Zpravodaj Československého sdružení uživatelů TeXu

Pavel Janík ml.

Písma v PostScriptu

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 9 (1999), No. 4, 201–235

Persistent URL: http://dml.cz/dmlcz/149861

Terms of use:

© Československé sdružení uživatelů TeXu, 1999

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.

This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://dml.cz

Odkazy

[1] L. Dobiáš: *Začínáme s METAPOSTem aneb Udělejte si vlastní logo.* Zpravodaj Československého sdružení uživatelů TEXu, **8** (3–4), 167–175 (1998), R. Špalek: *METAPOST.* Zpravodaj Československého sdružení uživatelů TEXu, **8** (3–4), 175–218 (1998).
[2] `http://therion.homepage.com`
[3] `http://plan9.bell-labs.com/cm/cs/who/hobby/mphist.html`
[4] P. Šedivý, M. Brož, J. Gřondilová, M. Píše, K. Houfek: *Kreslíme METAFONTem.* Zpravodaj Československého sdružení uživatelů TEXu, **8** (1), 1–63 (1998).

Martin Budaj
`Martin.Budaj@st.fmph.uniba.sk`

Písma v PostScriptu

Pavel Janík ml.

V tomto článku bude nastíněn význam jazyka PostScript pro počítačovou sazbu a formáty písem použitelné v PostScriptu. Popíšeme si také reprezentaci fontu v postscriptovém interpretu a odlišné položky ve slovníku fontu pro různé formáty písem. Hlavní důraz bude kladen na formáty Type 3 a Type 1.

Jazyk PostScript

PostScript je jazyk pro popis stránky. Je to interpretovaný programovací jazyk, který obsahuje dostatečné prostředky pro kompletní popis tiskového materiálu, použitých barev a fontů. Vznikl v roce 1985, kdy společnost Adobe Incorporated tento jazyk otevřela i pro odbornou veřejnost (některé prameny uvádějí vznik v roce 1982 [2]). V té době bylo velmi obtížné sdílet dokumenty případně připravovat podklady pro tiskárny. Každá tiskárna totiž používala proprietární formát a dokonce i fonty specifické pro daný typ tiskárny. V těchto podmínkách bylo téměř nemožné vyměňovat dokumenty, a tak byl formát PostScript, který je nezávislý na rozlišení výstupního zařízení, přijat přívětivě i odbornou veřejností. V roce 1985 byly uvedeny i první produkty obsahující tzv. *postscriptový interpret*. Tiskárny Apple LaserWriter obsahovaly vestavěný interpret jazyka PostScript.

Kompletní specifikace jazyka PostScript je vydávána knižně [6]. Jednotlivé verze jazyka jsou číslovány pomocí tzv. *Levelu*. První verze jazyka je nazývána PostScript Level 1. Druhá verze je Level 2 a aktuální verze jazyka PostScript je Level 3. Specifikace PostScript Level 3 je k dispozici i na webové stránce společnosti Adobe určené partnerům (`http://partners.adobe.com/`). Referenční postscriptový interpret společnost Adobe neuvolnila, ale společnost Aladdin Enterprises (`http://www.ghostscript.com/`) pravidelně uvolňuje nové verze svého interpretu Ghostscript pod licencí GNU GPL.

Formáty písem v PostScriptu

Protože je jazyk PostScript určen především pro popis tiskové strany, musí mít optimalizovánu práci s texty a s jednotlivými znaky. V této kapitole se zmíníme o tom, jak jsou jednotlivá písma v PostScriptu organizována, jak přidáme nové písmo do našeho dokumentu a také jak nové písmo v PostScriptu použijeme.

Ještě než se zmíníme o podrobnostech práce s fonty v PostScriptu, budeme si muset osvětlit používanou terminologii. Pod pojmem *znak* budeme rozumět symbol, tedy nikoli jeho vlastní tvar, sklon či jiné charakteristiky, ale spíše jeho abstraktní reprezentaci. Znakem rozumíme jednotlivá písmena abecedy: A, B, C apod. Reprezentaci znaku v daném fontu nazýváme *písmeno* (*glyph*). Písmenem tedy jsou např. A, B, C, tedy grafické reprezentace znaků A, B a C z fontu Helvetica. Pojmem *font* budeme chápat množinu písmen reprezentujících skupinu znaků ve specifické reprezentaci. Font je možné chápat jako program, který je nutno před jeho použitím provést. Každé písmeno je v tomto programu reprezentováno podprogramem.

Poté, co je font reprezentovaný programem, spuštěn, vytvoří si postscriptový interpret tzv. *slovník fontu*, který je možné posléze využívat pro získání grafické reprezentace jednotlivých písmen.

Samotné rastrování písmen již probíhá v režii postscriptového interpretu. Uživatel pouze specifikuje, který znak z kterého fontu (slovníku) chce použít a postscriptový interpret si pomocí slovníku příslušného fontu najde proceduru, která příslušné písmeno vytvoří.

PostScript podporuje několik typů fontů, které rozlišuje podle položky **FontType** ve slovníku fontu. Hodnotou této položky je nezáporné celé číslo. Proto hovoříme o fontech Type 0, Type 1, Type 42 apod. Každý typ fontu má jasně danou strukturu. V této práci se zmíníme pouze o fontech Type 1, Type 3 a krátce také o Type 42. Fonty Type 1, 2, 3, 14 a 42 jsou nazývány *základními fonty*.

Fonty Type 0 jsou tzv. *složené fonty*. Je možné je přirovnat k virtuálním fontům v TEXu. Libovolné písmeno tohoto fontu může být převzato z jiného fontu a výsledný font tak bude „složený“ z původních fontů. Fonty Type 1 mají jasně danou strukturu, která je popsána v knize Adobe Type 1 Font Format [9].

Většina latinkových fontů společnosti Adobe je dostupná právě v této podobě. Fonty ve formátu Type 1 používají omezenou sadu postscriptových operátorů, což zajišťuje jednodušší zpracování a vyšší rychlost vykreslování písmen. Formát Type 2 se nazývá Compact Font Format (CFF). Bližší informace o tomto formátu je možné nalézt v knize PostScript Language Reference Manual [6] a ve specifikaci formátu CFF [14] dostupné na serverech společnosti Adobe. Fonty ve formátu Type 3 obsahují pro každý znak kompletní postscriptový program definující vzhled vyrastrovaného písmene. Při jeho vytváření je možné použít jakéhokoli postscriptového operátoru.

Formáty Type 4 a Type 5 slouží pro trvalé uchovávání fontů v paměti tiskárny. PostScript Level 2 totéž umožňuje i pro fonty Type 1, a proto je formát Type 4 již zastaralý. Formát Type 5 se liší od Type 1 pouze strukturou souboru, nikoli jeho obsahem, který je stejný jako u Type 1. Oba tyto formáty jsou proprietární, a proto jsou i nedokumentované. Formát Type 42 je založen na formátu True Type. Je to pouze jakási obálka nad vlastním True Type fontem. Další typy fontů (Type 9, 10, 11, 14 a 32) jsou popsány v knize PostScript Language Reference Manual [6] a na webových stránkách společnosti Adobe (`http://partners.adobe.com/supportservice/devrelations/japan/typeforum/ftypes.html`).

Položky ve slovníku fontu

Slovník fontu je reprezentací programu fontu v postscriptovém interpretu. Jeho jednotlivé položky obsahují informace specifické pro daný typ fontu. Existuje však množina vlastností, které jsou společné pro všechny typy fontů a mohou tedy být uvedeny ve slovníku každého fontu. Tyto vlastnosti si popíšeme v tabulce 1.

Základní fonty mohou navíc kromě těchto položek ve slovníku fontu obsahovat i další položky, které jsou přehledně shrnuty v tabulce 2. Přesný význam jednotlivých položek bude vysvětlen na straně 211 na příkladu fontu ITC Anna.

Type 3 fonty

Type 3 fonty jsou z hlediska jazyka PostScript soubory několika postscriptových podprogramů. Ostatní typy fontů použitelné v PostScriptu jsou vytvořeny zcela nezávisle na jazyce PostScript a podléhají jiným specifikacím (např. Adobe Type 1 Font Format [9]) částečně či úplně nezávislým na jazyce PostScriptu.

V tabulce 3 si uvedeme dvě další položky, které mohou být obsaženy ve slovníku fontu Type 3. Jsou to procedury **BuildGlyph** a **BuildChar**, které slouží pro vlastní vykreslení písmene.

Položka	Typ	Význam hodnoty
FontType	celé číslo	Typ fontu. Je určující hodnotou pro strukturu fontu a pro reprezentaci jednotlivých písmen.
FontMatrix	pole	Matice přechodu od báze souřadného systému písmene k uživatelskému souřadnému systému.
FontName	řetězec	Jméno fontu, které není postscriptovým interpretem k ničemu použito. Má pouze informační hodnotu.
FontInfo	slovník	Slovník fontu obsahující další informace, ke kterým nepřistupuje postscriptový interpret přímo.
LanguageLevel	celé číslo	Minimální požadovaná verze postscriptového interpretu pro korektní chování fontu. Standardní hodnotou je 1.
WMode	celé číslo	Směr písma daného fontu. Tato hodnota určuje, který ze dvou směrů písma (horizontální, vertikální) bude použit. Definováno v PostScriptu Level 2. Standardní hodnota je 0 (horizontální písmo).
FID	fontID	Speciální objekt typu fontID. Jeho hodnota je definována po prvním použití operátoru **definefont**.

Tabulka 1: Položky slovníku fontu společné pro všechny typy fontů

Rozdíl mezi těmito procedurami je pouze v tom, že procedura **BuildChar** očekává na zásobníku kromě slovníku fontu ještě kód znaku, zatímco procedura **BuildGlyph** očekává kromě slovníku fontu jméno znaku. Procedura **BuildChar** tedy ještě musí uvnitř svého kódu zjistit, jak se jmenuje procedura vykreslující písmeno pro znak daného kódu. Nově vytvářené fonty ve formátu Type 3 by proto měly obsahovat definici procedury **BuildChar**, která zajišťuje plnou zpětnou kompatibilitu.

