Zpravodaj Československého sdružení uživatelů TeXu

Petr Macháček

Počeštěte si PostScriptový font

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 8 (1998), No. 3-4, 153–159

Persistent URL: http://dml.cz/dmlcz/149824

Počeštěte si PostScriptový font

Petr Macháček

Abstrakt

Mnoho uživatelů nejen TEXu už někdy pocítilo potřebu mít variantu nějakého PostScriptového fontu obsahující české znaky. *Počeštěním* zde budeme rozumět zprovoznění požadovaného fontu pro práci s českými znaky bez perfekcionistických nároků na typografickou kvalitu dosaženého výsledku. Čím dokonalejší má být výsledek, tím více ruční práce je třeba mu věnovat. Zaměříme se na použití tzv. kompozitních znaků, vytvářejících kresbu českého znaku složením kresby základního písmene a akcentu. Zvídavější či náročnější čtenáře odkážeme na obecný program `t1accent`.

Velice stručné vysvětlení několika pojmů

PostScriptových fontů může být několik druhů. Nejčastěji se jimi rozumí tzv. Type1 fonty, šířené v souborech `.pfa` nebo `.pfb` (Printer Font Ascii/Binary) — ty také většinou máme k dispozici, začneme-li uvažovat o počeštění. Font je spíše než statickým popisem jednotlivých znaků programem ve (zjednodušeném) PostScriptu. Každý znak je pojmenovanou procedurou, jejíž vyvolání (jedním z definovaných způsobů) vytvoří kresbu požadovaného znaku.

Jména těchto procedur jsou nezávislá na konkrétním rozložení znaků v používaném fontu. Jejich vyvolání při vypisování textu je řízeno tzv. Encoding vektorem, polem obsahujícím pro každý z 256 možných kódů znaků název procedury, která by měla znak s tímto kódem vykreslit (nejčastějším jménem je ovšem vyhrazené slovo `.notdef` — příslušný znak nemá definovaný vzhled). Při použití fontu v PostScriptu se dá snadno říci, jaký Encoding vektor má být aplikován.

Většina znaků je pojmenována prostě a intuitivně. Písmeno A se jmenuje `A`, mezera `space`, znaky samostatných akcentů třeba `caron` (háček) či `acute` (čárka). Nás budou zajímat hlavně znaky sestavené ze základního písmene a akcentu; i jejich názvy tuto skutečnost odrážejí — á je `aacute`, Ř je `Rcaron` (ale třeba ť a ď se jmenují `tcaron` a `dcaron`, přestože pro kresbu je většinou použit akcent `quoteright`).

Co potřebujeme

Chceme-li počešťovat font, máme už nejspíš font obsahující základní znaky a potřebujeme doplnit k některým z nich akcenty. Většina fontů, které za počeštění stojí, obsahuje i samostatné akcenty. V nejhorším můžeme sami vytvořit předpis pro vykreslení akcentů. Máme-li hotové základní znaky i akcenty, musíme také vědět, jak je sesadit k sobě. Ukazuje se, že právě toto bývá největší problém — výsledný znak by měl dobře vypadat nejen sám o sobě, ale i v hladkém textu začleněný do slov (o udržení jednotného stylu nemluvě).

Nakonec nám zbyde jen v podstatě triviální úloha — sesadit již existující kresby znaků a akcentů podle nějak získaných pravidel k sobě tak, aby vznikl nový použitelný font. Soubory `.pfa`, `.pfb` vypadají na první pohled nečitelně, neboť jsou nejen kvůli úspoře místa zakódovány. Naštěstí je jejich formát podrobně dokumentován a existují i volně dostupné utility pro převod do čitelné textové podoby a zpět. Úprava čitelného zdrojového textu je snadno proveditelná ať už editorem či automatizovaně. Navíc se někdy můžeme obejít i bez úprav výchozího fontu — což se může hodit, je-li například font dostupný jen v tiskárně nebo nechceme riskovat právní problémy.

Případné nadšení problematiky ne až tak znalých čtenářů nad jednoduchostí celého řešení možná zchladí výhled na obtíže, které přinesou pokusy o konečné začlenění vytvořeného fontu do pracovního prostředí. Ale to by byl námět na značně rozsáhlou a s TEXem ještě méně související studii.

