Zpravodaj Československého sdružení uživatelů TeXu

Arnošt Štědrý

# Problémy s fonty v TeXu

*Zpravodaj Československého sdružení uživatelů TeXu*, Vol. 6 (1996), No. 4, 250–268

Persistent URL: http://dml.cz/dmlcz/149773

# Problémy s fonty v TEXu

Arnošt Štědrý

Poslední dobou roste zájem o použití *jiných* fontů v TEXu. Tomuto problému bylo věnováno již několik článků ve Zpravodaji (např. [1, 3, 4, 6]). Všechny však vycházely z premisy, že zpracovávaný materiál (font) je po formální stránce bezchybný. Z vlastní dlouhé praxe vím, že tomu tak není a chyby ve fontech, znemožňující jejich použití, jsou velice záludné. Úkolem tohoto příspěvku by mělo být položení teoreticko-praktického základu k řešení některých potíží.

Tedy abych byl přesný, co myslím *jiným* fontem: jedná se o PostScriptové fonty Type1, Type3, případně TrueTypy. METAFONTové fonty považuji za natolik srostlé s TEXem, že ani do článku o potížích nepatří.

Zaměřme se nyní na PostScriptové fonty, TrueTypy odbudeme nakonec krátkou poznámkou o konverzích.

## Co budeme potřebovat k TEXovské sazbě

Každý uživatel TEXu ví, nebo si četbou např. [2] doplní, že TEX má informaci o fontech rozdělenou do dvou komponent. První část – metrická – je uložena v TFM souborech, podle ní TEX počítá sazbu. Druhá obsahuje tvary písmenek a její formát je závislý na použitém ovladači (typicky jsou to pk-fonty).

### TEXovské metriky

Metriky jsou soubory s koncovkou TFM (zkratka od TEX Font Metrics) – podívejme se trochu zblízka, co obsahují. Programem `tftopl` je převedeme do čtivé podoby, můžeme se v nich i trochu porýpat a pak programem `pltotf` převést zpět. Pamatujme však, že provedeme-li změny, jedná se již o zcela jiný font, což by se mělo projevit i v názvu.

Takže nejprve hlavička:

```
(FAMILY CMR)
(FACE O 352)
```

```
(CODINGSCHEME TEX CS TEXT)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 15736045506)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.333334)
   (STRETCH R 0.166667)
   (SHRINK R 0.111112)
   (XHEIGHT R 0.430555)
   (QUAD R 1.000003)
   (EXTRASPACE R 0.111112)
   )
```

Stručně řečeno hlavička obsahuje základní informace o fontu – do které rodiny patří, jak je kódován, kontrolní součet (pokud byl font vytvořen METAFONTem, je to informace pro ovladač, zdali byly pk-soubory stvořeny ze stejného zdrojového kódu), a nakonec parametry fontu, ke kterým lze přistupovat v TEXu pomocí `\fontdimen`.

**SLANT** Sklonění fontu: odtud se čerpá informace o umístění akcentů a kurzívních korekcích.
**SPACE** Normální mezera mezi slovy.
**STRETCH** Jak se může tato mezera roztáhnout.
**SHRINK** Nebo stáhnout.
**XHEIGHT** Výška písmene X (opět kvůlivá akcentům) v TEXu jednotka ex.
**QUAD** Tohle je emsize neboli čtverčík, v TEXu em, tedy šířka písmene M.
**EXTRASPACE** Mezera, která se dává mezi větami v anglofonních oblastech.

V matematických fontech je v této části údajů mnohem více.

Následuje informace o ligaturách a kerningu:

```
(LIGTABLE
   (LABEL O 40)
   (KRN C l R -0.277779)
   (KRN C L R -0.319446)
```

```
   (STOP)
...
 (LABEL O 41)
   (LIG O 140 O 74 )
   (STOP)
atd
)
```

První část zhruba říká, že znak s oktalovým kódem 40 má se znakem l kerning −0.277779 a se znakem L −0.319446. V druhé části se specifikuje, že znak na místě 41 následován znakem 140 dává znak 74.

Následují informace o jednotlivých znacích:

```
(CHARACTER O 54
   (CHARWD R 0.277779)
   (CHARHT R 0.105556)
   (CHARDP R 0.194445)
   (COMMENT
      (LIG O 54 O 376)
      )
```

Tedy znak 54 (čárka, opět oktalově) má definovánu šířku, výšku a hloubku (v komentáři uvedeno, že tvoří ligaturu sama se sebou, a to spodní uvozovky). Podrobněji např. [7].

**PostScriptové metriky**

I PostScriptové fonty mají své metriky. Nejvíce metrických informací je uloženo v souborech s koncovkou AFM (Adobe Font Metrics), které lze prohlížet přímo bez nějakých filtrů. Tedy do toho:

```
StartFontMetrics 2.0
Comment Copyright (c) 1984 Adobe Systems Incorporated.
Comment Creation Date:Mon Mar 31 17:10:33 PST 1986
FontName Times-Roman
EncodingScheme AdobeStandardEncoding
FullName Times Roman
FamilyName Times
Weight Roman
ItalicAngle 0.0
IsFixedPitch false
UnderlinePosition -109
```

```
UnderlineThickness 49
Version 001.000
Notice Times Roman is a trademark of Allied Corporation.
FontBBox -170 -217 1024 896
CapHeight 662
XHeight 448
Descender -217
Ascender 682
```

Opět hlavička s mnoha informacemi, jednotky jsou v PostScriptových (Didotových) bodech[1] (72 do palce) vynásobeno stem.

Následují informace o jednotlivých znacích:

```
StartCharMetrics 210
...
C 33 ; WX 333 ; N exclam ; B 109 -14 224 676 ;
...
C 102 ; WX 333 ; N f ; B 20 0 383 682 ; L i fi ; L l fl ;
C 174 ; WX 556 ; N fi ; B 33 0 521 678 ;
...
C -1 ; WX 722 ; N Aacute ; B 15 0 706 892 ;
...
EndCharMetrics
```

Po C je uvedeno číslo znaku (dekadicky v rozsahu 0–255). Pokud je tam $-1$, znamená to, že znak je kompozitní (složený z více znaků) a jeho definice bude uvedena později. Za WX je uveden posun sázecího bodu, tedy vlastně šířka znaku. Po N následuje jméno (Aacute znamená velké A s čárkou). Následuje BoundingBox, první dvě čísla udávají levý spodní roh, druhá dvě čísla pravý horní roh, nejprve x-ová osa, poté y-ová osa. Za položkou L je uvedeno, tvoří-li znak s nějaký jiným znakem ligatury, tedy f tvoří se znakem i ligaturu, která se jmenuje fi. Číslo znaku udává pozici v tzv. encoding vektoru, o kterém si povíme v kapitole o PFB a PFA souborech.

Následují kerningové informace:

```
StartKernData
StartTrackKern 3
Comment Light kerning
```

[1] Podle definice, 1 m = 2660 Didotových bodů (v TEXu se značí dd). Velikost PostScriptového bodu je jiná, v TEXu se značí bp, 1 in = 72 bp. Pozn. red.

```
TrackKern -1 14 0 72 -1.89
...
EndTrackKern
StartKernPairs 113
KPX A y -92
KPX A w -92
...
EndKernPairs
EndKernData
```

Jak vidno, tento oddíl sestává ze dvou částí: v první se definuje kerning takříkajíc plošný (bezkontextový). Při určování rozpalů se nehledí na jednotlivé znaky, ale mezi každé dva se vrazí mezera jisté velikosti závislá na TrackKernu a velikosti písma. Stručně řečeno, pro písma menší než 14 pt je kerning 0, pro písma větší než 72 pt je kerning $-1.89$, pro písma mezi se kerning spočítá lineární interpolací.

Druhá část se týká kontextového (párového) kerningu a říká: mezi písmeny A a y stáhni mezeru o 0.92 pt. Nutno podotknout, že zde může být uveden i vertikální kerning, pokud KPX nahradíme KPY, případně KP, pokud chceme užít obě posunutí.

Zbývá již jen popsat kompozity:

```
StartComposites 56
CC Zcaron 2 ; PCC Z 0 0 ; PCC caron 139 214 ;
...
EndComposites
EndFontMetrics
```

Tedy znak Zcaron sestává ze dvou komponent, první – Z – je položena na pozici 0 0, druhá – caron (háček) – je posunuta na pozici 139 214. Hloubavější zájemci o formát AFM nechť nahlédnou do [8].

### Program `afm2tfm`

Z předchozího výkladu je zřejmé, že se obě metriky rámcově podobají. Jediný podstatný rozdíl (pomineme-li, že AFM-soubory mohou obsahovat ještě informace pro směr sazby a mnoho „jemnůstek“) jsou kompozitní znaky a fakt, že v TFM souboru může být nejvýše 256 znaků. Avšak TEX pro zhotovení (spočtení) sazby nijak nepotřebuje informaci o tom, jak jsou které znaky vytvořeny. Proto se tato informace uchovává odděleně v takzvaném virtuálním fontu, o kterém si povíme za chvíli.

Nikoho v tuto chvíli asi nepřekvapí, že lze z AFM souboru získat TFM soubor, a to pomocí programu `afm2tfm`. Program rozumí kerningům, ligaturám i kompozitním znakům.

Nejjednodušší, co můžeme udělat, je napsat:

```
afm2tfm Times-Roman rptmr
```

Program pak ze souboru Times-Roman.afm vybere základní znaky (t. j. ty které mají číslo 0–255) a vytvoří TFM soubor z těchto znaků, kerningy, ligatury a kompozity zanedbá. Tento font se pak nazývá *raw* (viz to r na začátku názvu rptmr).

Takový font je nám vlastně k ničemu, mnohem častěji použijeme volání

```
afm2tfm Times-Roman -v ptmr rptmr
```

a výsledkem kromě souboru `rtpmr.tfm` bude i soubor `ptmr.vpl`. Přípona VPL (virtual property list) znamená, že soubor je mixem TFM informací a informací virtuálního fontu. Je to tedy jakási obdoba AFM souboru (t. j. obsahuje metrické, kerningové i kompozitní informace). Jediná slabost VPL souboru spočívá v tom, že může obsahovat pouze 256 znaků, zatímco AFM soubor jich smí obsahovat libovolné množství (ale jen 256 jich má svoje číslo). Podívejme se nyní na soubor `ptmr.vpl`:

```
(VTITLE Created by afm2tfm ptmr0 -v ptmr)
(COMMENT Please edit that VTITLE if you edit this file)
(FAMILY TeX-rptmr)
(CODINGSCHEME TeX text + AdobeStandardEncoding)
(DESIGNSIZE R 10.0)
(DESIGNUNITS R 1000)
(COMMENT DESIGNSIZE (1 em) IS IN POINTS)
(COMMENT OTHER DIMENSIONS ARE MULTIPLES OF DESIGNSIZE/1000)
(CHECKSUM O 7575461244)
(FONTDIMEN
   (SPACE D 250)
   (STRETCH D 200)
   (SHRINK D 100)
   (XHEIGHT D 448)
   (QUAD D 1000)
   (EXTRASPACE D 111)
   )
```

```
(MAPFONT D 0
   (FONTNAME rptmr)
   (FONTCHECKSUM O 30202316533)
   )
...
```

Vidíme, že se podobá TFM souboru, pouze přibyly položky MAPFONT, jimiž specifikujeme, z kterých fontů se bude výsledný font skládat, a VTITLE.

Dále ligační schéma:

```
(LIGTABLE
   (LABEL O 40)
   (LIG C L O 350)
   (LIG C l O 370)
   (STOP)
   (LABEL O 41)
   (LIG O 140 O 16)
   (STOP)
   (LABEL O 47)
   (LIG O 47 O 272)
   (KRN O 47 R -74)
   (KRN C s R -55)
   (KRN C t R -18)
   (STOP)
...
```

Tedy nic nového.

A specifice znaků:

```
(CHARACTER O 15 (comment quotesingle)
   (CHARWD R 180)
   (CHARHT R 685)
   (MAP
      (SETCHAR O 251)
      )
   )
```

Vidíme, že na pozici 15 se namapoval znak 251 z fontu rptmr, který má dané rozměry (pokud změníme šířku znaku, můžeme tak vlastně vytvořit proložený font),

nebo:

```
(CHARACTER O 43 (comment numbersign)
   (CHARWD R 500)
   (CHARHT R 662))
```
znamená, že se na tuto pozici mapuje stejný znak (43), nebo konečně kompozity:
```
(CHARACTER O 202 (comment Aacute)
   (CHARWD R 722)
   (CHARHT R 892)
   (MAP
      (SETCHAR O 101)
      (MOVERIGHT R -528)
      (MOVEUP R 214)
      (SETCHAR O 302)
      )
)
```
Tedy na pozici 202 namapuj nejprve znak 101, pak se šoupni doprava (doleva), nahoru a namapuj znak 302. Možnosti virtuálního fontu jsou mnohem širší. Virtuální popis může obsahovat navíc i různé `specialy`, např. PostScriptové. Pěkný příklad je v [7, str. 75–79], za nahlédnutí stojí i [5].

Co tedy dál? Na soubor `ptmr.vpl` zavoláme program:

```
vpltovf ptmr.vpl
```

který nám vytvoří soubory `ptmr.tfm` a `ptmr.vf`. První potřebuje TEX pro sazbu, druhý (tzv. virtuální font) je vyžadován ovladačem pro zobrazení. Pokud ovladač nepodporuje virtuální fonty (např. dviwin pro Windows), pak si lze pomoci programem `dvicopy`, který dvi-soubor „odvirtualizuje“ (nahradí v něm kompozity skutečnými znaky).

Kódování výsledného fontu se ve spodní části kryje s kódováním cm-fontů (pokud to font umožňuje), ve vrchní části pak je doplněno původní kódování. Nejčastější změnou, kterou chceme s fontem udělat, je právě změna kódování (na kódování $\mathcal{CS}$-fontů). Program `afm2tfm` to umožňuje hned dvěma způsoby:

1. Překódování *raw*-fontu (tedy toho, na který odkazujeme z virtuálního fontu). Tato metoda se hodí hlavně v případě, že máme velký font t. j. má více než 256 skutečných (nekompozitních) znaků. Toto tvrzení se zdá poněkud ve sporu s předchozími prohlášeními, že v AFM souboru jich může být pouze 256. Takové obludy snadno

vznikají např. konverzí z TrueTypů, a možná že je novější verze AFM dokonce podporují. Pokud je nějaký znak, na který se chceme odkazovat, mimo rozsah 0–255, pak nezbývá, než ho posunout na jiné místo, neboť `afm2tfm` odmítne na takovýto znak napsat odkaz (nemluvě už o TEXu a ovladačích). Pokud pro zobrazení a tisk používáme pk-bitmapy generované programem `ps2pk`, musíme použít parametru `-E` pro překódování. Pokud chceme využít DVIPS a PFB font, pak do souboru `psfonts.map` napíšeme něco jako:

```
rptmr Times-Roman "CSencoding ReEncodeFont" <myenc.enc
```

viz kapitola o DVIPS. Dlužno podotknouti, že se mi toto nikdy nepodařilo[2], t. j. nikdy se mi nevygeneroval ten správný postscript. Volání programu `afm2tfm` je v tomto případě:

```
afm2tfm Times-Roman -p myenc.enc -v ptmr rptmr
```

2. Druhá metoda nechává *raw*-font na pokoji (tedy vlastně počítá s tím, že má nejvýše 256 znaků) a mění VPL, potažmo VF font, tedy znaky pouze přemapuje. Lze použít i s pk-fonty, i s DVIPS poměrně bez problémů. Její volání je

   ```
   afm2tfm Times-Roman -t myenc.enc -v ptmr rptmr
   ```

**Encoding**

v obou předchozích metodách jsme potřebovali soubor myenc.enc, který specifikuje naše kódování. Povězme si nyní něco málo o něm.

Vlastní soubor sestává ze dvou částí (nepočítaje komentáře). První je vlastně Encoding vektor (PostScriptový array) a má přesně 256 (jaký to div) položek. Číslo položky je specifikováno jejím pořadím. Tedy:

```
/CSencoding [ /Gamma /Delta /Theta /Lambda /Xi /Pi /Sigma
/Upsilon1 /Phi /Psi /Omega /ff /fi /fl /ffi /ffl /dotlessi
/dotlessj /grave /acute /caron /breve /macron /ring
/cedilla /germandbls /ae /oe /oslash /AE /OE /Oslash
/.notdef /exclam /quotedblright /numbersign /dollar
/percent /ampersand /quoteright /parenleft /parenright
```

[2] Redakčním zásahem byl vyhozen konec řádku ve výše uvedeném příkladu, kde bylo napsáno `<ptmr.pfb`, a to hned ze dvou důvodů. Jednak se tak dlouhý řádek nevešel to tiskového zrcadla, a navíc právě zde je příčina neúspěchu. Font Times-Roman je totiž rezidentní ve všech PostScriptových zařízeních, takže soubor `ptmr.pfb` na disku obvykle nenajdete. Pozn. red.

```
/asterisk /plus /comma /hyphen /period /slash /zero /one
/two /three /four /five /six /seven /eight /nine /colon
/semicolon /exclamdown /equal /questiondown /question /at
/A /B /C /D /E /F /G /H /I /J /K /L /M /N /O /P /Q /R /S /T
/U /V /W /X /Y /Z /bracketleft /quotedblleft /bracketright
/circumflex /dotaccent /quoteleft /a /b /c /d /e /f /g /h
/i /j /k /l /m /n /o /p /q /r /s /t /u /v /w /x /y /z
/endash /emdash /hungarumlaut /tilde /dieresis /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/perthousand /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /Agrave
/.notdef /.notdef /.notdef /hyphen /ogonek /guillemotleft
/guillemotright /.notdef /Acedilla /breve /Lslash /currency
/Lcaron /Sacute /section /dieresis /Scaron /Sogonek /Tcaron
/Zacute /.notdef /Zcaron /Zdotaccent /ring /acedilla
/cedilla /lslash /acute /lcaron /sacute /caron /agrave
/scaron /sogonek /tcaron /zacute /hungarumlaut /zcaron
/zdotaccent /Racute /Aacute /Acircumflex /Abreve /Adieresis
/Lacute /Cacute /Ccedilla /Ccaron /Eacute /Eogonek
/Edieresis /Ecaron /Iacute /Icircumflex /Dcaron /Dslash
/Ndotaccent /Ncaron /Oacute /Ocircumflex /Ohungarumlaut
/Odieresis /multiply /Rcaron /Uring /Uacute /Uhungarumlaut
/Udieresis /Yacute /Togonek /germandbls /racute /aacute
/acircumflex /abreve /adieresis /lacute /cacute /ccedilla
/ccaron /eacute /eogonek /edieresis /ecaron /iacute
/icircumflex /dcaron /dslash /ndotaccent /ncaron /oacute
/ocircumflex /ohungarumlaut /odieresis /divide /rcaron
/uring /uacute /uhungarumlaut /udieresis /yacute
/quotedblbase /quotedblleft ]
```

Vidíme, že znaky jsou uvedeny pod svými adobe názvy. Pokud na daném místě není namapován žádný znak, stojí tam napsáno: `/.notdef` Všimněme si, že kódování má své jméno: `CSencoding`, které musí být unikátní (zvláště pokud užíváme více kódování fontů v jednom dokumentu). Pokud není, mohou později nastat potíže.

Druhá část specifikuje ligatury:

```
% LIGKERN hyphen hyphen =: endash ; endash hyphen =: emdash ;
```

```
% LIGKERN quoteleft quoteleft =: quotedblleft ;
% LIGKERN quoteright quoteright =: quotedblright ;
% LIGKERN exclamdown exclamdown =: guillemotleft ;
% frenchdblquotes
% LIGKERN questiondown questiondown =: guillemotright ;
% csquoteleft
% LIGKERN comma comma =: quotedblbase;
% LIGKERN space {} * ; * {} space ; zero {} * ; * {} zero ;
% LIGKERN one {} * ; * {} one ; two {} * ; * {} two ;
% LIGKERN three {} * ; * {} three ; four {} * ; * {} four ;
% LIGKERN five {} * ; * {} five ; six {} * ; * {} six ;
% LIGKERN seven {} * ; * {} seven ; eight {} * ; * {} eight ;
% LIGKERN nine {} * ; * {} nine ;
% LIGKERN question {} quoteleft ; exclam {} quoteleft ;
```

První řádek říká: dva hypheny dají jeden endash, endash a hyphen dá emdash. Příkazy typu `one {} *` říkají: odstraň kerning mezi jedničkou a čímkoliv (např. `* {} *` odstraní všechny kerningy, notace `a{}b` je podobná TEXovské). Více (např. koncové ligatury) v manuálu k DVIPS.

Program `afm2tfm` má ještě několik jiných možností jako například: natahování (zužování) znaků pomocí parametru `-e`, vytváření kapitálek pomocí `-c`, nebo skloněného fontu pomocí `-s`, jimiž se však nebudeme na tomto místě zabývat. Dlužno podotknouti, že původně měl obsahovat i vytvoření prokládaného fontu, od této myšlenky se však upustilo pro potíže s ligaturou fi (a ostatními). Nebylo jasné, má-li se trhat, nebo ponechávat. Více manuál DVIPS: A TEX Driver, nebo článeček [1].

## Outliny a bitmapy

Takže už umíme připravit metriky a zbývá naučit driver zobrazovat správně znaky. Nejprve tedy trocha teorie:

### PFB, PFA a ostatní

Existuje stará pověra:

> *PostScriptové fonty jsou dvou typů Type1 a Type3, první mají příponu PFA, druhé PFB.*

Jak to tedy je? Za prvé existují asi čtyři druhy PostScriptových fontů Type1, Type3, Type4 a Type5, z niž druhé dva se od prvního liší pouze formátem (Type4 je formát pro uložení na disku, Type5 pro uložení v tiskárně). Ovšem přípony PFA a PFB se obě týkají formátu Type1. První říká, že jde o ASCII kódování, druhá, že jde o kódování binární (úsporné).

Co jsou tedy vlastně Type1 a Type3 zač:

**Type1** Vektorový popis fontu, který neobsahuje „plný“ PostScript. Obsahuje pouze operace pro vykreslení obrysu „lineto, curveto...“ plus navíc hintovací operace opravující špatný vzhled bitmap v nižších rozlišeních (potažmo v malých velikostech). Protože jejich jazyk je slabý, lze je vykreslit poměrně rychle. O to se stará ATM (Adobe Type Manager), který je rastruje a výsledné bitmapy posílá aplikaci.

**Type3** Využívá „plný“ PostScript. Je vhodný zejména pro bitmapové fonty nebo různě divoké výplně fontů (třeba fraktálové, máte-li dost času na tisknutí). Neobsahuje hinty (to není záležitost PostScriptu) a ATM jim nerozumí.

Asi už vám došlo, že dál budeme sjíždět pouze Type1. Nejprve je trochu rozpitváme, abychom lépe porozuměli tomu, co se děje pod pokličkou. Použijeme sadu volně dostupných operačních nástrojů T1UTILS. Programem T1DISASM převedeme font do čitelné podoby. Trochu se podívejme na výsledek. Nejprve hlavička:

```
%!PS-AdobeFont-1.0
%%CreationDate: Sat Feb 04 01:32:58 1995
%%VMusage: 27904 27904
% Created with FontMonger Copyright (c) 1991-2 ...
11 dict begin
/FontInfo 9 dict dup begin
/version (001.000) readonly def
/Notice () readonly def
/FullName (HelveticaInseratE  Medium) readonly def
/FamilyName (HelveticaInseratE-) readonly def
/Weight (Medium) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -100 def
```

```
/UnderlineThickness 50 def
end readonly def
/FontName /HelveticaInseratEMedium def
```

Nejedná se vlastně o hlavičku, ale o počáteční definice. Následuje Encoding Vector (zkráceno):

```
/Encoding 256 array
0 1 255{1 index exch /.notdef put}for
dup 32 /space put
dup 33 /exclam put
dup 34 /quotedbl put
dup 35 /numbersign put
dup 36 /dollar put
dup 37 /percent put
dup 38 /ampersand put
dup 39 /quotesingle put
dup 40 /parenleft put
dup 41 /parenright put
dup 42 /asterisk put
dup 43 /plus put
...
readonly def
```

Způsobů zadávání Encodingu je celá řada. Nelekněte se proto, když narazíte na jiný složitější (podívejte se např. na font Utopia). Zde vidíme, že celý array se nejprve vynuluje (vynotdefuje), a pak se teprve plní. Porovnejte jej s kódováním v příslušném AFM souboru, měly by být stejné.