Informace, které jsme si výše popsali, si nyní demonstrujeme na velmi jednoduchém fontu. Tento font obsahuje pouze jedno jediné písmeno, které reprezentuje znak I.

```
8 dict begin
% Volitelná položka ve slovníku fontu
```

Položka	Typ	Význam hodnoty
Encoding	pole	Pole jmen znaků, které je použito při zobrazení znaku zadaného kódem.
FontBBox	pole	Pole čtyř čísel definující bounding box fontu.
UniqueID	celé číslo	Nezáporné celé číslo menší než $2^{24} - 1$ jednoznačně identifikující font.
XUID	pole	Pole celých čísel jednoznačně identifikující daný font včetně jeho varianty. Tato položka je definována v PostScript Level 2.

Tabulka 2: Položky slovníku fontu společné pro základní typy fontů

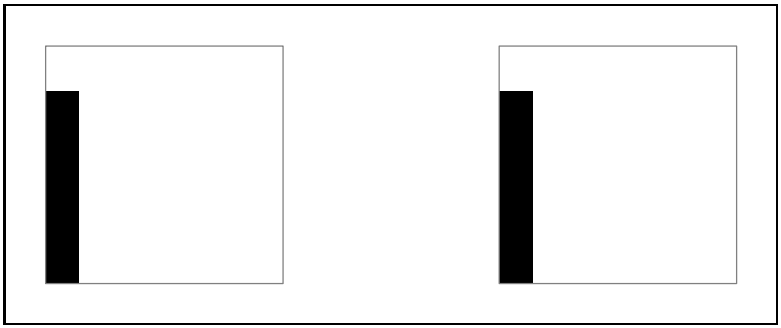
Položka	Typ	Význam hodnoty
BuildGlyph	procedura	Procedura, která vyrastruje písmeno pro požadovaný znak. Při volání této procedury již zásobník musí obsahovat slovník fontu a jméno znaku, který má procedura vyrastrovat. Tato procedura je volitelná a je definována až v PostScript Level 2.
BuildChar	procedura	Procedura, která vyrastruje písmeno pro požadovaný znak. Tato procedura musí být obsažena ve fontech určených pro PostScript Level 1 a pokud chybí i procedura **BuildGlyph** v PostScript Level 2 nebo 3, tak i zde. Při volání této procedury již zásobník musí obsahovat slovník fontu a kód znaku, který má procedura vyrastrovat.

Tabulka 3: Další položky slovníku fontu pro fonty Type 3

```
/FontName (MyTypeThreeFont) def
% Nutné položky ve slovníku fontu
/FontType 3 def
/FontMatrix [.001 0 0 .001 0 0] def
/FontBBox [0 0 750 750] def
% Definice kódovacího vektoru
/Encoding 256 array def
```

```
   0 1 255 {Encoding exch /.notdef put} for
   Encoding 73 /myonlychar put
   % Vedlejší slovník obsahující popisy jednotlivých písmen
   /CharProcs 2 dict def
      CharProcs begin
         /.notdef {} def
         % Definice vlastního znaku
         /myonlychar
         {
            % Definice cesty
            0 0 moveto
            100 0 lineto
            100 800 lineto
            0 800 lineto
            closepath
            % Vyplnění definované cesty
            fill
         } bind def
      end
   % Procedura BuildGlyph
   /BuildGlyph
   {
      % Nastavení výstupního a vyrovnávacího zařízení
      1000 0
      0 0 750 750
      setcachedevice
      % Vyzvednutí jména procedury ze slovníku CharProcs
      exch /CharProcs get exch
      2 copy known not
         { pop /.notdef }
         if
      % Provedení procedury
      get exec
   } bind def
   % Procedura BuildChar pro zpětnou kompatibilitu
   /BuildChar
   {
      1 index /Encoding get exch get
      1 index /BuildGlyph get exec
   } bind def
   currentdict
end
/MyTypeThreeFont exch definefont pop

% Nyní definovaný font použijeme
/MyTypeThreeFont findfont 100 scalefont setfont
0 0 moveto
(IAI) show
```

Pokud tento program v jazyce PostScript předložíme postscriptovému interpretu, dostaneme výstup podobný obrázku 1.



Obrázek 1: Výstup z námi definovaného fontu ve formátu Type 3

Mezi dvěma čtverci, které pouze zvýrazňují souřadný prostor písmen a nejsou výsledkem našeho programu, je volné místo o šířce 750 jednotek v souřadném systému písmene, protože jsme v příkladu použili znak A, jehož písmeno není v našem fontu definováno.

Formát Type 1

Jazyk PostScript tedy umožňuje definovat vlastní fonty pomocí formátu Type 3. Obrovskou nevýhodou tohoto formátu je nutnost implementace kompletního postscriptového interpretu. Navíc fonty ve formátu Type 3 mohou být velmi složité, jejich jednotlivé kontury se mohou vypočítávat případně mohou být i rekurzivně definovány, což je pro implementaci a větší rozšíření velmi nevýhodné. Společnost Adobe se proto pokusila navrhnout další formát, který by tyto nevýhody odstranil. A proto vznikl i dokument Adobe Type 1 Font Format [9], specifikace písem ve formátu Type 1. Formát Type 1 je mezinárodním standardem organizace ISO registrovaným pod číslem 9541 [5].

Formát Type 1 je stejně jako formát Type 3 založen na postscriptových příkazech, ale jejich syntaxe je velmi výrazně zjednodušena. Stejně tak je zúžena i množina použitelných příkazů. Postscriptový interpret pro rastrování fontů ve formátu Type 1 používá jiné algoritmy, které zajišťují kvalitnější výstup (zejména při extrémně malých velikostech písma). Další podstatnou výhodou fontů ve formátu Type 1 jsou i tzv. *hinty*, které charakterizují významné tahy či hranice jednotlivých písmen fontu. Jazyk PostScript žádnou podobnou vlastnost nenabízí a tudíž ji fonty ve formátu Type 3 nemohou využít. Fonty ve formátu Type 3 však mohou využít kompletní příkazovou sadu jazyka PostScript, zatímco formát Type 1 disponuje pouze omezenou sadou příkazů.

Formát Type 1 pracuje na podobném principu jako formát Type 3 pouze s tím rozdílem, že žádný font neobsahuje proceduru **BuildChar**. Tato procedura je totiž implementována přímo v postscriptovém interpretu. Pokud bychom se podívali např. na implementaci této procedury v postscriptovém interpretu Ghostscript, našli bychom podobný kód jako u fontů ve formátu Type 3:

```
% Here are the BuildChar and BuildGlyph implementation for Type 1 fonts.
% The names %Type1BuildChar and %Type1BuildGlyph are known \
  to the interpreter.
% The real work is done in an operator:
%       <font> <code||name> <name> <charstring> .type1execchar -

(%Type1BuildChar)  cvn {        % <font> <code> %Type1BuildChar -
  1 index /Encoding get 1 index get .type1build .type1execchar
} bind def
(%Type1BuildGlyph) cvn {        % <font> <name> %Type1BuildGlyph -
  dup .type1build .type1execchar
} bind def
```

Tento kód je obsažen v souboru `gs_type1.ps` v adresáři `lib` příslušné distribuce postscriptového interpretu Ghostscript. Procedura **BuildChar** u Type 1 fontů tedy očekává na zásobníku index do pole **Encoding**, které obsahuje seznam jmen procedur vykreslujících písmena pro jednotlivé znaky (indexy). Tato vlastnost se s výhodou dá využít k tzv. *překódování* fontu, kdy pouhou změnou pole **Encoding** dochází ke změně použité procedury pro rastrování znaku a tím i k rozdílnému výstupu bez nutnosti změny vlastního fontu (což je většinou znemožněno autorskými právy svázanými s daným fontem).

Jméno procedury, které **BuildChar** získá z pole **Encoding** je poté použito opět jako index do slovníku **CharStrings**, který je obsažen ve fontu. Slovník **CharStrings** obsahuje binárně zakódované instrukce definující vlastní proceduru, jež obsahuje definici jednotlivých kontur písmene. Tato procedura je poté provedena a jí definovaná kontura je buď vyplněna nebo pouze vykreslena podle toho, jakou hodnotu obsahuje položka **PaintType** ve slovníku fontu. Uživatel tedy může rozhodnout o tom, zda bude font a jeho písmena vyplněná či budou vykresleny pouze jejich obrysy pouhou změnou jedné položky ve slovníku fontu. To je jednak výhodou proti fontům ve formátu Type 3, protože tato úprava by znamenala přepsání definice samotného písmene a také je to nevýhodné pro tvůrce fontů Type 1, kteří musí brát v potaz obě varianty sazby a musí upravit definici kontur písmene.

Slovník Type 1 fontu

Stejně jako u Type 3 fontů jsou i Type 1 fonty pro postscriptový interpret reprezentovány slovníkem, který vznikne provedením samotného Type 1 fontu. Položky slovníku fontu společné všem typům fontů jsou uvedeny v tabulce 1.

Tabulka 4 obsahuje další položky, které mohou být součástí slovníku fontu ve formátu Type 1.

Slovník Type 1 fontu může stejně jako slovník kteréhokoli jiného typu fontu obsahovat slovník **FontInfo**, který například u fontu ITC Anna obsahuje následující informace:

```
/FontInfo 10 dict dup begin
/version (001.000) readonly def
/Notice (Copyright (c) 1993 Adobe Systems Incorporated. \
         All Rights Reserved. \
         ITC Anna is a~trademark of International Typeface Corporation.
        ) readonly def
/FullName (ITC Anna) readonly def
/FamilyName (ITC Anna) readonly def
/Weight (Regular) readonly def
/isFixedPitch false def
/ItalicAngle 0 def
/UnderlinePosition -100 def
/UnderlineThickness 50 def
end readonly def
```

První a poslední řádek výpisu jsou operátory jazyka PostScript, které uvozují (resp. ukončují) definici slovníku **FontInfo**. Jednotlivé řádky uvnitř definice slovníku definují verzi fontu (položka **version**), poznámku o autorských právech či ochranných známkách svázaných s fontem (položka **Notice**), plné jméno fontu (položka **FullName**), rodinu písma (položka **FamilyName**) a duktus (**Weight**). Položka **IsFixedPitch** slouží k identifikaci neproporcionálních písem, položka **ItalicAngle** udává sklon vertikálních tahů od vertikální osy. Zbylé položky (**UnderlinePosition** resp. **UnderlineThickness**) definují doporučené umístění resp. šířku podtrhovací linky. Podrobnější informace o jednotlivých položkách slovníku **FontInfo** jsou uvedeny v tabulce 5.