Možné přístupy

Ač by se mohlo zdát, že počeštění PostScriptového fontu je jednoznačně definováno jako tvorba nového Type1 fontu, může být občas schůdnější či dokonce vhodnější dosáhnout požadovaného efektu i jinými prostředky.

Počeštění bez práce s PostScriptem

Je zřejmé, že půjde o řešení problému v případě použití nepočeštěného PostScriptového fontu v nějakém vyšším programu. Nemá valného smyslu zabývat se zde něčím jiným než TEXem. TEX disponuje mocným mechanismem virtuálních fontů, díky němuž lze takřka libovolně sestavit požadovaný font z jiných fontů, ba i dalších objektů. Jedním z programů umožňujících takto vytvořit český font je `accents` (varianta pro ISO-Latin-2 kódování fontů `l2accents`) Jiřího Zlatušky [2], dobře popsaný v článku [1].

Chceme-li si nachystat řešení použitelné ve více vyšších programech, můžeme se přiblížit PostScriptu použitím AFM (Adobe Font Metrics) souboru doplňujícího základní font o dodatečné informace dobře využitelné při sazbě, ale nezpracovávané samotným PostScriptovým RIPem. V AFM souboru jsou kromě

metrik jednotlivých znaků uvedeny například kerningové či ligaturní údaje umožňující sofistikovanějším programům generovat typograficky hodnotnější výstup. Lze v něm též popsat složení kompozitních znaků. Použití takto zadefinovaných znaků v TEXu umožňují už standardní utility pro práci s PostScriptovými fonty (`afm2tfm`). Vytvoření `.afm` souboru s potřebnými složeninami za nás může obstarat program Petra Olšáka `a2ac` [3], jehož součástí je i tradičně obsáhlá a poučná dokumentace.

Využití PostScriptu, nezasahování do Type1 fontu

Už výše byla zmíněna poměrně značná flexibilita PostScriptu v zacházení s fonty. Jsou-li dodržena jistá pravidla, lze vytvořit font použitelný (z hlediska interpretu PostScriptu) stejně jako Type1 fonty, ale využívající plné síly PostScriptu. Procedura příslušná znaku takovéhoto fontu není omezena na použití několika málo vybraných příkazů, ale může kupříkladu vyvolat vykreslení znaků z jiného fontu. Tyto tzv. Type3 fonty ([6, strana 278]) lze chápat obdobně virtuálním fontům TEXu (samozřejmě je ale nutné mít na paměti, že TEX je sázecí systém, kdežto PostScript jazykem pro popis stránek — tedy něco mezi METAFONTem, TEXem a `DVI` formátem).

Vytváření Type3 fontů je rozumným kompromisem mezi uplným obcházením PostScriptu vázaným na konkrétní vyšší program (TEX) a úpravou Type1 fontů, která může být v rozporu s licenčními podmínkami jejich šíření. Použití Type3 fontů je ovšem vázáno na plnohodnotný interpreter PostScriptu (RIP – Raster Image Processor), což postačuje například pro úpravy tiskových výstupů z různých programů (XFig, Netscape,...), ale nelze je použít tam, kde jsou očekávány Type1 fonty (např. v X Window systému).

Pomocí Type3 fontů se požadavkem na tisk nestandardních znaků vyrovnává balík `ogonkify` [10] od Juliusze Chroboczka. Primárním cílem je zajistit tisk PostScriptu obsahujícího znaky s akcenty generovaného například Netscapem (ale i ApplixWarem ci StarOfficem). K tomu ale je ale zapotřebí font obsahující kresby takovýchto znaků. V balíku je pro tento účel program pro vytváření Type3 fontů s kompozitními znaky, jejichž složení ale musí popsat uživatel — přibaleny jsou patrně ručně doladěné metriky pro základní PostScriptové fonty (rodiny Times, Courier, Helvetica), pro vytvoření dalších je možné použít výše zmíněný program `a2ac`.

Skutečné počeštění Type1 fontu

Úprava a vytvoření nového `.pfa/.pfb` souboru obsahujícího definice všech českých znaků a správný Encoding vektor je bez pochyb nejuniverzálnější metodou, jak se s počeštěním vypořádat. