{toto}{m}{n}{10}
\endinstallfonts
\bye
```

Action

9.20 Comment avoir des ligatures ct et st ?

Il faut tout d'abord trouver une fonte qui contienne ces glyphes : par exemple, les fontes d'Omega. Pour les utiliser, il suffit de préciser, dans le codage, qu'il y a des ligatures `ct` et `st`. On crée un fichier `ot1_ctst.etx` à partir de `ot1.etx` en rajoutant les lignes suivantes à la fin.

```
\setslot{ct}
  \comment{added ligature ‘ct’}
\endsetslot
\setslot{ct}
  \comment{added ligature ‘st’}
\endsetslot
```

On modifie aussi les entrées correspondant aux lettres `c` et `s`.

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

Le programme `fontinst` permettant de créer les nouvelles fontes est le suivant.

```
\input fontinst.sty
\installfonts
\installfont{times_ctst}
            {omsela}
            {OT1_ctst}{OT1}{toto}{m}{n}{10}
\installfont{times_ctst_i}
            {omselai}
```

```
            {OT1_ctst}{OT1}{toto}{m}{it}{10}
\installfont{times_ctst_b}
            {omselab}
            {OT1_ctst}{OT1}{toto}{bx}{n}{10}
\installfont{times_ctst_bi}
            {omselabi}
            {OT1_ctst}{OT1}{toto}{bx}{it}{10}
\endinstallfonts
\bye
```

Et on peut l'utiliser comme ceci.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}

\DeclareFontFamily{OT1}{toto}{}
\DeclareFontShape{OT1}{toto}{m}{n}{   <-> times_ctst }{}
\DeclareFontShape{OT1}{toto}{m}{it}{  <-> times_ctst_i }{}
\DeclareFontShape{OT1}{toto}{bx}{n}{  <-> times_ctst_b }{}
\DeclareFontShape{OT1}{toto}{bx}{it}{ <-> times_ctst_bi }{}

\renewcommand{\encodingdefault}{OT1}
\renewcommand{\rmdefault}{toto}

\begin{document}
Action Astérisque
\textit{Action Astérisque}
\textbf{Action Astérisque}
\textbf{\itshape Action Astérisque}
\end{document}
```

Action Asterisque
Action Asterisque
Action Asterisque
Action Asterisque

9.21 Comment changer l'espace entre deux caractères ?

On peut changer la table de kerning. Ainsi, l'exemple suivant va espace un peu moins le i et le j quand ils se suivent, sans rien changer sinon.

```
\relax
\metrics
  \setkern{I}{J}{-300}
\endmetrics
```

Une autre manière de procéder consiste à créer une ligature, comme nous l'avons vu précédemment.

9.22 Comment appliquer une même transformation à tous les caractères d'une fonte ? À toutes les majuscules ? À toutes les minuscules ?

Il suffit d'écrire une macro qui prenne en argument un nom de caractère et qui effectue la modification sur ce caractère. L'exemple suivant souligne toutes les majuscules

```
\relax
\metrics
\def\doit#1{%
  \ifisglyph{#1}\then
    \message{Processing character `#1'^^J}
    \resetglyph{#1}
    \push
      \moveup{-150}
      \glyphrule{\width{#1}}{40}
    \pop
    \glyph{#1}{1000}
    \endresetglyph
  \fi}
\input majuscules
\endmetrics
```

Il suffit ensuite d'appeler cette macro avec la liste de tous les caractères (ou de toutes les majuscules, ou de toutes les minuscules). Mais comment avoir ces listes ? On peut prendre la liste des caractères qui est dans les fichiers définissant le codage, par exemple `t1.etx`. La commande suivante permet d'obtenir la liste de tous les caractères non alphanulmériques.

```
grep '^\\setslot' `locate /t1.etx | head -1` |
  grep -v '\\sc|\\lc|\\uc|\\digit' |
  sed 's/^.setslot.//; s/.$//' |
  perl -p -e 's/^.currency/dollar\nsterling/' |
  perl -p -e 's/^/\\doit\{/; s/$/\}/;' |
  sort | uniq >symboles.tex
```

La commande suivante permet d'obtenir les chiffres.

```
grep '^\\setslot' `locate /t1.etx | head -1` |
  grep '\\digit' |
  sed 's/^.setslot.//; s/.$//' |
  perl -p -e 's/^.currency/dollar\nsterling/' |
  perl -p -e 's/^\\digit/\\doit/' |
  sort | uniq >chiffres.tex
```

La commande suivante permet d'obtenir les lettres minuscules.

```
grep '^\\setslot' `locate /t1.etx | head -1` |
  grep '\\lc|\\uc|\\sc' |
  sed 's/^.setslot.//; s/.$//' |
  perl -p -e 's/^.currency/dollar\nsterling/' |
  perl -p -e 's/^\\(lc|uc|sc|uclig|lclig|lctop|uctop)\{.*?\}/\\doit/' |
  sort | uniq >minuscules.tex
```

La commande suivante permet d'avoir la liste des majuscules.

```
grep '^\\setslot' `locate /t1.etx | head -1` |
  grep '\\lc|\\uc|\\sc' |
  sed 's/^.setslot.//; s/.$//' |
  perl -p -e 's/^.currency/dollar\nsterling/' |
  perl -p -e 's/^\\(lc|uc|sc|uclig|lclig|lctop|uctop)(\{.*?\}).*/\\doit$2/' |
  sort | uniq >majuscules.tex
```

Et voici la liste de tous les caractères.

```
for i in symboles chiffres minuscules majuscules
do
  echo '\input' $i
```

```
done > caracteres.tex
```

Nous pouvons maintenant reprendre les exemples précédents et les appliquer à toutes les lettres ou toutes les majuscules.

Dans Cette Fonte, Toutes Les Majuscules Sont Soulignées...

Maintenant, tous les caractères alphanumériques sont soulignés — mais pas les symboles de ponctuation...

Dans Cette Fonte, Toutes Les Majuscules Sont Encadrées...