Jazyk PostScript nestanoví přesná pravidla, která by měly splňovat jednotlivé položky slovníku fontu, ale postupně se vytvořily jisté konvence, které jsou předpokládány některými aplikačními programy. Jméno fontu (položka **FontName**) je zkrácenou podobou plného jména fontu (**FullName**). Protože je hodnota této položky použita pro postscriptový operátor **definefont**, je zvykem vypustit mezery z plného jména fontu, případně změnit za znak minus. Např. font ITC Zapf Chancery Medium Italic má v položce **FontName** slovníku **FontInfo** hodnotu ZapfChancery-MediumItalic. Některými aplikacemi je rodina písma (položka **FontFamily**) použita pro hierarchické zařazení fontu. Proto jsou např. fonty Times Bold, Times Italic a Times Roman součástí větve Times v hierarchii fontů. Tato konvence je např. aplikována v programu Adobe Type Reunion [8]. Plné jméno fontu (položka **FullName**) většinou začíná jménem rodiny (**FontFamily**) a je následováno seznamem slov popisujícím kresbu písma. Jednotlivá slova jsou oddělena mezerami. Již zmíněný font ITC Zapf

Položka	Typ	Význam hodnoty
PaintType	celé číslo	Hodnotou této položky je dán typ kresby písmen. Pokud je hodnotou položky **PaintType** číslo 0, budou kontury písmen vyplněné. Pokud zde bude hodnota 2, budou pouze vykresleny kontury a nebudou vyplněny. Tato položka je povinná.
StrokeWidth	číslo	Šířka kontury. Tato hodnota je rovna šířce kontur, pokud nejsou vyplněny (tj. pokud je hodnota položky **PaintType** rovna 2). Pokud není definována, je její hodnota 0. Standardně distribuované fonty tuto položku neobsahují, je ji proto nutné při definici nového fontu, který bude bez vyplněných kontur, přidat do slovníku fontu s nenulovou hodnotou. Jedná se tedy o nepovinnou položku.
Metrics	slovník	Slovník **Metrics** obsahuje metrické informace pro horizontální směr písma. Tento slovník je nepovinný a v původních fontech bývá nedefinován. Používá se pouze při změně metrických informací písmen.
Metrics2	slovník	Jazyk PostScript Level 2 umožňuje specifikovat pomocí slovníku **Metrics2** metrické informace i pro vertikální směry písma.
CDevProc	procedura	Tato procedura souvisí s kompozitními fonty, nemusí být nutně definována a je ignorována postscriptovými interprety Level 1, které nepodporují kompozitní znaky.
CharStrings	slovník	Tento slovník přiřazuje jménům znaků z pole **Encoding** vlastní procedury pro vykreslení příslušných písmen.
Private	slovník	Slovník **Private** obsahuje informace, které by měly být k dispozici pouze samotnému fontu a nikoli uživateli.

Tabulka 4: Další položky slovníku fontu pro fonty Type 1

Položka	Typ	Význam hodnoty
version	řetězec	Hodnotou této položky je číslo verze Type 1 fontu.
Notice	řetězec	Pokud se k fontu vztahují ochranné známky či autorská práva, měla by být tato skutečnost zmíněna v položce **Notice**.
FullName	řetězec	Řetězec jednoznačně identifikující jméno fontu.
FamilyName	řetězec	Jméno rodiny fontů, do které font patří.
Weight	řetězec	Duktus fontu.
IsFixedPitch	boolean	Identifikátor neproporcionálních písem.
ItalicAngle	číslo	Úhel, který svírají dominantní vertikální tahy se svislou osou. Úhel je specifikován ve stupních.
UnderlinePosition	číslo	Doporučená vzdálenost podtrhovací linky od účaří.
UnderlineThickness	číslo	Doporučená šířka podtrhovací linky.

Tabulka 5: Položky slovníku **FontInfo**

Chancery Medium Italic je členem rodiny písem ITC Zapf Chancery. Slova Medium a Italic tedy specifikují kresbu písma a jeho řez.

Tato pravidla však nejsou pro tvůrce fontů závazná, společnost Adobe pouze doporučuje jejich dodržování a všechny její aplikace a fonty tato pravidla také dodržují. Tvůrci fontu také musí dodržovat jisté elementární předpoklady, aby jejich font mohly používat i aplikace, které neobsahují kompletní postscriptový interpret. Takovým softwarem je např. i Adobe Type Manager [9, Adobe Type Manager Compatibility] umožňující práci s postscriptovými fonty na operačních systémech společnosti Microsoft, které pro práci s fonty interně používají formát True Type.

Příklad Type 1 fontu

V tomto odstavci si uvedeme zdrojový kód fontu ITC Anna, který budeme v dalším textu používat pro demonstraci vlastností formátu Type 1.

```
%!PS-AdobeFont-1.0: Anna 001.000
%%CreationDate: Tue Jul 13 10:29:56 1993
%%VMusage: 20678 27570
%% ITC Anna is a trademark of International Typeface Corporation.
11 dict begin
```

```
/FontInfo 10 dict dup begin
/version (001.000) readonly def
/Notice (Copyright (c) 1993 Adobe Systems Incorporated. \
         All Rights Reserved. \
         ITC Anna is a~trademark of International Typeface Corporation.
        ) readonly def
/FullName (ITC Anna) readonly def
/FamilyName (ITC Anna) readonly def
/Weight (Regular) readonly def
/isFixedPitch false def
/ItalicAngle 0 def
/UnderlinePosition -100 def
/UnderlineThickness 50 def
end readonly def
/FontName /Anna def
/Encoding StandardEncoding def
/PaintType 0 def
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0] readonly def
/UniqueID 41116 def
/FontBBox{-81 -250 932 856}readonly def
currentdict end
currentfile eexec
cf0eeff329fe1db159a37891f19957002e7d435f06065e5e0a71393ede47
7128a76e9ca6da8df0c353e1352afd7fb01e92cbd565fc568222229162e2
... zkráceno
d54edd725c881f49e22678743289b3e9ef9f262271cab2be4a8872ca6c8b
9624a36451f78916e9db508a003da4b339ece06582ef04f84a
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
... zkráceno
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
cleartomark
```

Fonty ve formátu Type 1 jsou programy v jazyce PostScript, a proto by měly stejně jako ostatní programy v PostScriptu začínat dvojicí znaků `%!` . Některé operační systémy tuto dvojici znaků potřebují k identifikaci dokumentů ve formátu PostScript například pro určení vhodného filtru pro tisk. Zbytek řádku identifikuje formát fontu, jeho jméno a verzi:

```
%!PS-AdobeFont-1.0: Anna 001.000
```

Řetězec `PS-AdobeFont-1.0:` je používán fonty od společnosti Adobe. Některé fonty mohou používat i odlišnou specifikaci:

```
%!FontType1-1.0: Anna 001.000
```

V obou případech řetězec `1.0` identifikuje verzi specifikace formátu Adobe Type 1. V současné době je aktuální verzí specifikace verze 1.1 [9]. Řetězec `Anna` je jméno fontu a `001.000` je číslo verze fontu.

Řádek obsahující řetězec `CreationDate: Tue Jul 13 10:29:56 1993` udává čas, kdy byl font vytvořen.

```
%%VMusage: 20678 27570
```

Tento komentář je užitečný spíše pro aplikační programy než přímo pro postscriptový interpret. Udává totiž množství paměti, které font spotřebuje v postscriptovém interpretu. Aplikační program se proto ještě před jeho zavedením může ujistit, že postscriptový interpret disponuje dostatečným množstvím paměti. Druhé číslo (v našem případě 27570) vznikne rozdílem obsazení paměti po a před prvním zavedením fontu do paměti. První číslo vznikne odečtením obsazené paměti po a před druhým zavedením. První číslo je vždy menší nebo rovno druhému číslu, protože postscriptové interprety umožňují sdílení řetězců mezi fonty. Pro přibližné zjištění těchto čísel je možno použít i postscriptového interpretu Ghostscript. Následující postscriptový program zjistí paměťové nároky fontu:

```
vmstatus pop /number1 exch def pop
(Anna.pfa) run
vmstatus pop /number2 exch def pop
(Anna.pfa) run
vmstatus pop /number3 exch def pop

(%%VMusage is: ) =

number3 number2 sub =
number2 number1 sub =

quit
```

Příkladem jeho výstupu pro náš vzorový font ITC Anna je:

```
%%VMusage is:
23153
27073
```

Pokud bychom tedy font ITC Anna vytvářeli, uložili bychom do jeho definičního souboru řádek

```
%%VMusage: 23153 27073
```

Poslední komentářový řádek je pouze vyjádřením společnosti International Typeface Corporation, která vlastní ochrannou známku ITC Anna.

Následuje definice slovníku **FontInfo**, která je popsána výše. Poté je nutné definovat další položky slovníku fontu. Položka **FontName** obsahuje řetězec `/Anna`, proto bude možné font používat pod jménem `/Anna`.

Položka **Encoding** definuje kódovací vektor fontu. Místo vlastního pole je použito jedno z předdefinovaných polí. Kódovací vektor skrývající se pod jménem `StandardEncoding` je definován ve specifikaci jazyka PostScript

[6, příloha E.6]. Tato specifikace obsahuje definice dalších vektorů (`Expert`, `ISOLatin1Encoding`, `Symbol`). To ovšem neznamená, že by tvůrce fontu byl omezen jen těmito kódovacími vektory. Ani uživatel fontu není omezen těmito vektory a může si stejně jako tvůrce fontu definovat vlastní. Potom by definice kódovacího vektoru mohla vypadat například takto:

```
/Encoding 256 array
0 1 255 {1 index exch /.notdef put} for
dup 32 /space put
dup 33 /exclam put
dup 34 /universal put
dup 35 /numbersign put
dup 36 /existential put
dup 37 /percent put
... zkráceno
dup 250 /bracketrightex put
dup 251 /bracketrightbt put
dup 252 /bracerighttp put
dup 253 /bracerightmid put
dup 254 /bracerightbt put
readonly def
```

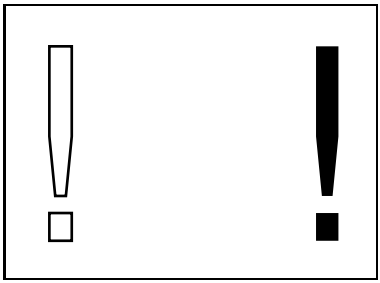
Tento kód v jazyce PostScript nejprve definuje pole **Encoding** o 256 prvcích, přiřadí každému prvku hodnotu `/.notdef` a teprve poté definuje význam jednotlivých položek v poli, kterým přiřazuje názvy procedur pro vykreslení jednotlivých písmen. Příklad je převzat z fontu Symbol. Pokud bude uživatel tohoto fontu chtít vykreslit znak s kódem 33, převezme procedura **BuildChar** z kódovacího vektoru název procedury `/exclam`, zavolá ji, a znak bude poté vykreslen, případně budou vykresleny pouze jeho kontury v závislosti na hodnotě položky **PaintType**.

```
/PaintType 0 def
```

Ta je v našem příkladu rovna 0, a proto bude znak vykřičník i vyplněn. Pokud by byla hodnota položky **PaintType** rovna 2, byl by vykřičník pouze vykreslen:

Další dvě položky jsou v téměř všech fontech ve formátu Type 1 stejné. Položka **FontType** musí obsahovat hodnotu 1 a hodnotou položky **FontMatrix** je matice přechodu od báze souřadného systému písmene k uživatelskému souřadnému systému. Jinými slovy je tak definován souřadný systém, ve kterém jsou prováděny procedury vykreslující jednotlivá písmena. Většina fontů ve formátu Type 1 používá souřadný systém o 1000 jednotkách, stejně jako náš vzorový font ITC Anna.

Položka **FontBBox** definuje obdélník, který vznikne opsáním útvaru vzniklého po vyrastrování všech znaků ve společném počátku. Tento údaj je využíván zejména postscriptovým interpretem k efektivnější práci s vyrovnávací pamětí a také pro ořezávání.



Obrázek 2: Vliv položky **PaintType** na vzhled písmen

Poslední čitelnou položkou fontu ITC Anna je **UniqueID**. Tato položka slouží k uchovávání vyrovnávací paměti s vyrastrovanými písmeny mezi jednotlivými úlohami postscriptového interpretu. Pokud font tuto položku neobsahuje, je tato paměť po skončení aktuální úlohy vymazána. Pokud však je položka **UniqueID** ve slovníku fontu obsažena, je vyrovnávací paměť uchována a při příštím použití tohoto fontu (resp. fontu se stejnou hodnotou **UniqueID**) již není proces volání jednotlivých procedur prováděn, vše je zjednodušeno vzhledem k obsahu vyrovnávací paměti. Tento proces však předpokládá jednoznačné přiřazení hodnot položky **UniqueID** jednotlivým fontům. Společnost Adobe proto drží databázi již přiřazených hodnot a tvůrci nového fontu ji mohou požádat o přidělení nového čísla. Přesný postup pro podání žádosti je uveden ve specifikaci formátu Type 1 [9, kapitola 2.5]. Hodnotou položky může být celé číslo menší než 16 777 215 ($2^{24} - 1$).

Zbylá část fontu je zašifrována, aby nebylo možné jednoduše „přečíst“ procedury vykreslující jednotlivá písmena. Metoda použitá pro šifrování je ale přehledně popsána ve specifikaci formátu Type 1 [9, kapitola 7]. Ostatně existuje i několik Open Source implementací této metody (např. knihovna t1lib, případně balík utilit t1utils).

Šifrovaná část obsahuje dvě zbylé položky slovníku fontu. Jsou to slovníky **Private** a **CharStrings**.

Slovník Private

Slovník **Private** o bsahuje informace, které se týkají každého písmene fontu. Jsou v něm obsaženy procedury, které jsou sdíleny podprogramy pro kresbu jednotlivých znaků, jsou zde uvedeny také hintovací informace pro celý font apod. Slovník **Private** je obsažen v zašifrované části definičního souboru fontu. Do čitelné podoby je možné jej převést např. programem `t1disasm` z balíku t1utils. Slovník **Private** našeho vzorového fontu obsahuje následující informace:

```
dup /Private 16 dict dup begin
/-||{string currentfile exch readstring pop}executeonly def
/||-{noaccess def}executeonly def
/||{noaccess put}executeonly def
/BlueValues [ -25 0 704 722 436 436 728 741 ] ||-
/OtherBlues [ 281 281 -49 -49 ] ||-
/MinFeature{16 16} ||-
/StdHW [77] ||-
/StdVW [77] ||-
/ForceBold false def
/password 5839 def
/UniqueID 41116 def
/Subrs 170 array
dup 0 {
   3 0 callothersubr
   pop
   pop
   setcurrentpoint
   return
   } ||
... zkráceno
dup 169 {
   169 68 96 138 0 171 rrcurveto
   0 171 -96 137 -169 68 rrcurveto
   return
   } ||
||-
```

Slovník **Private** obsahuje definici tří procedur, které budou ve vlastním slovníku často používány. Další položky slovníku **Private** již definují informace podstatné pro postscriptový interpret resp. pro funkci **BuildChar**. Hodnotou pole **BlueValues** je několik dvojic, které definují tzv. *písmovou osnovu* [1, strana 7]. Jsou to oblasti, do kterých zasahují jednotlivé tahy písmen. První číslo z každé dvojice udává spodní a druhé číslo udává horní hranici každé oblasti. První dvojice (pokud je přítomna) vždy udává oblast pod účařím (tj. první číslo je y-ová souřadnice tzv. *dolního přetahu* a druhé číslo je y-ová souřadnice *účaří*). Druhá dvojice v našem příkladu (704, 722) udává *verzálkovou dotažnici*. Třetí dvojice udává *střední dotažnici*. Poslední dvojice z našeho příkladu definuje *horní dotažnici*. Podobný význam mají hodnoty v položce **OtherBlues**.

Položky **StdHW** a **StdVW** definují standardní šířku horizontálních a vertikálních tahů. Tyto informace používá postscriptový interpret zejména při malých velikostech písma, kdy hraje roli každý bod. Postscriptový interpret pomocí těchto hodnot zajišťuje konzistentní vzhled písmen. Při větších velikostech písma nejsou rozdíly několika málo bodů viditelné a proto tyto hodnoty nejsou využity. Podobný význam mají i položky **StemSnapH** a **StemSnapV**, které ale nejsou ve vzorovém fontu použity.

Položka **ForceBold** má hodnotu `false`. Tato položka bude použita, pokud budou písmena fontu vykreslována v malých velikostech. V těchto velikostech se totiž ztrácejí rozdíly mezi normálním a tučným písmem. Právě proto je možno použít položku **ForceBold**, která informuje postscriptový interpret o tom, jak se má chovat k tučným písmenům. Pokud je hodnotou položky **ForceBold** řetězec `true`, postscriptový interpret rozšíří všechny tahy tučných písmen.

Položky **MinFeature** a **password** musí být ve slovníku **Private** přítomny, ale jejich význam je bohužel nedokumentován.

Přesný význam vysvětlených i dalších položek ve slovníku **Private** je definován ve specifikaci formátu Type 1 [9]. Bohužel ne všechny položky jsou popsány přesně a dostatečně, a proto je velmi obtížné se hlavně ve složitých fontech ve formátu Type 1 orientovat.

Slovník **Private** dále obsahuje i pole **Subrs**, které obsahuje často se opakující procedury. Tyto procedury mohou například vykreslovat akcenty, často se opakující vertikální či horizontální tahy, definovat hinty apod. V našem fontu pole **Subrs** obsahuje 170 procedur.

Slovník CharStrings

Slovník **CharStrings** obsahuje vlastní procedury pro vykreslování jednotlivých písmen a definice hintovací informace pro jednotlivá písmena. Procedury jsou indexovány svým jménem. Pokud tedy chceme vykreslit nějaké písmeno, musíme znát jméno znaku, který reprezentuje. Pokud známe kód znaku, můžeme jméno příslušné procedury (a tedy index do pole **CharStrings**) získat z pole **Encoding**. Definice slovníku **CharStrings** v našem fontu ITC Anna obsahuje následující text:

```
% Definice slovníku CharStrings
2 index /CharStrings 229 dict dup begin
% Písmeno mezera
/space {
   0 185 hsbw
   endchar
   } |-
% Písmeno vykřičník
/exclam {
   54 185 hsbw
   0 77 vstem
   0 77 hstem
   710 -20 hstem
   77 hmoveto
   137 callsubr
   77 148 rmoveto
   562 vlineto
   -77 hlineto
   -562 vlineto
```

```
   closepath
   endchar
   } |-
... zkráceno
/ecircumflex {
   -25 333 hsbw
   148 callsubr
   127 574 rmoveto
   86 callsubr
   } |-
% Nedefinovaný znak
/.notdef {
   0 185 hsbw
   endchar
   } |-
% Konec definice slovníku
end
```

Tento text je samozřejmě ve fontu kódován jednoduchým algoritmem [9, strana 47] a poté zašifrován pomocí metody eexec [9, strana 64]. Důvodem pro kódování je hlavně ušetření místa, protože jednotlivé příkazy procedur jsou kódovány do jednoho či maximálně dvou bajtů, zatímco jejich jména jsou mnohem delší. Důvody pro šifrování jsou spíše „politické“ či komerční. Společnost Adobe si nepřála, aby mohl kdokoli jednoduše kopírovat její písma, ale tato ochrana ani neklade příliš velké překážky. Existují dokonce volně šiřitelné implementace rozkrývající tuto jednoduchou ochranu.

Vlastní procedury ve slovníku **CharStrings** mohou obsahovat instrukce pěti typů:

- příkazy začínající nebo končící kontury písmen
- příkazy, které sestavují jednotlivé kontury
- příkazy pro práci s podprocedurami
- příkazy definující hintovací informace pro jednotlivé znaky
- aritmetické příkazy

Každá procedura ve slovníku **CharStrings** má následující strukturu:

```
% Definice procedury
% Jméno procedury
/jméno {
   % Left sidebearing point
   x y hsbw
   ...
   % Konec definice znaku
   endchar
   } |-
```

Tato sekvence příkazů definuje proceduru **jméno**. Každá procedura ve slovníku fontu začíná příkazem **hsbw** (případně jeho variantou **sbw**). Příkaz **hsbw**

očekává na zásobníku dvě hodnoty (x a y). Hodnota x udává x-ovou souřadnici levého okraje bounding boxu písmene (tzv. *left sidebearing point*). Druhá hodnota (y) udává šířku znaku. Tento příkaz také nastaví aktuální bod na souřadnice $(x, 0)$. K aktuálnímu bodu se vztahuje většina příkazů, které vytvářejí kontury písmene. Aktuální bod ale není součástí definice vlastní kontury, a proto je možné použít dále zmíněné příkazy pro nastavení počátečního bodu kontury.

