Kódování postscriptových písem

Na rozdíl od rastrových písem, která jsou prakticky vždy svázána s nějakým konkrétním kódováním (kódovou stránkou), nejsou postscriptová písma nijak kódována; o konkrétním kódování (o významu jednotlivých bajtů tištěného textu) se rozhodne až těsně před tiskem. Jinými slovy, postscriptová písma obsahují *sadu* (množinu) znaků, z níž jednotlivé aplikace či přímo operační systémy při tisku vybírají ty znaky, které potřebují pro svoje kódování. V písmu samotném tedy není nikde určeno, jaký kód má být té které liteře přiřazen.

Tato nezávislost kódování spolu s nezávislostí jazyka PostScript™ jako takového má několik důsledků:

1) nezávislost postscriptových písem na platformě, na které se používají (s výjimkou formátu zápisu v souboru, který si určuje každý operační systém). Stejné písmo se použije jak pro Macintosh, tak pro Windows, pro různé DOS aplikace, pro OS/2, Unixové stroje (tedy i NeXT) a bůhvíco dalšího.
   Důsledek 1: vytištěné písmo vypadá vždy stejně, ať je na konkrétním zařízení vytištěno z kterékoli platformy.
2) možnost sdílení písem zavedených do tiskárny různými stroji (Mac, PC, Unix), případně i jejich sdílení na sdíleném síťovém disku.
   Důsledek 1: Zavedeme-li písmo Futura na pevný disk osvitky z počítače Macintosh, můžeme *totéž* písmo používat aplikacemi z Windows nebo kteroukoli aplikací ze systému DOS (pokud podporuje PostScript) bez nutnosti opětovného zavádění tohoto písma či snad dokonce zavádění téhož písma s jiným kódováním. Totéž platí i obráceně. Tím se pochopitelně značně snižují nároky na kapacitu disku i paměti osvitky a tisk se v důsledku toho zrychluje (zbývá více místa pro zápisníkovou paměť, po síti se přenášejí menší objemy dat atd.).
   Důsledek 2: všechny aplikace na PC používají stejná postscriptová písma, ať už pracují pod (různě kódovanými) systémy Windows (CP 1252), DOS (CP 437) nebo OS/2 (ISO Latin 1). Totéž platí i pro textový procesor WordPerfect, který pracuje ve vlastní dvoubajtové znakové sadě, vycházející ze systému Unicode. To opět znamená úsporu místa na disku, tentokrát na disku počítače.
3) neomezený počet liter v písmu: postscriptové písmo není omezeno na 256 znaků, jak je tomu u ostatních (kódovaných) písem, ale může obsahovat libovolně velikou sadu; většinou obsahuje různé typografické symboly, uvozovky, akcentované znaky a podobně. To má ovšem za následek, že aplikace, které jsou schopny pracovat jen s jednobajtovou znakovou sadou, tedy s 256 znaky, nemusejí být schopny využít všechny litery, které jsou v postscriptovém písmu obsaženy. To je například případ Windows 3.0.

Poznamenejme, že podobný přístup, který umožňuje nezávislost kódování, je částečně implementován i v ostatních formátech tvarových písmech (Fontware, Speedo, Intellifont, TrueType); v žádném jiném případě se ale nepodařilo vytvořit systém natolik obecně přijímaný jako v případě formátu PostScript.

Výše uvedené nezávislosti kódování se dosahuje velice jednoduchým způsobem: všechny litery jsou v postscriptovém písmu *pojmenovány* (litera A se jednoduše nazývá `A`, hvězdička se nazývá `asterisk`, levá závorka `parenleft` a tak podobně). Pod stejnými jmény potom k literám přistupují jak ovladač postscriptové (a žádné jiné) tiskárny, tak rasterizéry (ATM a FontFoundry), a to na základě tabulek, které jsou v nich obsaženy. Nepoužívá se tedy dvoubajtové kódování liter, jako je například Unicode; důvodem je mj. to, že v době vzniku postscriptových písem žádný obecně přijímaný systém dvoubajtového kódování, jako je například Unicode, ještě neexistoval; vznikl až později ve spolupráci firmy Adobe s ISO, Apple a ostatními organizacemi a firmami. Je zcela pochopitelné, že postscriptové ovladače jednotlivých aplikací musejí dodržovat

konvenci pojmenovávání liter, jak ji zavedla firma Adobe a jak je uvedena v příslušné ISO normě; v opačném případě by výše popsaný mechanismus nepracoval.