Les caractères ont maintenant des couleurs aléatoires.

Les caractères sont légèrement décalés, d'une quantité aléatoire, un peu comme s'ils étaient tapés sur une vieille machine à écrire abîmée...

Les caractères sont légèrement décalés et tournés, d'une quantité aléatoire, un peu comme s'ils étaient tapés sur une vieille machine à écrire abîmée...

Les caractères sont légèrement décalés et tournés, d'une quantité aléatoire, un peu comme s'ils étaient tapés sur une vieille machine à écrire abîmée...

Dans Cette Fonte, Chaque Majuscule Est Légèrement Ombragée...

Dans cette fonte, chaque caractère est légèrement ombragé.

9.23 Comment avoir les caractères manquants ?

Lorsque l'on utilise une fontes TTF (ou autre), il peut manquer certains caractères : par exemple des lettres accentuées (le Z majuscule avec un point au dessus est rarement là — mais je ne sais pas à quoi il peut servir).

Le fichier `latin.mtx` s'occuppe de créer les caractères manquants quand c'est possible. C'est pourquoi on l'utilisera de manière quasi-systématique. [S'il est absent des exemples qui figurent dans ce document, c'est parce que je ne les ai pas encore corrigés.]

Prenons par exemple la fonte True Type `comic` : il manque le Z majuscule accent aigu. On crée les métrique comme nous l'avons vu dans un chapitre précédent.

```
ttf2tfm comic.ttf -q -T T1-WGL4.enc -v comic.vpl REC_comic.tfm >> ttfonts.map
vptovf comic.vpl comic.vf comic.tfm
tftopl comic.tfm > comic.pl
perl -p -i -e 's/(EC|T1)ENCODING/EXTENDED TEX FONT ENCODING - LATIN/' *.pl
</p>
<p>
  On peut maintenant lancer <tt>fontinst</tt>.
  <verb>
\input fontinst.sty
\installfonts
\declareencoding{ECENCODING}{T1}
\declareencoding{T1ENCODING}{T1}
```

```
\installfont{fontinst_comic}
            {comic,latin}
            {T1}{T1}{toto}{m}{n}{10}
\endinstallfonts
\bye
```

Il n'est pas très beau, mais on a maintenant un caractère Z accent aigu.

- — « » œŒŹŻ

9.24 Comment avoir une fonte soulignée ?

Il existe déjà un fichier uline.mtx, qui contient les lignes suivantes (voir 9.13 et 9.22)

```
\setcommand\uline#1{%   underlining
  \ifisglyph{#1}\then
    \resetglyph{#1}
      \push\moveup{-150}\glyphrule{\width{#1}}{40}\pop
      \glyph{#1}{1000}
    \endresetglyph
  \fi
}
```

Mais on peut vouloir que la barre de soulignement se poursuive aussi entre les mots. On doit pouvoir le faire en rendant le caractère espace actif et en le faisant agir comme `\leaders...`. Attention à ne pas inhiber la césure... ***** EXERCICE *****

On peut aussi vouloir que la barre de soulignement passe derrière les caractères : quand un caractère descend en dessous de la ligne de base, la barre de soulignement ne devrait pas couper ce caractère mais s'arrêter un peu avant la partie qui descent et reprendre un peu après. Une solution, pas entièrement satisfaisante, consiste à tracer la barre de soulignement ; puis le caractère, en blanc, légèrement décalé sur la droite et sur la gauche, ce qui efface certaines parties de la ligne ; et, enfin, le caractère avec la couleur normale. Ne pas utiliser sur un fond coloré...

```
\relax
\metrics
\def\doit#1{%
  \message{Processing character `#1'^^J}
  \ifisglyph{#1}\then
    \resetglyph{#1}
    \push
      \moveup{-150}
      \glyphrule{\width{#1}}{40}
    \pop
    \glyphspecial{ps: gsave 255 255 255 setrgbcolor}
      \push \movert{-40} \glyph{#1}{1000} \pop
      \push \movert{40}  \glyph{#1}{1000} \pop
    \glyphspecial{ps: grestore}
    \glyph{#1}{1000}
    \endresetglyph
  \fi}
\input chiffres
\input majuscules
\input minuscules
\endmetrics
```

Maintenant, tous les caractères alphanumériques sont soulignés — mais pas les symboles de ponctuation...

9.25 Comment avoir un j sans point quand il n'y en a pas ?

Il y a un fichier `dotlessj.mtx` qui contient :

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% This is the file dotlessj.mtx, part of the VFinst package
%% (version 0.95, December, 1997) for PostScript Type1 font
%% installation.  (Author: Alan Hoenig, ajhjj@cunyvm.cuny.edu)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


\relax

This fontinst file sets a dotless j glyph for a TeX
math italic font.  The following code is due to Thierry Bouche,
a paper of whose on virtual math fonts can be found at
ftp://fourier.ujf-grenoble.fr/pub/contrib-tex/gut/mathfonts-pk.ps.gz

\metrics

\ifisglyph{j}\then
  \resetglyph{dotlessj}
  \glyphspecial{ps: gsave gsave}
   \push
     \moveup{\neg{\depth{j}}}
      \movert{\neg{\width{j}}}
    \glyphspecial{ps: currentpoint /JDepth exch def /JLeft exch def}
     \moveup{\add{\depth{j}}{\height{dotlessi}}}
     %\movert{\mul{\width{j}}{2}}
     \movert{\scale{\width{j}}{2500}}
     \glyphspecial{ps: currentpoint /IHeight exch def /JRight exch def grestore}
     \glyphspecial{ps:   JLeft JDepth moveto JLeft neg JRight add 0 rlineto
        0 JDepth neg IHeight add rlineto JLeft neg JRight add neg 0
        rlineto 0 JDepth neg IHeight add neg rlineto closepath  clip}
     \pop
     \glyph{j}{1000}
     \glyphspecial{ps: grestore}
           \resetdepth{\depth{j}}
           \resetwidth{\width{j}}
           \resetitalic{\italic{j}}
           \resetheight{\height{dotlessi}}
  \endresetglyph
\fi