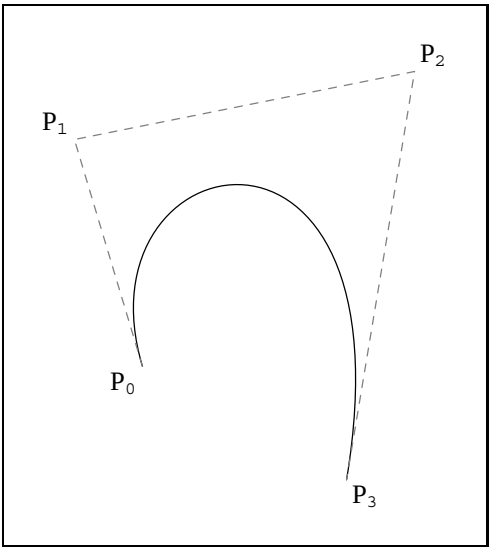
Každá procedura ve slovníku **CharStrings** musí končit příkazem **endchar**, který ukončuje definici kontur písmene a předává řízení proceduře **BuildChar**. Tato procedura volá speciálně upravené verze postscriptových procedur **stroke** a **fill**, které definované kontury vykreslí (v případě, že hodnotou položky **PaintType** ve slovníku fontu je 2) nebo vyplní (pokud je hodnota položky **PaintType** rovna 0).

Specifikace formátu Type 1 [9, strana 52] definuje tři příkazy pro změnu aktuálního bodu. Jsou to **rmoveto**, **hmoveto** a **vmoveto**. Příkaz **rmoveto** očekává na zásobníku dvě čísla (dx a dy) a posouvá aktuální bod o vektor (dx, dy). Pokud je tedy aktuální bod před voláním tohoto příkazu (x, y), bude po provedení tohoto příkazu aktuální bod umístěn na souřadnicích $(x + dx, y + dy)$. Tento příkaz tedy funguje naprosto stejně jako stejnojmenný příkaz jazyka PostScript [6, strana 483]. Formát Type 1 také definuje dvě jednoduché modifikace příkazu **rmoveto**. Příkazy **hmoveto** resp. **vmoveto** očekávají na zásobníku pouze jedno číslo (dx resp. dy) a jsou ekvivalentní příkazům `dx 0` **rmoveto** resp. `0 dy` **rmoveto**. Slouží tedy k horizontálním či vertikálním posunům aktuálního bodu.

Další skupina příkazů slouží k doplnění úsečky do kontury písmene. Stejně jako u příkazů pro změnu aktuálního bodu existuje jeden obecný příkaz a z něj odvozené speciální příkazy, existuje příkaz **rlineto**, který očekává na zásobníku dva argumenty (dx a dy). Pokud je aktuální bod před provedením tohoto příkazu umístěn na souřadnicích (x, y), bude do aktuální cesty (kontury) přidána úsečka s krajními body (x, y) a $(x + dx, y + dy)$. Příkaz tedy funguje stejně jako příkaz **rlineto** jazyka PostScript [6, strana 482]. Aktuálním bodem po provedení tohoto příkazu bude koncový bod přidané úsečky. Existují i speciální příkazy **hlineto** a **vlineto**, které očekávají na zásobníku pouze jeden argument a přidávají do cesty horizontální resp. vertikální úsečku. Příkaz `dx` **hlineto** je tedy ekvivalentní příkazu `dx 0` **rlineto** a `dy` **vlineto** má stejný význam jako `0 dy` **rlineto**.

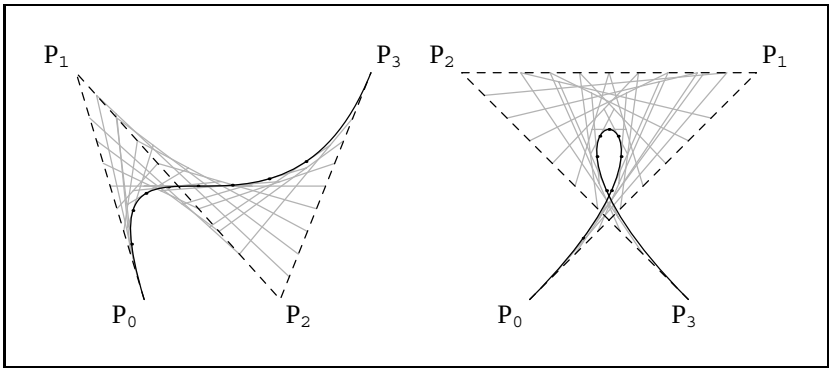
Poslední skupinu příkazů pro modifikaci kontur tvoří příkazy, které přidávají do aktuální cesty tzv. *Bézierovy křivky*. Tyto aproximační křivky nezávisle na sobě v letech 1952–62 používali P. de Casteljau a P. E. Béziere [13, strana 178]. Bézierovy křivky jsou zadány svým počátečním (P_0) a koncovým bodem (P_3) a dvojicí *kontrolních bodů* (P_1, P_2), které ovlivňují výsledný tvar křivky.

Na obrázku 3 je znázorněn „přirozený“ tvar Bézierovy křivky včetně kontrolních bodů a tzv. *kontrolního polygonu*. Pokud budou kontrolní body umístěny v rovině jiným způsobem, bude také tvar křivky odlišný (viz obrázek 4). Bézierova křivka se dokonce může sama protínat. Speciální volbou řídících bodů je



Obrázek 3: Bézierova křivka

možné dosáhnout i toho, že Bézierova křivka se transformuje v úsečku (pokud budou všechny body kolineární) případně i v bod (pokud všechny body splynou). Využívat těchto vlastností ve fontech Type 1 je zbytečné, neboť pro vykreslení úsečky existuje optimalizovaná funkce **rlineto** a vykreslení bodu nemá žádný význam.



Obrázek 4: Bézierovy křivky s jinými kontrolními body

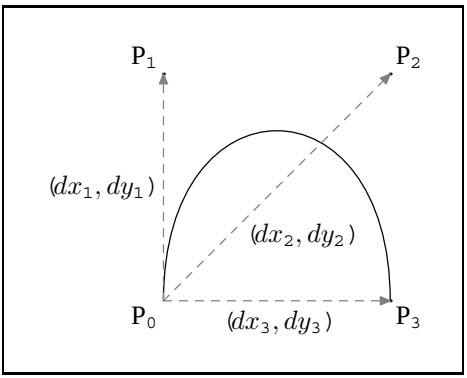
Matematický popis Bézierových křivek je dán následujícími vztahy:

$$P(t) = P_0 B_0^3(t) + P_1 B_1^3(t) + P_2 B_2^3(t) + P_3 B_3^3(t)$$

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

kde P_0, P_1, P_2 a P_3 jsou jednotlivé zadané body, $t \in [0,1]$; B_0^3, B_1^3, B_2^3, B_3^3 jsou tzv. *Bernsteinovy kubické polynomy* (Bernsteinovy polynomy třetího stupně), které jsou definovány výše uvedeným obecným vztahem (v našem případě se jedná o polynomy třetího stupně, tj. $n = 3$).

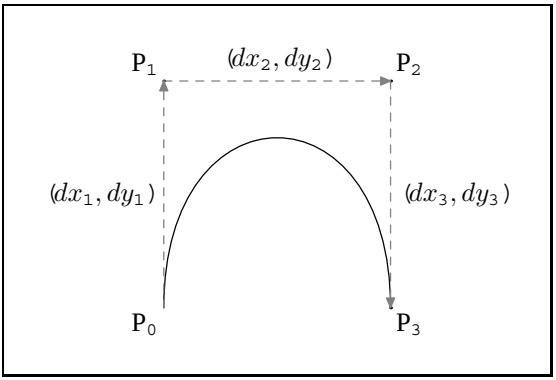
Bézierovy křivky jsou v jazyce PostScript implementovány příkazem **curveto** [6, strana 393], který do aktuální cesty vloží Bézierovu křivku. Tento příkaz má pouze šest parametrů (x-ové a y-ové souřadnice bodů P_1, P_2, P_3). Počáteční bod (P_0) je dán aktuálním bodem. Relativní obdoba tohoto příkazu (příkaz **rcurveto**) očekává na zásobníku opět šestici parametrů. První dvojice (dx_1, dy_1) udává relativní umístění druhého bodu od prvního. Druhá dvojice (dx_2, dy_2) udává umístění třetího bodu vůči prvnímu a poslední dvojice (dx_3, dy_3) reprezentuje vektor z počátečního do koncového bodu. Význam jednotlivých parametrů příkazu **rcurveto** je znázorněn na obrázku 5.



Obrázek 5: Argumenty postscriptového příkazu **rcurveto**

Formát Type 1 používá mírnou modifikaci tohoto příkazu, která očekává také šest argumentů: `dx1 dy1 dx2 dy2 dx3 dy3` **rrcurveto**. Tento příkaz přidá do aktuální cesty Bézierovu křivku, jejíž počáteční bod bude shodný s aktuálním bodem. První kontrolní bod křivky bude posunut o vektor (dx_1, dy_1) stejně jako u postscriptového příkazu **rcurveto**, ale druhý kontrolní bod bude posunut o vektor (dx_2, dy_2) od prvního kontrolního bodu (narozdíl od příkazu **rcurveto**

jazyka PostScript, kdy je druhý kontrolní bod posunut o daný vektor od počátku křivky). Koncový bod křivky bude posunut o vektor (dx_3, dy_3) od druhého kontrolního bodu. Grafická reprezentace jednotlivých parametrů je zřejmá z obrázku 6.