Následují další obecné definice a nakonec jednotlivé znaky:

```
/space {       0 250 hsbw
        endchar
        } ND
/exclam {      84 334 hsbw
        0 166 vstem
        0 184 hstem
        779 20 hstem
        799 vmoveto
        -312 vlineto
        38 -256 rlineto
```

```
90 hlineto
38 256 rlineto
312 vlineto
closepath
-799 vmoveto
184 vlineto
-166 hlineto
-184 vlineto
closepath
endchar
} ND
```

Pro neznalce PostScriptu podotýkám, že vše je v obrácené notaci, nejprve operandy, pak operátory. Řádky končící na hstem nebo vstem obsahují právě hintovací informace (z jejich výskytu lze usoudit kvalitu fontu).

## Jak to použít v TEXu

Takže máme AFM a PFB soubor a chceme vytvořit fungující (dejme tomu český) font. Předpokládejme na chvíli, že již obsahuje české znaky. Dále máme soubor `csenc.enc`, kde je specifikováno české kódování. Pro začátek zvolíme druhou metodu získání TFM souboru:

```
afm2tfm Times-Roman -t csenc.enc -v ptmr8z rptmr
```

Někoho možná napadne, proč zrovna ptmr8z a rptmr. Je to otázka konvence, a to konkrétně „Font Naming Conventions“, jejímž autorem je Karl Berry, a více se lze o ní dočíst v manuálu k DVIPS. Stručně: skládá se ze jména písmolijny, názvu, duktu a asi tří dalších položek (p znamená PostScript (adobe), tm – Times, r – Roman, 8z je kódování). Ze CTANu lze stáhnout (stále se prodlužující) seznam písem s jejich správnými (konvenčními) jmény. Je dobré se touto konvencí řídit, by nedošlo k zmatení jazyků.

Pokud v tomto okamžiku nastane chyba (při volání `afm2tfm` nebo následném `vpltovf`), může to znamenat, že font má více než 256 základních znaků, a my ve virtuálním fontu odkazujeme na nějaký vyšší znak. (Pozná se snadno z toho, co programy hlásí.) Situace se dá řešit několika způsoby:

1. Předělat AFM a PFB soubory k obrazu svému (někde jsem četl hlášku: „God saves the T1UTILS“). Pozor na práva, někteří au-

toři jsou hákliví na své dílo (byť jsou jeho kvality pochybené). Lze k tomu s uspěchem použít i komerční programy FontMonger a FontoGrapher.

2. Využít první metodu a překódovat *raw*-font a doufat, že konečně někdo rozchodí i ReEncodeFont. Jinak jsme odkázáni na `ps2pk`.

Některé verze `vptovf` se špatně snášejí s některými verzemi `afm2tfm`, zejména si nerozumí v koncích řádků. Lékem většinou bývá filtr `unix2dos` (nebo prosté načtení do některého z editorů a následné uložení). Občas vygeneruje `afm2tfm` dokonce špatný VPL soubor (pozná se to však až při prohlížení), a je potřeba celou operaci provést znovu. (Neznám genealogii této chyby, ale v poslední verzi programu jsem se s ní již nesetkal.)

Konečně přichází fáze testování. Já osobně používám program `ps2pk` k vygenerování bitmap, protože funguje pro obě varianty překódování fontu.

**Ps2pk**

Je Public Domain balík založený na komerčním rastrovacím programu, který IBM darovala X-consortiu. Umí tedy rastrovat (včetně hintingů) Type1 fonty do pk bitmap v různých rozlišeních a velikostech. Navíc umí překódování fontu, roztažení a sklonění fontu (dalo by se říci, že co dělá `afm2tfm` pro AFM soubory, dělá `ps2pk` pro PFB soubory). V nové verzi (určitě si ji stáhněte) je navíc spojen s programem, udržujícím resource databázi, a obdobou MFjobu, generující chybějící fonty (a to jak METAFONTové, tak Type1). Pokud si celý balík správně nainstalujete a udržujete, nemusíte se nadosmrti starat, co je jaký font, a jak je překódován. Pokud vývoj půjde tímto směrem, zřejmě se `ps2pk` stane jakýmsi ATM pro TEX.

Program má i své chyby. Například všechny znaky jsou generovány o jeden dot výše, což se projeví, pokud spojujete Type1 fonty např. s cm fonty, ale při rozlišení 300 dpi je na to potřeba už pořádná lupa. Druhý problém je s hintingy. Vyskytnou se fonty, které `ps2pk` vyrastruje příšerně na rozdíl od PostSriptové tiskárny, a navíc je to viditelné i ve vyšších rozlišeních. Smůla.