\endmetrics
```

ȷjȷj

9.26 Peut-on manipuler des familles de fontes(plutôt que des fontes isolées) avec fontinst ?

Oui, par exemple à l'aide de la commande `\latinfamily`, mais il faut impérativement que le nom des fontes soit compatible à `fontname`, donc illisible. Je ne l'ai jamais utilisé.

9.27 Comment avoir une fonte grasse quand il n'y en a pas (sans fontes virtuelles) ?

On peut utiliser la commande `\pmb` d'`amsmath`. Elle prend un (ou plusieurs) caractères et les superpose à eux-même après un léger décalage. Le résultat est à peu près correct, mais LaTeX est incapable de couper les mots, les lignes ou les pages à ces endroits-là). Voici un exemple, en mode mathématique (on remarquera que l'on utilise des fontes texte en mode mathématique pour ne pas avoir de problème avec les lettres accentuées).

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{  <-> oldeng }{}
\DeclareFontShape{T1}{toto}{m}{sl}{ <-> oldeng_slanted }{}
\DeclareFontShape{T1}{toto}{bx}{n}{ <-> sub * toto/m/n}{}
\DeclareFontShape{T1}{toto}{bx}{sl}{<-> sub * toto/m/sl}{}

\DeclareSymbolFont{toto}{T1}{toto}{m}{n}
\DeclareSymbolFontAlphabet{\mathTOTO}{toto}
\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\newcommand{\cat}[1]{\text{\rm\TOTO #1}}
\newcommand{\dcat}[1]{\text{\rm\TOTO\pmb{#1}}}

\usepackage{amsmath}

\begin{document}

Les noms de catégorie seront écrits dans une fonte gothique (par
exemple $\cat{Ens}$ pour la catégorie des ensembles, $\cat{Sh}\,X$
pour la catégorie des faisceaux étales sur $X$) et les $2$-catégories
par une fonte gothique grasse (par exemple $\dcat{Cat}$ pour la
$2$-catégorie des catégories ou $\dcat{Groupoïdes}$ pour celle des
groupoïdes).

Nous n'utiliserons pas de fonte gothique italique
(\textTOTO{\textsl{Toto}}).

\end{document}
```

9.28 Comment avoir une fonte grasse quand il n'y en a pas (avec des fontes virtuelles, premier essai, infructueux — A EFFACER) ?

Nous présentons ici divers essais, infructueux, pour tenter de mettre des caractères en gras. Au cours de ces différents échecs, nous verrons comment utiliser les différentes commandes de fontinst. Le lecteur désirant voir une bonne solution tout de suite peut passer à la question suivante.

Le fichier toto.tex contient :

```
\input fontinst.sty
\installfonts
\installfont{toto}
            {ecrm1000,fauxgras}
            {T1}{T1}{fcm}{m}{n}{10}
\endinstallfonts
```

```
\bye
```

Le fichier fauxgras.mtx contient :

```
\relax
\metrics
  \setglyph{A}
    \glyph{A}{1000}
    \movert{10} \glyph{A}{1000}
    \movert{10} \glyph{A}{1000}
    \movert{10} \glyph{A}{1000}
    \movert{10} \glyph{A}{1000}
  \endsetglyph
\endmetrics
```

Ça ne marche pas car j'ai oublié de mettre \unsetglyph Essayons avec :

```
\relax
\metrics
  \unsetglyph{A}
  \setglyph{A}
    \glyph{A}{1000}
    \movert{10} \glyph{A}{1000}
    \movert{100} \glyph{A}{1000}
    \movert{100} \glyph{A}{1000}
    \movert{100} \glyph{A}{1000}
  \endsetglyph
\endmetrics
```

Ça ne marche toujours pas, car le A n'est plus défini. Essayons avec:

```
\relax
\metrics
  \setglyph{newA}
    \glyph{A}{1000}
  \endsetglyph
  \unsetglyph{A}
  \setglyph{A}
    \glyph{A}{1000}
    \movert{100} \glyph{newA}{1000}
    \movert{100} \glyph{newA}{1000}
    \movert{100} \glyph{newA}{1000}
    \movert{100} \glyph{newA}{1000}
  \endsetglyph
\endmetrics
```

Ça marche un peu mieux, mais la commande \glyph tient compte de la largeur du A : on se retrouve, non pas avec des A presque superposés, mais avec des A les uns à côté des autres. Essayons avec :

```
\relax
\metrics
  \setglyph{newA}
    \glyph{A}{1000}
  \endsetglyph
  \unsetglyph{A}
  \setglyph{A}
    \glyph{A}{1000}
    \push \movert{100} \glyph{newA}{1000} \pop
    \push \movert{200} \glyph{newA}{1000} \pop
```

```
    \movert{300} \glyph{newA}{1000}
  \endsetglyph
\endmetrics
```

C'est à peu près ça. Il reste à affiner un peu les distances. On peut aussi utiliser la commande \resetglyph à la place de

```
  \setglyph{new...}
    ...
  \endsetglyph
  \unsetglyph{...}
  \setglyph{...}
    ...
  \endsetglyph
```

Essayons enfin avec :

```
\relax
\metrics
  \resetglyph{A}
    \push \glyph{A}{1000} \pop
    \push \movert{10} \glyph{A}{1000} \pop
    \push \movert{20} \glyph{A}{1000} \pop
    \push \movert{30} \glyph{A}{1000} \pop
    \push \movert{40} \glyph{A}{1000} \pop
    \movert{50} \glyph{A}{1000}
  \endsetglyph
\endmetrics
```

Ça marche comme on s'y attend.