Obrázek 6: Argumenty příkazu **rrcurveto** ve formátu Type 1

Stejně jako u příkazů pro přidání úsečky do aktuální kontury existují i zjednodušené verze příkazu **rlineto**, je možné ve formátu Type 1 použít zjednodušenou podobu příkazů pro Bézierovy křivky. Příkaz `dx1 dx2 dy2 dy3` **hvcurveto** je ekvivalentní příkazu `dx1 0 dx2 dy2 0 dy3` **rrcurveto** a příkaz `dy1 dx2 dy2 dx3` **vhcurveto** je ekvivalentní příkazu `0 dy1 dx2 dy2 dx3 0` **rrcurveto**. Obě zjednodušené verze eliminují dva argumenty, protože předpokládají horizontální či vertikální směr tečných vektorů v krajních bodech křivky.

Posledním příkazem ovlivňujícím kontury písmene je příkaz **closepath**, který uzavírá aktuální konturu. Každá kontura by měla být uzavřena pomocí tohoto příkazu. Pokud by tomu tak nebylo, mohlo by dojít k nepříjemným efektům špatného vyplnění kontury písmene či přesahů při vykreslování kontur. Jazyk PostScript má podobný příkaz [6, strana 375], pouze s tím rozdílem, že ve formátu Type 1 zůstává aktuální bod nezměněn, zatímco u postscriptové varianty je aktuální bod přesunut na konec cesty a dojde tedy k jeho posunu.

Třetí skupinu příkazů formátu Type 1 tvoří příkazy pro práci s procedurami. Stejně jako u jiných programovacích jazyků slouží procedury pro častěji se opakující posloupnosti příkazů. Specifikace formátu Type 1 doporučuje používat procedury např. pro specifikace hintovacích informací (viz dále) nebo pro sekvence vytvářející častěji se opakující části kontur jednotlivých písmen. Procedury jsou uloženy v poli **Subrs** ve slovníku **Private**. Každá procedura v tomto poli má následující strukturu:

```
% Definice procedury v poli Subrs
% Procudera bude uložena pod indexem číslo_procedury
dup číslo_procedury {

   % Tělo procedury
   seznam příkazů...

   % Ukončení procedury a návrat do volajícího kódu
   return
}
```

Příkaz jazyka PostScript **dup** [6, strana 404] zduplikuje v poli **Subrs** odkaz proceduru se zadaným pořadovým číslem. Příkaz **return** vrací řízení zpět a procedura je tím ukončena. Definované procedury je možné volat pomocí dalších příkazů formátu Type 1. Mezi tyto příkazy patří **callsubr** a **callothersubr**. První z těchto příkazů slouží k volání procedur definovaných v poli **Subrs**, druhý k volání procedur z pole **OtherSubrs**. Některá písmena mohou být definována pouze pomocí volání procedur. Příkladem takové definice je například velké písmeno I s čárkou z fontu ITC Anna:

```
% Definice procedury /Iacute
/Iacute {
   % Left sidebearing point
   2 185 hsbw
   % Vertikální dřík
   153 callsubr
   % Umístění akcentu
   130 84 rmoveto
   % Vlastní akcent
   95 callsubr
} |-
```

Obě kontury písmene jsou definovány pomocí procedur a je tudíž velmi jednoduché tyto části (svislý dřík znaku I i akcent) použít k definici jiného písmene.

Při podrobnějším studiu Type 1 fontů se můžeme setkat i s příkazy **pop** a **setcurrentpoint**, které se používají v souvislosti s voláním procedur v poli **OtherSubrs**.

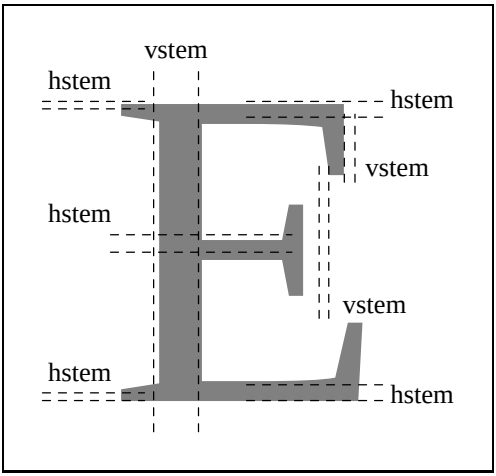
Formát Type 1 dále podporuje jediný příkaz sloužící k aritmetickým operacím. Příkaz **div** funguje stejně jako stejnojmenný příkaz jazyka PostScript [6, strana 404]. Na zásobníku očekává dva operandy, první z nich vydělí druhým a výsledek uloží na zásobník. Tento příkaz není ve většině fontů použit, neboť je poměrně pomalý a zejména na pomalejších počítačích by mohlo jeho použití výrazně zpomalit vykreslování.

Hintovací informace

Tvůrci formátu Type 1 při návrhu formátu vzali v úvahu i možnost jeho použití při nízkých rozlišeních, kdy často závisí na každém bodu. I proto umožňuje

formát Type 1 specifikovat tzv. *hintovací informace*, které upravují chování interpretu Type 1 fontů ve speciálních situacích. Každý *hint* (horizontální i vertikální) je deklarován dvěma souřadnicemi. Význam jednotlivých hintů je patrný z obrázku 7.

Definice písmene E z fontu Times Roman obsahuje celkem osm hintů (pět horizontálních a tři vertikální). S hintovacími informacemi pracuje přímo procedura **BuildChar** vestavěná v postscriptovém interpretu.

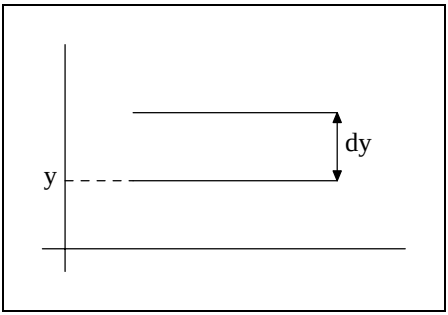


Obrázek 7: Hintovací informace písmene E z fontu Times Roman

Formát Type 1 obsahuje několik příkazů, které specifikují hinty. Horizontální hinty je možné specifikovat pomocí příkazů **hstem** a **hstem3**. Oba tyto příkazy používají pro specifikaci hintu dvou argumentů. První z nich je y-ová souřadnice spodní části hintu a druhý argument je výška hintu (viz obrázek 8).

Příkaz **hstem** očekává dva argumenty a jeho syntaxe je tedy `y dy` **hstem**. Příkaz **hstem3** umožňuje najednou specifikovat tři hinty a tudíž očekává šest argumentů zadaných podobně jako u příkazu **hstem**: `y0 dy0 y1 dy1 y2 dy2` **hstem3**. Podmínkou použití příkazu **hstem3** je stejná šířka všech hintů a umístění prostředního hintu přesně uprostřed mezi oběma krajními hinty. Tento příkaz je tedy možné použít např. pro specifikaci hintů symbolu matematické ekvivalence ($\equiv$).

Vertikální hinty se specifikují obdobně. Příkaz **vstem** očekává dva argumenty: `x dx` **vstem** a definuje vertikální hint s x-ovou souřadnicí mezi x a $x+dx$. Příkaz **vstem3** má podobnou syntaxi jako **hstem3** a používá se zejména pro specifikaci hintů písmene m a dalších se třemi vertikálními tahy.



Obrázek 8: Horizontální hinty

Formát Type 1 obsahuje ještě jeden příkaz, který informuje postscriptový interpret o specifické vlastnosti právě vytvářené kontury. Tento příkaz se jmenuje **dotsection** a používá se v páru. Jeho význam bude zřejmý z výpisu části definice písmene j:

```
   dotsection
   21 35 17 37 hvcurveto
   31 -26 26 -30 vhcurveto
   -30 -25 -25 -32 hvcurveto
   -38 34 -16 21 vhcurveto
   closepath
dotsection
```

Příkaz **dotsection** zde ohraničuje definici tečky, která je použita např. v písmenech reprezentujících znaky j nebo vykřičník. Tento příkaz je vhodný spíše pro použití se staršími interprety, které špatně interpretovaly samostatné kontury (obrysy tečky) a někdy došlo i k jejich slití s hlavní konturou písmene, což je nepřípustné.

Příklad Type 1 fontu

Stejně jako u formátu Type 3 jsme si definovali vlastní font s jedním znakem, vytvoříme nyní font ve formátu Type 1. Náš font bude obsahovat pouze jediné písmeno, které reprezentuje znak I. Samozřejmě nebudeme vytvářet přímo podobu vhodnou pro začlenění do postscriptového souboru, ale využijeme balíku t1utils, který zjednodušuje práci s jednotlivými podobami Type 1 fontů.

```
%!PS-AdobeFont-1.0: MyTypeOne 001.000
%%CreationDate: Sun Feb 13 15:32:57 2000
%%VMusage: 4031 7971
11 dict begin
```

```
/FontInfo 7 dict dup begin
/version (001.000) readonly def
/Notice (This is an example of Type 1 font.) readonly def
/FullName (My Type One) readonly def
/FamilyName (My Type One) readonly def
/Weight (Roman) readonly def
/isFixedPitch false def
/ItalicAngle 0 def
end readonly def
/FontName /MyTypeOne def
/Encoding StandardEncoding def
/PaintType 0 def
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0] readonly def
/FontBBox{0 0 800 800}readonly def
currentdict end
currentfile eexec
dup /Private 11 dict dup begin
/-||{string currentfile exch readstring pop}executeonly def
/Subrs 0 array def
2 index /CharStrings 2 dict dup begin
/I {
   0 800 hsbw
   100 0 rlineto
   0 800 rlineto
   -100 0 rlineto
   endchar
} def
/.notdef {
   0 800 hsbw
   endchar
} def
end
end
readonly put
noaccess put
dup/FontName get exch definefont pop
mark currentfile closefile
```

Výše uvedený text uložíme do souboru se jménem `MyTypeOne.disas` a převedeme jej do podoby PFB souboru (viz dále) pomocí programů `t1asm` a `t1binary` z balíku t1utils:

```
t1asm MyTypeOne.disas >MyTypeOne.pfa
t1binary MyTypeOne.pfa >MyTypeOne.pfb
```

Font je nyní připraven k použití postscriptovým interpretem. Tento fakt si můžeme ověřit prostým zavedením fontu do interpretu a jeho použitím. Na podporovaných operačních systémech (např. Linux) je možné využít např. interpretu Ghostscript:

```
(MyTypeOne.pfa) run
/MyTypeOne findfont 100 scalefont setfont
0 0 moveto
(IAI) show
```