Podívejme se nyní, jak je výše uvedený princip technicky realizován; nejdříve se ale musíme zmínit o pojmu *slovník,* jak je chápán v jazyce PostScript. Datová struktura *Slovník* (dictionary) v jazyce PostScript je seznam *pojmenovaných* datových struktur. Jménem přitom je libovolné jméno jazyka PostScript, jakási obdoba atomu, jak je znám z jazyků LISP nebo Prolog. Tento seznam pojmenovaných datových struktur si lze také představit jako seznam *dvojic*, kdy první člen obsahuje jméno dvojice (tedy přesněji ukazatel do tabulky jmen) a druhý člen obsahuje ukazatel na konkrétní datovou strukturu. Tyto slovníky jsou prakticky shodné se slovníky, jak jsou známy z jazyka Forth. Podstatnou skutečností je to, že přístup k jednotlivým položkám slovníku se děje zásadně prostřednictvím jména, nikoli například pomocí indexu, jak je tomu v případě polí, nebo pomocí směrníku, jak je tomu u klasických spojovaných seznamů.

A tím se dostáváme k jádru věci: každé písmo je v podstatě slovník, tedy seznam struktur, popisujících litery, které jsou pojmenovány příslušnými jmény. Při rastrování textu tedy hledá rutina BuildChar příslušnou literu v daném písmu, tedy v jeho slovníku, pomocí jména. Text, který se má vytisknout, ovšem aplikace do tiskárny posílá jako jednotlivé bajty (mám na mysli pochopitelně aplikaci, pracující s jednobajtovým kódováním), které se teprve v tiskárně převedou na postscriptové jméno. Tento převod se děje přes tzv. kódovací vektor (encoding vector), což je jednoduché pole 256-ti prvků, kde každý prvek obsahuje postscriptové jméno příslušející kódu, danému pozicí tohoto prvku; popisuje-li tedy kódovací vektor kódování ASCII, bude na pozici 65 tohoto vektoru postscriptové jméno `A`, protože v kódu ASCII má znak A kód 65. Jinými slovy, každý bajt textu, který se má tisknout, se použije jako index kódovacího vektoru, čímž získáme postscriptové jméno litery; toto jméno se pak použije pro hledání ve slovníku liter (tedy v písmu).

Kódovací vektor není součástí písma, ale zavádí ho při tisku do tiskárny každá tisková úloha; proto se může každá úloha tisknout v jiném kódování při použití stejného písma. Je také vcelku pochopitelné, že do tiskové úlohy zapíše kódovací vektor daná aplikace či operační systém, přesněji řečeno její tiskový ovladač. Budeme-li tedy chtít, aby aplikace pracovala v jiné znakové sadě, musíme změnit kódovací vektor, který je v postscriptovém ovladači obsažen; písmo samozřejmě zůstane beze změny.

Je důležité si uvědomit, že výše popsaný mechanismus tiše předpokládá, že všechny litery, které aplikace z písma používá (tedy které jsou uvedeny v kódovacím vektoru), jsou v písmu skutečně obsaženy; pokud by požadovaná litera v písmu scházela, nenašla by ji rutina BuildChar ve slovníku (= v písmu) a vytiskla by místo ní mezeru nebo jiný náhradní znak. Tento problém není řešen žádným speciálním způsobem; namísto toho všechna písma Adobe obsahují jakousi minimální znakovou sadu, která je v písmu vždy obsažena a na jejíž existenci se může aplikace spolehnout. Tato znaková sada o 228 literách se nazývá Adobe Standard Character Set nebo také ISOAdobe (je určena ISO normou) a je obsažena ve všech písmech s latinkovou abecedou. Podobná sada existuje pro písma cyrilická nebo pro japonské znaky. Písma, která neobsahují tuto standardní znakovou sadu, ať už se jedná o písma z expertní sady nebo o různé ornamenty, se zpracovávají poněkud jiným způsobem a budeme o nich psát dále. Poznamenejme ještě, že tato písma se standardní znakovou sadou ISOAdobe se také často označují jako písma kódovaná v Adobe Standard Encoding. Standardní písma totiž ve skutečnosti *jsou* kódovaná, jakkoli se vám v celém článku snažím namluvit opak, a to právě ve znakové sadě Adobe Standard Encoding. Žádná jiná písma přitom v tomto kódovaní zakódována nejsou. Důvod, proč jsou tato písma kódována, je pouze historický, a jejich kódování se ve skutečnosti vždy ignoruje, což neplatí pro žádné jiné kódovaní. V praxi tedy platí, že písma s kódováním Adobe Standard Encoding obsahují znakovou sadu Adobe Standard Character Set (= ISOAdobe) a při tisku se vždy chovají jako nezakódovaná, tedy se překódují do potřebného kódu.

O písmech kódovaných v jiných kódech a zvláště pak o problémech se středoevropskou znakovou sadou se zmíníme v příštím pokračování seriálu.