Více tedy manuál k `ps2pk`. Obecná poučka zní: výsledné pk-fonty musí mít stejné jméno a kódování jako *raw*-fonty (podobně je tomu s rozšířením a skloněním). Takže pokud jste generovali TFM soubor pomocí:

```
afm2tfm Times-Roman -p csenc.enc -v ptmr8z rptmr
```
voláme pak `ps2pk`
```
ps2pk -E csenc.enc Times-Roman rptmr.pk
```
V případě
```
afm2tfm Times-Roman.pfb -t csenc.enc -v ptmr8z rptmr
```
použijeme
```
ps2pk Times-Roman.pfb rptmr.pk
```
Umístíme nyní TFM, VF, a PK soubory na správná místa (aby je ovladač našel) a zkusíme si vytisknout tabulku fontu. Používám k tomu makro `testfont`. Výslednou tabulku porovnáme s tabulkou fontu csr10. Pokud je vše O.K., můžeme si blahopřát a přejít k dalšímu kroku. Pokud pouze chybí některé znaky, podíváme se do AFM souboru, zda jsou ve fontu obsaženy. Pokud skutečně nejsou, nebo pouze chybí kompozitní popisy (jako třeba ve fontu Utopia), můžeme s výhodou použít program a2ac Petra Olšáka. Někdy se stane, že v AFM souboru jsou znaky označeny jako kompozity, chybí v nich jejich popis (CC položka), ale v PFB souboru jsou obsaženy jejich obrysy. Pak stačí do AFM souboru opsat číslo z PFB. Při té příležitosti není na škodu zkontrolovat, jestli obě kódování souhlasí. Rovněž stojí za to zkontrolovat, zdali mají všechny znaky uvedené v Encoding vektoru odpovídající popisy v PFB souboru, případně, jsou-li všechny obrysy uvedeny v Encoding vektoru (zbytečně bychom tak přišli o nějaké znaky).

Nejhorší případ nastane, když znaky nejsou na správných místech. Pak je třeba nejprve pro jistotu zkontrolovat náš kódovací soubor a není-li chyba v něm, pak vězte, že máte co do činění s chybou nejstrašnější, bohužel častou. Někteří samozvaní počešťovatelé nejen že vyházejí z fontu ligatury (tomu jsme již přivykli), leč se nenamáhají dát obrysům jejich pravá jména (dlouhé A pak např. najdeme pod názvem iacute). Řešení je vcelku bolestivé. Zjistíme správnou permutaci buď pomocí komerčního software (používám FontoGrapher) nebo pomocí GhostScriptu nebo lze využít `pkedit` a námi vygenerovanou bitmapu. Pak např. awkovským skriptem za tímto účelem zhotoveným proženeme „vadné“ soubory a zjednáme nápravu. Celý postup opakujeme.

Většinou zbývá doladit několik jemnůstek. Pokud například víme, že ve fontu je obsažena ligatura fi, ale nám se ne a ne zjevit, je to způsobeno většinou absencí příznaku `L i fi` v AFM souboru a není těžké jej doplnit.

### DVIPS

Konečně jsme tedy font odladili a zbývá jej rozchodit s DVIPS. Pokud jsme nepoužívali překódování *raw*-fontu, je to relativně snadné. Doplníme do souboru `psfonts.map` řádek:

```
rptmr Times-Roman <Times-Roman.pfb
```

První jméno je *raw*-font, druhé je FontName z AFM souboru a třetí je PFB soubor. Umístíme pak `Times-Roman.pfb` na místo, kde ho DVIPS najde a pokusíme se nyní vytisknout tabulku fontu pomocí DVIPS. Pokud vše proběhne v pořádku (DVIPS nenahlásí žádnou chybu a opravdu font natáhne), ale tiskárna vytiskne vše fontem Courier (psací stroj), podíváme se ještě, zdali `FontName` je stejné v AFM i PFB souboru. Pokud ne, sjednotíme je (nejlépe tak, že to opravíme v souboru AFM a `psfont.map`, je to nejméně předělávání a nemusíme znova spouštět `afm2tfm`)

Pokud jste použili metodu s překódováním *raw*-fontu, v manuálu DVIPS je psáno, že funguje vložení následujícího řádku do `psfonts.map`:

```
rptmr Times-Roman "CSencoding ReEncodeFont" <csenc.enc
```

kde název `CSencoding` musí být nachlup stejný, jaký je v souboru `csenc.enc`. Podotýkám, že mně to zatím nechodí[3], snad budete šťastnější (v poslední distribuci DVIPS je celý direktorář věnovaný tomuto problému).

### Varování

I když máte font legálně koupený, nelze věřit takřka ničemu. Sám jsem opravoval font, kde se neshodovalo: FontName v AFM a PFB souboru, některé znaky byly uvedeny v AFM souboru jako kompozitní, byť jejich obrysy byly obsaženy v PFB souboru (a pochopitelně jejich kompozitní popisy v AFM souboru chyběly), a některé jiné obrysy obsažené v PFB souboru nebyly ani v jednom Encoding vektoru (jednalo se o písmena ť a ď, čili nic zanedbatelného). Nejpodivnější bylo, že se celý font choval pod Windows správně[4].