On compile le tout en tapant :

```
  tftopl ecrm1000 > ecrm1000.pl
  tex essai.tex
```

On l'essaye comme d'habitude :

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{toto}{}
\DeclareFontShape{T1}{toto}{m}{n}{ <-> toto }{}

\renewcommand{\encodingdefault}{T1}
\renewcommand{\rmdefault}{toto}

\newcommand{\textTOTO}[1]
{{\fontencoding{T1}\fontfamily{toto}\selectfont #1}}
\newcommand{\TOTO}{\fontencoding{T1}\fontfamily{toto}\selectfont}

\begin{document}

Voici la nouvelle fonte.

ABCDEFGHIJKLMNOPQRSTUVWXYZ.
abcdefghijklmnopqrstuvwxyz.

\end{document}
```

Compliquons les choses : essayons d'obtenir tous les caractères en gras.

Dans le fichier essai.tex, on commence par définir une macro qui va créer une fausse lettre grasse

```
\input fontinst.sty
\installfonts
\installfont{toto}
            {ecrm1000,fauxgras}
            {T1}{T1}{fcm}{m}{n}{10}
\endinstallfonts
\bye
```

On peut comprend d'abord la définition d'une macro qui crée une fausse lettre grasse,

```
\def\fakebold#1{%
  \ifisglyph{#1}\then
    \resetglyph{#1}
      \push \glyph{#1}{1000} \pop
      \push \movert{10} \glyph{#1}{1000} \pop
      \push \movert{20} \glyph{#1}{1000} \pop
      \push \movert{30} \glyph{#1}{1000} \pop
      \push \movert{40} \glyph{#1}{1000} \pop
      \movert{50} \glyph{#1}{1000}
    \endresetglyph
  \fi
}
```

Puis on le complète à l'aide d'une boucle. Pour cela, il nous faut une liste de noms de glyphes, que l'on obtient à partir des fichiers `t1.etx` et `ot1.etx`. (Détailler.)

```
echo '\\relax\\metrics' > fauxgras.mtx

for i in AE Aacute Abreve Acircumflex Adieresis Agrave Aogonek Aring Atilde A B Cacute Ccaron Ccedilla C Dcaron
do
  echo '\\fakebold{'$i'}'
done >> fauxgras.mtx

echo '\\endmetrics' >> fauxgras.mtx
```

9.29 Comment avoir une fonte grasse quand il n'y en a pas (avec des fontes virtuelles — A EFFACER) ?

Les essais précédents étaient presque acceptables, mais on ne déplaçait mes caractères qu'horizontalement : les traits verticaux étaient bien épaissis, les traits obliques moins et les traits horizontaux pas du tout. Nous allons donc déplacer les caractères en cercle, pour que l'apaississement soit plus uniforme.

Je ne trouve pas la solution que je présente entièrement satisfaisante : pour savoir de combien il faut déplacer le caractère, j'utilise un petit script en Perl. On devrait pouvoir, soit faire ces calculs en TeX (si, si), soit en PostScript (beaucoup plus raisonable).

```
#!perl -w

$n = 10; # Nombre de copies du caractère.
$r = 20; # Rayon du cercle
$PI = 3.14159;

print '\input fontinst.sty' ."\n";
print '\def\fakebold#1{%' ."\n";
print '  \ifisglyph{#1}\then' ."\n";
```

```
print '    \resetglyph{#1}' ."\n";

for($i=0; $i < 2*$PI; $i += 2*$PI/$n){
  print '      \push\movert{'.
        sprintf("%.0f",$r*cos($i))
.'}\moveup{'.
sprintf("%.0f",$r*sin($i))
.'}\glyph{#1}{1000}\pop' . "\n";
}

print '      \glyph{#1}{1000}' ."\n";
print '    \endresetglyph' ."\n";
print '  \fi' ."\n";
print '} ' ."\n";
print '\installfonts' ."\n";
print '\installfont{toto}' ."\n";
print '            {ecrm1000,fauxgras}' ."\n";
print '            {T1}{T1}{fcm}{m}{n}{10}' ."\n";
print '\endinstallfonts' ."\n";
print '\bye' ."\n";
```

On obtient quelque chose qui ressemble à

```
\def\fakebold#1{%
  \ifisglyph{#1}\then
    \resetglyph{#1}
      \push\movert{20}\moveup{0}\glyph{#1}{1000}\pop
      \push\movert{16}\moveup{12}\glyph{#1}{1000}\pop
      \push\movert{6}\moveup{19}\glyph{#1}{1000}\pop
      \push\movert{-6}\moveup{19}\glyph{#1}{1000}\pop
      \push\movert{-16}\moveup{12}\glyph{#1}{1000}\pop
      \push\movert{-20}\moveup{0}\glyph{#1}{1000}\pop
      \push\movert{-16}\moveup{-12}\glyph{#1}{1000}\pop
      \push\movert{-6}\moveup{-19}\glyph{#1}{1000}\pop
      \push\movert{6}\moveup{-19}\glyph{#1}{1000}\pop
      \push\movert{16}\moveup{-12}\glyph{#1}{1000}\pop
      \glyph{#1}{1000}
    \endresetglyph
  \fi
}
```

9.30 Comment avoir une fonte « petites capitales » quand les caractères sont là ?

Pour les petites capitales : il semblerait que vfinst suppose qu'elles existent dans la fonte, avec des glyphes ayant pour noms Asmall, Bsmall, etc.

Même chose pour les vieux chiffres, qui ont pour noms oneoldstyle, twooldstyle, etc.

```
...
```

9.31 Comment avoir une fonte « petites capitales » quand les caractères ne sont pas là ?

***** A REVOIR *****

**** Que fait le fichier `t1c.etx` ???

L'idée est de prendre une majuscule mais à une taille différente. On a donc besoin des *couples* majuscule/minuscule.