Font je tedy možné využít v postscriptovém interpretu, ale některé další programy jej využít nemohou, protože nemáme k dispozici metrické informace. Vytvoříme tedy ještě soubor s těmito informacemi a ověříme jejich správnost např. pomocí programu pdfTEX. Metrické informace pro postscriptové fonty jsou většinou distribuovány v podobě AFM souboru.

```
StartFontMetrics 2.0
FontName MyTypeOne
EncodingScheme AdobeStandardEncoding
FullName My Type One
FamilyName My Type One
Version 001.000
FontBBox 0 0 800 800
StartCharMetrics 1
C 73 ; WX 800 ; N myonechar ; B 0 0 800 800 ;
EndFontMetrics
```

AFM soubor obsahuje minimum informací o fontu samotném (jeho jméno, číslo verze a bounding box). Jsou zde také uvedeny metrické informace našeho písmene I. Sázecí systém TEX potřebuje k použití tohoto fontu metrické informace ve své nativní podobě (v souboru ve formátu TFM) a tudíž jej budeme muset z AFM souboru vytvořit. Použijeme k tomu program `afm2tfm`, který je standardní součástí kterékoli distribuce TEXu:

```
afm2tfm MyTypeOne.afm
```

Abychom mohli použít náš font v systému pdfTEX, musíme jej ještě informovat o tom, kde má hledat vlastní font. pdfTEX totiž na rozdíl od TEXu potřebuje vědět, kde se font v našem souborovém systému nalézá, neboť jej musí vložit do výsledného souboru ve formátu PDF [12]. TEX do výsledného souboru ve formátu DVI vkládá pouze odkaz na font a nikoli font samotný. pdfTEX proto používá soubor `pdftex.map` [4], který naplníme následujícím řádkem:

```
MyTypeOne MyTypeOne <<MyTypeOne.pfb
```

Nyní je náš font připraven pro použití v systému pdfTEX.

Formáty souborů fontů Type 1

Výše popsaný formát souborů je možné poslat přímo postscriptové tiskárně, jejíž postscriptový RIP dokáže tento sedmibitový proud dat interpretovat [7, Font File Formats]. Totéž umí i softwarové interprety jazyka PostScript. Fonty ve

formátu Type 1 jsou v tomto formátu ukládány na discích počítačů v souborech s příponou `.pfa` a jsou přímo připraveny pro zavedení do postscriptového interpretu.

Tento formát však není vhodný pro ukládání souborů na disku, protože je poměrně náročný na diskový prostor. Hlavička fontu s definicí slovníků je v čitelné podobě a zakódovaná část je reprezentována řetězcem hexadecimálních číslic:

```
currentfile eexec
CF0EEFF329FE1DB159A37891F19957002E7D435F06065E5E0A71393EDE477128
A76E9CA6DA8DF0C353E1352AFD7FB01E92CBD565FC568222229162E248445364
52FEC4350886980508A70995587C8A9C47468FF87BA0F9AFD847390FD5215BAB
... zkráceno
9E5236D615703C5E421C4B129A95EA38D54EDD725C881F49E22678743289B3E9
EF9F262271CAB2BE4A8872CA6C8B9624A36451F78916E9DB508A003DA4B339EC
E06582EF04F84A
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
... zkráceno
0000000000000000000000000000000000000000000000000000000000000000
cleartomark
```

Je zřejmé, že uložení dat v tomto formátu je velmi neefektivní. Proto společnost Adobe vytvořila i binární formát určený pro reprezentaci fontů ve formátu Type 1 a právě tento formát je použit pro ukládání i distribuci fontů pro platformu PC. Příslušné soubory potom mají příponu `.pfb`.

Pro zavedení do postscriptového interpretu je nutné tento formát konvertovat zpět do textové podoby. Tato činnost je u většiny aplikací s postscriptovým výstupem automatizována a nemusíme jí věnovat pozornost. Pokud bychom vytvářeli soubor v PostScriptu ručně, je možné použít utilitu `pfb2pfa` z TEXové distribuce teTEX, program `t1ascii` z balíku t1utils nebo skript `pfbtopfa`, který je součástí postscriptového interpretu Ghostscript.

Formát souborů `.pfb` je velmi jednoduše navržen. Celý soubor je rozdělen do několika segmentů, které obsahují záhlaví a vlastní tělo. Segment může být jeden ze tří typů:

- text složený ze znaků ASCII tabulky (typ `0x01`),
- binární data (typ `0x02`),
- indikátor konce souboru (typ `0x03`).

První typ segmentu je použit pro hlavičku formátu Type 1, která je také v čitelné podobě. Segmenty prvního typu je tedy možné při překódování do formátu `.pfa` pro zavedení do postscriptového interpretu jenom překopírovat bez jakéhokoli zásahu (kromě konvertování znaků konce řádku). Binární data (druhý typ segmentu) by při převodu do formátu `.pfa` měla být konvertována do hexadecimální podoby. Segment posledního typu musí být posledním segmentem v souboru a měl by být v souboru obsažen právě jednou.

Hlavička každého segmentu obsahuje nejméně dva bajty. První z nich zahajuje segment a má vždy hodnotu `0x80` (127 v desítkové soustavě). Druhý bajt obsahuje typ segmentu (viz výše). Segment indikující konec souboru tedy obsahuje bajty `0x80 0x03`. Bajt `0x03` by měl být posledním bajtem souboru.

Datové segmenty mají hlavičku rozšířenu o čtyři bajty specifikující délku datové sekce sektoru, přičemž bajty jsou uspořádány podle rostoucí významnosti, první bajt je tedy nejméně významný. Ihned za těmito bajty následuje datová sekce sektoru.

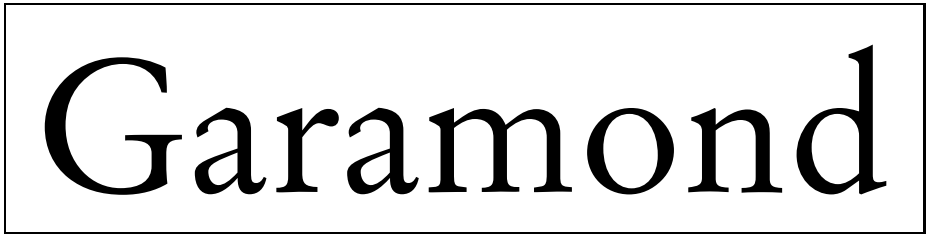
Výše uvedený text by měl po převodu do binární podoby (`.pfb`) obsahovat následující sekvence bajtů:

```
0x80 0x01 0xxx 0x00 0x00 0x00 ... následuje 0xxx znaků hlavičky fontu
0x80 0x02 0xxx 0x00 0x00 0x00 ... následuje 0xxx binárních bajtů

0xCF 0x0E 0xEF 0xF3 0x29 0xFE ... zkráceno

0x80 0x01 0x14 0x02 0x00 0x00 ... následuje 0x0214 znaků '0'
0x80 0x03
```

Soubor ve formátu `.pfb` může obsahovat libovolný počet datových segmentů, které mohou být libovolně kombinovány.

Písma ve formátu Multiple Master

V dobách minulých, kdy ještě nebyly k dispozici počítače a o fontech se tedy mohlo typografům a sazečům pouze zdát, vytvářely písmo umělci (*písmolijci*). Příkladem může být písmo Garamond původně navržené Claudem Garamondem v letech 1530–1540 [3].



Obrázek 9: Font Garamond

Ačkoli původní autor navrhl pouze několik velikostí a variací písma (základní písmo, italika, kapitálky), máme nyní k dispozici i tučné písmo a můžeme si vytvořit písmo zúžené apod. Technologie vektorových písem navíc umožňují

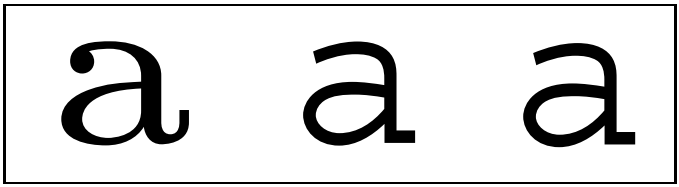
změnit velikost písma bez toho, abychom museli měnit vlastní řez písma. Máme tedy obrovskou výhodu před sazeči minulých dob, kteří museli požádat písmolijce o vytvoření další velikosti písma. Totéž platí i pro písma různých duktů. Dávní písmolijci téměř nevytvářeli písma tučná a nyní máme u většiny písem k dispozici celou škálu od velmi jemného písma přes polotučné až k tučným řezům. Na obrázku 10 je znázorněno písmo Humanist521BT v různých duktech.

Obrázek 10: Písmo různých duktů

Nevýhodou dnešního přístupu k písmům je stejný vzhled písma v malých i velkých velikostech. Dříve byla písma stejného řezu různých velikostí mírně odlišná (opticky škálovaná), nyní jsou písma ve větších velikostech pouze zvětšeninami písma v menších velikostech. Tento nedostatek je možné řešit tzv. *parametrizací písem* [3]. Tento mechanismus umožňuje písma popsaná pomocí kontur upravovat pouhou změnou několika (i mnoha) parametrů. Původcem tohoto mechanismu je pravděpodobně Donald E. KnuthRejstříkDonald E.Knuth, který jej implementoval v systému METAFONT.

Na obrázku 11 je mechanismus optického škálování znázorněn na písmenech a z fontů cmr5, cmr10 a cmbx10 z rodiny Computer Modern ve velikosti 3 cm. Rozdíly v kresbě jednotlivých písmen jsou patrné zejména z chvostu a paty písmene. Zrno je zejména u písmene a z tučného fontu cmbx10 nevýrazné, zatímco u fontů cmr5 a cmr10 je zřetelný přechod mezi zrnem chvostu a horní částí písmene. Pata je u tučného písmene a z fontu cmbx10 redukována.

Podobný mechanismus se pokusila do svého formátu Type 1 [9] implementovat i společnost Adobe, která v lednu roku 1994 zveřejnila dodatek ke specifikaci formátu Type 1 [10]. Tento dodatek doplňuje specifikaci fontů Type 1 o tzv. *Multiple Master fonty*. Multiple Master fonty byly společností Adobe používány již v roce 1992, tedy ještě před zveřejněním jejich specifikace.