[3] Viz poznámku na str. 258 nebo též kapitolu 5.3 v dokumentaci programu DVIPS. Pozn. red.

[4] Vysvětlení je nasnadě. ATM pro Windows nepoužívá AFM soubory. Část metrických informací (BoundingBox) čerpá přímo z PFB souboru, druhou část (kerning) má uloženu v PFM souboru (speciální binární tvar). Na Encoding se zřejmě neohlíží vůbec.

## TrueTypy

Alternativou k Type1 jsou TrueTypy firem Apple a Microsoft. Jejich hlavní výhodou oproti PostScriptovým fontům má být automatické odstraňování „dropoutů“, tedy přerušení čáry v nižších rozlišeních, bohatší možnosti hintování a rychlejší vykreslování kvadratických splajnů (Type1 užívají kubické). Projeví se to zejména na obrazovce. Obecně lze bohužel říci, že TrueTypy jsou horší kvality nežli Type1, a to z několika důvodů:

1. Většinou vznikly špatnou konverzí z Type1. Konverze obrysů je nepoměrně jednoduší (byť netriviální), než konverze hintů.
2. Software na výrobu dobře ohintovaných TrueTypů je poměrně drahá záležitost (rozhodně to nezvládne ani FontoGrapher ani FontStudio) a hinty samotné jsou nezřídka ruční práce. V době psaní tohoto článku jsem neznal žádný dostupný WYSIWYG software (pro Windows) produkující skutečné TrueTypy. Vždy se jednalo o kubické splajny posléze konvertované na kvadratické, které ovšem program tajil (nezobrazoval).
3. Formát TrueTypů se zcela macešsky chová k ligaturám. To neznamená, že tam nejsou, ale pouze chybí informace, které páry tvoří jaké ligatury. Sázecí software může pouze tipovat.

Zatím jedinou možností, jak implantovat TrueTypy do TEXu, je jejich konverze do Type1[5]. Naštěstí konverze kvadratických splajnů na kubické dopadá veskrze lépe než obrácená, nemluvě o tom, že některé softwary umí zrestaurovat původní (kubické) obrysy, vznikl-li font např. konverzí z Type1.

Žel neznám jiný než komerční konverzní software.

**FontMonger** je program na tvorbu vektorových fontů zvládající i konverzi mezi mnoha formáty. Verze, kterou jsem užíval, měla potíže s kódováním. Důsledkem bylo zrušení některých znaků ve fontu během editace či konverze, čili nešla použít na většinu počeštěných fontů.

**FontoGrapher** neumí konvertovat mezi tolika formáty jako FontMonger, ale na rozdíl od něho se nezalekne ani skurilních kódování

[5] Existuje sice komerční verze TEXu TrueTEX, ale nepředpokládám její hojný domácí výskyt.*

*Pracuje se na konvertoru TTF2AFM (nekomerčním), který umožní použití TrueType fontů v TEX2PDF. Pozn. red.

ani více znaků ve fontu než 256. Rovněž zvládá i vektorizaci (malých) bitmap, což jej povyšuje na vhodný nástroj k domácí výrobě Type1 (resp. TrueType) fontů. Nezanedbatelná je i (vypnutelná) autohintovací funkce.

Hodně štěstí

## Literatura


[1] Petr Sojka: Virtuální fonty, accents a přátelé. Zpravodaj Československého sdružení uživatelů TEXu, 1994, # 2, str. 56–69.
(`ftp://ftp.muni.cz/pub/tex/local/cstug/sojka/aboutacc/`)

[2] Joachim Schrod: Komponenty TEXu. Zpravodaj Československého sdružení uživatelů TEXu, 1993, # 1, str. 1–10.
(`ftp://ftp.muni.cz/pub/tex/CTAN/documentation/components-of-TeX/`)

[3] Tomáš Mráz: PostScriptová písma v TEXu. Zpravodaj Československého sdružení uživatelů TEXu, 1994, # 2, str. 70–76.

[4] Petr Olšák: Program a2ac – manipulace s fontem na úrovni PostScriptu. Zpravodaj Československého sdružení uživatelů TEXu, 1996, # 1, str. 23–30.
(`ftp://math.feld.cvut.cz/pub/cstex/msdos/cspsfont.zip`)

[5] Donald Knuth: Virtual fonts, a more fun for grand wizards. TUGboat, 1990, # 1, str. 13–23.
(`ftp://ftp.muni.cz/pub/tex/CTAN/documentation/virtual-fonts.knuth`)

[6] Pavel Herout: PostScriptové fonty pro ty, co o nich moc nevědí. Zpravodaj Československého sdružení uživatelů TEXu, 1996, # 1, str. 43–64.

[7] Petr Olšák: Typografický systém TEX. Československé sdružení uživatelů TEXu, 1995.

[8] Adobe Font Metrics File Format Specification Version 4.0. Adobe Systems Incorporated, February 1992.



*Arnošt Štědrý*
`arnost@uivt.cas.cz`