```
grep '^\\setslot' `locate /t1.etx | head -1` |
  grep '\\lc|\\uc|\\sc' |
  sed 's/^.setslot.//; s/.$//' |
  perl -p -e 's/^.currency/dollar\nsterling/' |
  perl -p -e 's/^\\(lc|uc|sc|uclig|lclig|lctop|uctop)\{/\\doittwo\{/' |
  sort | uniq > couples.tex
```

```
\relax
\metrics
\def\doittwo#1#2{%
\ifisglyph{#1}\then
  \resetglyph{#2}
    \glyph{#1}{800}
  \endresetglyph
\fi}
\input couples
\endmetrics
```

De Fausses Petites Capitales...

On peut rafiner un peu et prendre comme hauteur des petites capitales, la hauteur des minuscules. La hauteur des minuscules est dans la variable `xheight` et la hauteur des majuscules dans `capheight`. On veut le quotient des deux.

```
\relax
\metrics
\def\doittwo#1#2{%
\ifisglyph{#1}\then
  \resetglyph{#2}
    \setint{tmpoa}{\mul{1000}{\int{xheight}}}
    \setint{tmpob}{\div{\int{tmpoa}}{\height{H}}}
    \glyph{#1}{\int{tmpob}}
  \endresetglyph
\fi}
\input couples
\endmetrics
```

De Fausses... De Fausses Petites Capitales...

Nos fausses petites capitales ont les défauts habituels des fausses petites capitales : elles sont trop maigres et trop rapprochées. On peut tenter de les épaissir en reproduisant plusieures fois le même caractère, avec de légers décalages.

```
\relax
\metrics
\setint{tmpoa}{\mul{1000}{\int{xheight}}}
\setint{taille}{\div{\int{tmpoa}}{\height{H}}}
\setint{epaisseur}{8}
\def\doittwo#1#2{%
\ifisglyph{#1}\then
  \resetglyph{#2}
    \push \movert{\int{epaisseur}}\glyph{#1}{\int{taille}} \pop
    \push \moveup{\int{epaisseur}}\glyph{#1}{\int{taille}} \pop
    \push \movert{\neg{\int{epaisseur}}}\glyph{#1}{\int{taille}} \pop
    \glyph{#1}{\int{taille}}
  \endresetglyph
\fi}
```

```
\input couples
\endmetrics
```

De Fausses... De Fausses Petites Capitales...

Il n'est pas possible de connaître a priori le décalage qu'il va falloir prendre : pour cela, il faudrait connaître l'épaisseur des traits, qui n'est pas dans les métriques (cette épaisseur est d'ailleurs variable). Pour trouver la « bonne » valeur, nous allons tatonner.

1 AaBbCcDdEeFf...XxYyZz

2 AaBbCcDdEeFf...XxYyZz

3 AaBbCcDdEeFf...XxYyZz

4 AaBbCcDdEeFf...XxYyZz

5 AaBbCcDdEeFf...XxYyZz

6 AaBbCcDdEeFf...XxYyZz

7 AaBbCcDdEeFf...XxYyZz

8 AaBbCcDdEeFf...XxYyZz

9 AaBbCcDdEeFf...XxYyZz

10 AaBbCcDdEeFf...XxYyZz

11 AaBbCcDdEeFf...XxYyZz

12 AaBbCcDdEeFf...XxYyZz

13 AaBbCcDdEeFf...XxYyZz

14 AaBbCcDdEeFf...XxYyZz

15 AaBbCcDdEeFf...XxYyZz

16 AaBbCcDdEeFf...XxYyZz

17 AaBbCcDdEeFf...XxYyZz

18 AaBbCcDdEeFf...XxYyZz

19 AaBbCcDdEeFf...XxYyZz

20 AaBbCcDdEeFf...XxYyZz

On peut améliorer le bricolage précédent en utilisant un nombre plus grand de copies du caractère.

```
\relax
\metrics
\setint{tmpoa}{\mul{1000}{\int{xheight}}}
\setint{taille}{\div{\int{tmpoa}}{\height{H}}}
\begingroup
  \catcode`\{=11
  \catcode`\}=11
  \catcode`\(=1
  \catcode`\)=2
  \global\def\doittwo#1#2(%
  \ifisglyph(#1)\then
    \resetglyph(#2)
      \glyphspecial(ps: gsave 0 30 360 { )
      \glyphspecial(ps: dup sin .8 mul exch cos .8 mul translate )
      \push \glyph(#1)(\int(taille)) \pop
      \glyphspecial(ps: } for grestore )
      \glyph(#1)(\int(taille))
    \endresetglyph
  \fi)
  \catcode`\{=1
  \catcode`\}=2
  \catcode`\(=12
  \catcode`\)=12
\endgroup
\input couples
\endmetrics
```

De Fausses... **De Fausses Petites Capitales...**

9.32 Comment utiliser `vfinst` ?

`vfinst` permet d'installer une famille de fontes PostScript. Supposons que nous voulions installer la famille `AmericanGaramondBT`, ie, les fichiers `bg3b.afm`, `bg3b.pfb`, `bg3bi.afm`, `bg3bi.pfb`, `bg3i.afm`, `bg3i.pfb`, `bg3r.afm`, `bg3r.pfb`,

On commence par installer `vfinst` et mettre les fontes que l'on veut installer dans un répertoire temporaire.

```
tar zxvf vfinst.tar.gz
cd vfinst
mkdir work
cd work
cp ../../*.afm ../../*.pfb .
```

Il faut ensuite mettre les bon paramètres dans le fichier `../vfinst.par`. Je suppose que les fontes seront installées dans le (nouveau) répertoire `$HOME/gnu/Linux/lib/texmf_tmp`.

```
chmod u+w ../vfinst.par
echo '$vfencoding = "'`locate 8r.enc | head -1`'";' >> ../vfinst.par
echo '$vfenc = "OT1";' >> ../vfinst.par
echo '$vftexmf  = "'$HOME/gnu/Linux/lib/texmf_tmp'";' >> ../vfinst.par
mkdir -p $HOME/gnu/Linux/lib/texmf_tmp/dvips/base/
cp `locate psfonts.map | head -1 ` $HOME/gnu/Linux/lib/texmf_tmp/dvips/base/psfonts.map
```

```
perl ../vfinst
perl ../1vfinst
```

```
less fonts.lst
perl ../2vfinst
```

Pour terminer, on fait ce qu'il demande.