Obrázek 11: Princip optického škálování

Obrázek 12: Multiple Master font s jednou osou

Type 1 font obsahuje definici jednoho řezu písma a postscriptový interpret může tento řez zvětšit, či zmenšit, vyplnit či jen vykreslit apod. Multiple Master font může obsahovat až osm párů těchto řezů, které jsou nazývány *hlavními* (master design). Každá dvojice definuje jednu „souřadnou osu“ a všechny hlavní řezy společně definují bázi prostoru fontu. V nejjednodušším případě bude Multiple Master font obsahovat pouze dva hlavní řezy, které budou definovat souřadnou osu jedné charakteristické vlastnosti písma (např. duktu). Multiple Master font znázorněný na obrázku 12 obsahuje dva hlavní řezy. První je znázorněn vlevo od souřadné osy (základní písmo). Druhý řez je tučná varianta téhož písma. Souřadná osa (design axis) tedy definuje duktus fontu. Výsledný font je funkcí parametru α, který specifikuje uživatel fontu. Parametr α musí být reálné číslo z intervalu $[0, 1]$. Pokud uživatel specifikuje hodnotu α rovnou 0 nebo 1, získá původní hlavní řez. Pokud specifikuje hodnotu z vnitřku intervalu, výsledný font vznikne lineární interpolací mezi hlavními řezy.

Multiple Master font může obsahovat více těchto os (maximálně však čtyři), a proto bude výsledný font lineární interpolací hlavních řezů závislou na vektoru parametrů:

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in [0, 1]^4$$

V Multiple Master fontu mohou být kromě hlavních řezů, které specifikují souřadnou osu, ještě další hlavní řezy zjemňující souřadnou osu (intermediate

designs). Celkový počet hlavních řezů však stále musí být menší nebo roven 16. Pokud budeme chtít použít v Multiple Master fontu čtyři osy, nebudeme moci použít zjemňující hlavní řez. To je obrovská nevýhoda, protože je velmi obtížné lineární interpolací krajních hlavních řezů dosáhnout kvalitního písma. Je také otázkou, zda je možné vzdálenosti v typografii interpolovat lineárně. Yannis Haralambous se v článku Parametrization of PostScript fonts through METAFONT—an alternative to Adobe Multiple Master fonts [3] domnívá, že kvadratická aproximace je pro tento účel vhodnější. Dalším důležitým aspektem zdůrazňujícím nedokonalost designu specifikace Multiple Master fontů je jiné rozmístění či počet kontrolních bodů pro definici jiných řezů písma. Tento aspekt není vůbec v Multiple Master fontech zohledněn.

Společnost Adobe však na počátku roku 2000 oznámila, že ukončuje podporu fontů Multiple Master a tak potvrdila faktický stav podpory Multiple Master fontů, jejichž specifikace je sice veřejně k dispozici, ale neobsahuje přesné a výstižné informace, nejsou k dispozici vzorové fonty, komerční Multiple Master fonty jsou příliš drahé, neexistují (až na velmi drahé komerční nástroje) utility pro jejich tvorbu a podpora ze strany aplikací je stále zanedbána. Opět se tak prokázala neúspěšnost uzavřených standardů a specifikací.

Písma ve formátu Type 42

Jazyk PostScript je určen pro popis tiskové strany a je používán v mnoha různých typech tiskáren. Postscriptové tiskárny (tedy tiskárny vybavené přímo postscriptovým interpretem či kartou implementující jazyk PostScript) jsou používány pro tisk i z operačních systémů používajících pro reprezentaci písem fonty, které nejsou přímo podporované jazykem PostScript. Např. operační systémy Windows společnosti Microsoft používají v grafickém uživatelském rozhraní technologii TrueType a pro tisk dokumentů obsahující tato písma je nutné vlastní kontury písmen konvertovat buď přímo do formátu Type 1 nativně podporovaného postscriptovým interpretem tiskárny nebo do bitmap.

Nevýhodou prvního přístupu (tedy konverze do Type 1) je ztráta hintovacích informací, které jsou ve formátu TrueType podstatně výkonnější, a také nepřesné kontury, neboť ve formátu TrueType jsou použity kvadratické B-spliny místo kubických Bézierových křivek formátu Type 1. Naproti tomu druhý přístup (konverze do bitmap) je naprosto nevyhovující pro velikost bitmap, které jsou zejména při velkých rozlišeních objemné.

Výše popsaná situace je tedy (resp. byla) zdánlivě neřešitelná. Společnost Adobe však do svého postscriptového interpretu zabudovala rasterizér fontů ve formátu TrueType a nová verze specifikace jazyka PostScript Level 3 již tyto fonty podporuje v podobě fontů ve formátu Type 42, který tvoří jakousi

postscriptovou obálku nad formátem TrueType. Specifikace formátu Type 42 [11] je opět veřejná a je k dispozici na serverech společnosti Adobe.

Písma ve formátu Type 42 mají podobnou strukturu jako Type 1. Definují slovník fontu o téměř stejném obsahuje. Slovník fontu Type 42 obsahuje navíc položku **sfnts**, která reprezentuje vlastní TrueType font v mírně pozměněné podobě. Položka **FontType** musí obsahovat hodnotu 42. Stejně jako u Type 1 fontů je předepsána (doporučena) i podoba první řádky souboru obsahujícího TrueType font:

```
%!PS-TrueTypeFont-TTVerze-VerzeFontu
```

Číslo *TTVerze* definuje číslo specifikace formátu TrueType (viz kapitola TrueType fonty) a číslo *VerzeFontu* obsahuje číslo verze vlastního fontu. Na následujícím výpise si uvedeme příklad fontu Verdana, který je distribuován společností Microsoft v operačních systémech Windows:

```
%!PS-TrueTypeFont-XXX-65536
%%VMusage: 0 0
11 dict begin
  /FontType 42 def
  /FontMatrix [ 1.0 0.0 0.0 1.0 0.0 0.0 ] def
  /FontName /Verdana def
  /FontInfo 8 dict dup begin
    /Notice (Typeface and data (c) 1996 Microsoft Corporation.
             All Rights Reserved) readonly def
    /UnderlineThickness 0.0585938 def
    /isFixedPitch false def
    /Version (Version 2.10) readonly def
    /UnderlinePosition -0.0878906 def
    /FullName (Verdana) readonly def
    /ItalicAngle 0.0 def
    /FamilyName (Verdana) readonly def
  end readonly def
  /Encoding 256 array
    dup 0 /.notdef put
    dup 32 /space put
    dup 33 /exclam put
    ... zkráceno
    dup 254 /thorn put
    dup 255 /ydieresis put
  readonly def
  /FontBBox [ -0.0498047 -0.206543 1.44678 1.03369 ] def
  /PaintType 0 def
  /sfnts  [
00010000001201000004002040C5453487930F0C20000012C000002474F532F32267590C3
0000037400000056564444D5874F17C6D000003CC000005E0636D617037C4A855000009AC
0000059C637674204CA740E300000F4800000198667067 6DEE371553000010E000000138
... zkráceno
23442B732B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B737373737373737373732B732B732B2B2B
2B2B73732B2B2B2B732B2B2B2B00752B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B73
747373742B73742B00>] def
```

```
  /CharStrings 579 dict dup begin
    /onehalf 242 def
    /Igrave 206 def
    ... zkráceno
    /q 84 def
    /kgreenlandic 358 def
  end readonly def
FontName currentdict end definefont pop
```

V první části výpisu je definován typ fontu a matice přechodu od báze souřadného systému písmene k uživatelskému souřadnému systému. Tato matice definuje identickou transformaci na rozdíl od matice přechodu obvyklé u Type 1 fontů. Následuje definice jména fontu a slovník **FontInfo**, kódovací vektor a samotný TrueType font. V závěru je již definován pouze slovník **CharStrings**, který obsahuje pouze odkazy na jednotlivé procedury definované uvnitř TrueType fontu.

Bibliografie

[1] Vydavatelství Úřadu pro normalizaci a měření. *Polygrafické názvosloví: Písmo, písmařství a písmolijectví.* Listopad 1974.

[2] Lindy Amato. *PostScript, Pre-Press a barva.* Computer Press, a. s., 1. edition, 1996.

[3] Yannis Haralambous. *Parametrization of PostScript fonts through METAFONT—an alternative to Adobe Multiple Master fonts.* Electronic Publishing Vol 6(3), Září 1993.

[4] Hans Hagen Thành, Sebastian Rahtz. *The pdfTEX user manual.* Září 1999.

[5] Adobe Systems Incorporated. *Adobe Type 1 Fonts–Communication Handbook.*

[6] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley Publishing Company, Inc., Reading, MA, USA, 2. edition, 1990.

[7] Adobe Systems Incorporated. *Supporting Downloadable PostScript Language Fonts (Technical Note Nr. 5040).* Březen 1992.

[8] Adobe Systems Incorporated. *Supporting Font in the PostScript Language Environment (Technical Note Nr. 5075).* Březen 1992.

[9] Adobe Systems Incorporated. *Adobe Type 1 Font Format.* Addison-Wesley Publishing Company, Inc., Reading, MA, USA, 3. edition, 1993.

[10] Adobe Systems Incorporated. *Type 1 Font Format Supplement (Technical Specification Nr. 5015).* Leden 1994.

[11] Adobe Systems Incorporated. *The Type 42 Font Format Specification (Technical Note Nr. 5012).* Leden 1994.

[12] Adobe Systems Incorporated. *Portable Document Format Reference Manual Version 1.3*. 1. edition, Březen 1999.
[13] Jiří Žára a kolektiv. *Počítačová grafika – principy a algoritmy*. GRADA, a. s., 1. edition, 1992.
[14] Adobe Developer Support. *The Compact Font Format Specification*. Adobe Systems Incorporated., 1. edition, Prosinec 1996.

Pavel Janík ml.
<Pavel.Janik@linux.cz>

Výroční cena CSTUGu

Výbor Československého sdružení uživatelů TEXu udělil výroční cenu

Štěpánu Potockému

za digitalizaci starších čísel Zpravodaje (do roku 1995 včetně). Práce byla oceněna finanční odměnou ve výši 5000 Kč.