```
mkdirs.bat
newnames.bat
cp ../dotlessj.mtx .
cp ../latinx.mtx
cp ../uline.mtx .
tex makefont
maketfm.bat
putfonts.bat
putvftf.bat
texhash
cleanup.bat
```

Les fontes sont installées (il y a toutefois un léger problème : le fichier `psfonts.map` qu'il a modifié n'est pas le bon — mais il suffit de ne pas effacer celui du répertoire courrant) : on peut maintenant essayer de les utiliser.

```
latex testltx.tex
```

A LaTeX Test File
Prepared by vfinst1.00 at 18:19:11, on Wednes, 28 Jun 100.

The text in the left margin separated by slashes yields the NFSS combination of encoding, family, series, and shape that generated that test line.

`OT1/bg3/m/n`	A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?
`OT1/bg3/m/sl`	*A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?*
`OT1/bg3/m/sc`	A FINE FLIGHT OFFICIOUSLY RUFFLES MY PUFF! 12345 ,.;: ÇE PHŒNIX VA À MÜNCHEN?
`OT1/bg3/m/un`	A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?
`OT1/bg3/m/it`	*A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?*
`OT1/bg3/m/ui`	A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?
`OT1/bg3/m/si`	*A FINE FLIGHT OFFICIOUSLY RUFFLES MY PUFF! 12345 ,.;: ÇE PHŒNIX VA À MÜNCHEN?*
`OT1/bg3/m/ni`	*A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?*
`OT1/bg3/b/n`	**A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?**
`OT1/bg3/b/sl`	***A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?***
`OT1/bg3/b/sc`	**A FINE FLIGHT OFFICIOUSLY RUFFLES MY PUFF! 12345 ,.;: ÇE PHŒNIX VA À MÜNCHEN?**
`OT1/bg3/b/un`	**A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?**
`OT1/bg3/b/it`	***A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?***
`OT1/bg3/b/ui`	**A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?**
`OT1/bg3/b/si`	***A FINE FLIGHT OFFICIOUSLY RUFFLES MY PUFF! 12345 ,.;: ÇE PHŒNIX VA À MÜNCHEN?***
`OT1/bg3/b/ni`	***A fine flight Officiously Ruffles My Puff! 12345 ,.;: Çe Phœnix Va à München?***

9.33 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (1) ?

Nous avons déjà vu une solution partielle (4.14) : prendre les caractères textuels dans la fonte texte et les autres dans la fonte mathématique Computer Modern ou Euler.

9.34 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (2) ?

On peut le faire avec des fontes virtuelles. On a besoin de trois fontes mathématiques, de codages respectifs `OMS`, `OMX` et `OML` et d'une fonte texte au codage `OT1`. Nous allons partir d'une fonte texte `toto` et des trois fontes mathématiques par défaut `cmmi10 (OML)` `cmmib10 (OML)` `cmsy10 (OMS)` `cmbsy10 (OMS)` `cmex10 (OMX)` (à mettre avec sfixed*cmex10). Les lettres grecques majuscules sont attendues dans la fonte texte (`OT1`) : nous irons les chercher dans `cmr10`. Voici le programme `fontinst`.

```
\input fontinst.sty
\installfonts
\installfont{fontinst_comic_ot1}
            {comic,latin,dotlessj,removegreek,cmr10}
            {OT1}{OT1}{comic}{m}{n}{10}
\installfont{fontinst_comic_t1}
            {comic,latin,dotlessj,removegreek,cmr10}
            {T1}{T1}{comic}{m}{n}{10}
\installfont{comic_oml}
            {comic,latin,dotlessj,removegreek,cmmi10}
            {OML}{OML}{comic}{m}{n}{10}
\installfont{comic_bold_oml}
            {comic,latin,dotlessj,removegreek,cmmib10}
            {OML}{OML}{comic}{bx}{n}{10}
\installfont{comic_oms}
            {comic,latin,dotlessj,removegreek,cmsy10}
            {OMS}{OMS}{comic}{m}{n}{10}
\installfont{comic_bold_oms}
            {comic,latin,dotlessj,removegreek,cmbsy10}
            {OMS}{OMS}{comic}{bx}{n}{10}
\installfont{comic_omx}
            {comic,latin,dotlessj,removegreek,cmex10}
            {OMX}{OMX}{comic}{m}{n}{10}
\endinstallfonts
\bye
```

Il faut faire attention à `latin.mtx` : il va remplacer les glyphes que l'on ne peut pas créer par un message d'erreur. Pour éviter que cela se produise, on va simplement effacer les caractères grecs que l'on ira chercher dans `cmr10`. Voici le fichier `removegreek.mtx`.

```
\relax
\metrics
\unsetglyph{Gamma}
\unsetglyph{Delta}
\unsetglyph{Theta}
\unsetglyph{Lambda}
\unsetglyph{Xi}
\unsetglyph{Pi}
\unsetglyph{Sigma}
\unsetglyph{Upsilon}
\unsetglyph{Phi}
\unsetglyph{Psi}
\unsetglyph{Omega}
\endmetrics
```

On peut l'utiliser de la manière suivante.

```
\documentclass{demo}

\DeclareFontFamily{OT1}{comic}{}
\DeclareFontShape{OT1}{comic}{m}{n}{ <-> fontinst_comic_ot1 }{}
```

```
%\DeclareFontShape{OT1}{comic}{bx}{n}{ <-> fontinst_bold_comic }{}

\renewcommand{\encodingdefault}{OT1}
\renewcommand{\rmdefault}{comic}

\newcommand{\textcomic}[1]
{{\fontencoding{OT1}\fontfamily{comic}\selectfont #1}}
\newcommand{\comic}{\fontencoding{OT1}\fontfamily{comic}\selectfont}

\DeclareFontFamily{OML}{comic_oml}{}
\DeclareFontShape{OML}{comic_oml}{m}{it}{ <-> comic_oml }{}
\DeclareFontFamily{OMX}{comic_omx}{}
\DeclareFontShape{OMX}{comic_omx}{m}{n}{ <-> comic_omx }{}
\DeclareFontFamily{OMS}{comic_oms}{}
\DeclareFontShape{OMS}{comic_oms}{m}{n}{ <-> comic_oms }{}

\DeclareSymbolFont{operators}   {OT1}{comic} {m}{n}
\DeclareSymbolFont{letters}     {OML}{comic_oml} {m}{it}
\DeclareSymbolFont{symbols}     {OMS}{comic_oms}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{comic_omx}{m}{n}

\begin{document}

ABC $ 0 = 1 ABC \sin x \int \otimes$

\end{document}
```

```
#!/bin/sh

ttf2tfm comic.ttf -q -T T1-WGL4.enc -v comic.vpl REC_comic.tfm >> ttfonts.map
vptovf comic.vpl comic.vf comic.tfm

cp `locate /cmmi10.tfm` .
cp `locate /cmmib10.tfm` .
cp `locate /cmsy10.tfm` .
cp `locate /cmbsy10.tfm` .
cp `locate /cmex10.tfm` .

for i in *.tfm
do
  j=`echo $i | sed 's/\.tfm$//'`
  tftopl $j.tfm > $j.pl
done

perl -p -i -e 's/(EC|T1)ENCODING/EXTENDED TEX FONT ENCODING - LATIN/' *.pl

tex 1.fontinst.tex
for i in *.vpl
do
  j=`echo $i | sed 's/\..*//'`
  vptovf $j
done

latex 1
dvips -f -E 1.dvi > 1.ps

exit
```

```
rm -f  *.log *.vf *.vpl *.pl *\~ *.ps *.dvi *.aux toto*
```

9.35 Comment avoir des majuscules mathématiques droites avec des fontes virtuelles ?

***** A FAIRE *****

9.36 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (3) ?

On peut faire la même chose, mais avec d'autres fontes mathématiques : Euler, Lucida, MathTimes, Mathematica ; c'est ce que fait `mathinst`. Il suppose que les fontes sont installées en respectant `fontname` (génant, mais normal), que tous les fichiers sont dans la même arborescence (car le script n'utilise pas `kpsewhich`) et qu'on a le droit d'écrire dedans. Pour cette raison, je ne l'utiliserai pas. (Il se plaint aussi de l'absence de certains répertoires.)

```
tar zxvf mathinst.tar.gz
cd mathinst
mkdir work
chmod u+w *
echo >> eu.par
echo '$texroot_       = "'$HOME/gnu/Linux/lib/texmf_tmp'";' >> eu.par
echo '$psfonts_       = "$texroot_/fonts/type1";' >> eu.par
echo '$afm_           = $psfonts_ . "/afm";' >> eu.par
echo '$vf_            = $texroot_ . "/fonts/vf";' >> eu.par
echo '$tfm_           = $texroot_ . "/fonts/tfm";' >> eu.par
echo '$inputs_        = "$texroot_/inputs";' >> eu.par
echo '$map_           = "$texroot_/dvips/base"; ' >> eu.par
echo '$texmf_ = "'$HOME/gnu/Linux/lib/texmf_tmp'";' >> eu.par
echo '$encoding_ = "ot1";' >> eu.par
cd work

cp `kpsewhich psfonts.map` psfonts.map
cat `locate pdftex.map | head -1` >> psfonts.map
cat $HOME/gnu/Linux/lib/texmf_tmp/dvips/base/psfonts.map >> psfonts.map
cp psfonts.map $HOME/gnu/Linux/lib/texmf_tmp/dvips/base

mkdir -p $HOME/gnu/Linux/lib/texmf_tmp/fonts/type1/ams/euler/
cp `locate fonts/type1/bluesky/euler/` $HOME/gnu/Linux/lib/texmf_tmp/fonts/type1/ams/euler/
mkdir -p $HOME/gnu/Linux/lib/texmf_tmp/fonts/tfm/ams/euler/
cp `locate fonts/tfm/ams/euler/` $HOME/gnu/Linux/lib/texmf_tmp/fonts/tfm/ams/euler/

perl ../mathinst eu bg3
mkdirs.bat
makepl.bat
tex makemath.tex
makevf.bat
latex testmath.tex
Make the adjustments as outlined in
     `mathinst.tex';
putfonts.bat
texhash
```

9.37 Comment créer une fonte mathématique qui se marrie bien avec une fonte texte donnée (3) ?

Une manière beaucoup plus satisfaisante de procéder consiste à créer une nouvelle fonte mathématique, en prenant les caractères de la fonte texte là où c'est possible et en prenant des caractères provenant de Computer Modern pour les autres, mais avec des paramètres choisis afin qu'elle se marrie bien avec la fonte texte. C'est ce que fait `mathkit`. Ce script présente les mêmes limitations que `mathinst` : il se refuse à utiliser `kpsewhich`, si bien que l'utilisateur doit lui dire exactement où se trouve chaque fichier, il suppose que tous les fichiers sont installés dans une arborescence unique et suppose que l'utilisateur a le droit d'y écrire. Je ne l'utiliserai donc pas.

9.38 Divers (réponses sans questions et questions sans réponses à classer)

```
Est-il vrai que les espaces inter-lettres sont légèrement extensibles ?
(Dans l'exemple précédent, regarder entre les lettre p et q).

Si c'est le cas (À VÉRIFIER), comment faire pour que le soulignement
paraisse continu ?

Il suffit de modifier la commande de soulignement pour que le trait se poursuive
un peu au delà du caractère.

\def\uline#1{%   underlining
  \ifisglyph{#1}\then
    \resetglyph{#1}
      \push
        \moveup{-150}
        \glyphrule{40}{40}
        \glyphrule{\width{#1}}{40}
        \glyphrule{40}{40}
      \pop
      \glyph{#1}{1000}
    \endresetglyph
  \fi
}





Les espaces inter-mots sont eux aussi extensibles, et bien plus.
Y a-t-il quand-meme une solution pour que les espaces soient eux-aussi soulignés ?



Entourer chaque caractère d'un cadre.



Remplacer chaque caractère par un cadre à ses dimensions.


Parler de la commande << latin >> (si j'ai bien compris, elle vérifie si
tous les caractères dont on a besoin sont là et, si ce n'est pas le
cas, elle essaye de fabriquer ceux qui manquent).

Quels sont les fichiers *.mtx qui sont déjà là ?
```

```
Qu'est-ce que vfinst ?

Dans la fonte gara.ttf, je constate qu'il y a les ligatures fi et fl :
LaTeX les utilise-t-il ?
Que se passe-t-il au niveau des accents : je constate qu'il y a plein
d'accents, LaTeX est-il capable de les utiliser ?
Est-ce le fichier latin.mtx qui s'occuppe de cela ?



Il reste peut-être une chose :
expliquer comment mélanger des commandes PostScript dans une fonte.
Par exemple : le j sans le point.
Autre exemple : divers effets graphiques (que l'on peut faire avec
tous types de fontes, sans géner l'algorithme de césure de \LaTeX...)


----------------------------------------------------------------------

I want to construct poor man's oldstyle figures by doing the following:

3,4,5,7,9,      lower by (capheight-xheight)
6,8,            do nothing
1,2,0           scale *height* by xheight/capheight

this last one poses a problem to me:

[...]

If you want to scale only on the y axis, you need to make a font
scaled on the x axis and use it in a VF 'scaled' something.

namely : create pppr8rx with xscale capheight/xheight (define it in
psfonts.map with ExtendFont), then use the 1 from that font scaled
xheight/capheight in your OsVF.

that way, you can also do better small caps for sans serif fonts.

[...]

Right, the closest you can come to approximating smallcaps is to scale
differently (less) horizontally than vertically and to not go down to
x-height, but some fraction of cap-height (the "best" value of
which depends on the particular font).  It will, of course, not look nearly
right unless you also adjust the stem widths and the side-bearings.

----------------------------------------------------------------------
```

Chapitre 10

À FAIRE

- Essayer d'obtenir des versions texte et postscript de ce document.
- Essayer de trouver des fontes PostScript gratuites.
- NFSS : donner un exemple d'utilisation de plusieures versions mathématiques.
- Parler de Babel pour écrire dans d'autres langues. Essayer de regarder dans certains cas particuliers (grec, russe).
- Mettre des images illustrant des fontes non latines.
- Catalogue de fontes MF : revoir le script.
- MF : revoir où mettre les fichiers *.mf *.tfm
- Utilisation d'une fonte texte en mode mathématique : donner un exemple.
- Modification des fontes mathématiques : images.
- Catalogue de fontes PS : après avoir troivé de telles fontes.
- Corriger les « ECEncoding ReEncodeFont » : ce devrait être T1 à la place de EC. (Vérifier que cela marche.)
- Enlever mon nom du fichier Template.tex
- Mettre des liens sur les répertoires contenant les exemples.
- Comment pencher une fonte PS (comme pour les fontes TTF) (D'ailleurs, en PostScript, on doit pouvoir faire d'autres choses, plus compliquées, avec les fontes. Essayer de donner des exemples.)
- xdvi et les fontes ttf : où mettre le fichier ttfonts.map et les fontes TTF (ou alors, corriger le fichier mktexpk patché par freetype — regardé si cela n'a pas changé dans la dernière version de freetype.
- Peut-on utiliser des fontes TTF avec pdftex ?
- Donner un exemple d'utilisation de fontes TTF avec latin.mtx.
- Utilisation de fintinst avec des fichiers *.afm
- Utilisation des fichiers *.etx.
- Utiliser systématiquement latin.mtx. Que contient textcomp.mtx ?
- Syntaxe d'un fichier *.mtx
- Les arguments de la commande \installfint. Pourquoi le codage doit-il être précisé deux fois ?
- Aller chercher un caractère dans une autre fonte
- Changer le codage d'une fonte : exemple de ae.
- Pour faire des essais simples avec fontinst, quelles fontes prendre (c'est une question très simple : quelles sont les fontes par défaut de LaTeX ? Depuis qu'elles portent des noms évocateurs comme ecrm1728.mf, avec encore plus de numéros qu'avant, je suis un peu perdu.)
- Fonte penchée
- Fonte soulignée (avec du blanc)
- Lettres bicolores
- Lettres dégradées (verticalement, ou centralement)
- faked SC : utiliser la hauteur des lettres lue dans la fonte.
- Quelles sont les différentes dimensions spécifiées dans une fonte ? Comment les modifier (sous LaTeX) ? (Où mettre cette question ?)
- Dans une question séparée : que contiennent les fichiers t1.etx ou ot1.etx ? Comment extraire cette information ? (plus loin, nous avons besoin de la liste des glyphes et de la liste des majuscules, avec les minuscules correspondantes.
- Soulignement continu : rendre l'espace actif.
- vfinst

- Essayer de comprendre mathinst et mathkit (pas tout de suite).
- Utiliser la version PS des fontes CM (par exemple, pour les avoir à n'importe quelle taille).
- utiliser hyperref dans la version LaTeX de ce document
- Les fontes sans sérif sont-elles plus lisibles ?
- Comment changer l'espacement des caractères en mode mathématique ? (par exemple, s'il y a plein de $Diff$ dans un fichier que l'on aimerait compiler mais que l'on n'a pas le courrage de corriger). Procéder comme pour avoir les majuscules droites, mais avec une fonte italique.
- Comment obtenir une foncte manuscrite dont la hauteur des majuscules et des minuscuiles soit correcte ? Utiliser des fontes virtuelles uniquement sur les minuscules (pour les aggrandir) ou uniquement sur les majuscules (non conseillé : il faudrait faire des calculs plus compliqués). On prendra garde à la graisse des caractères résultants.
-
-
-
-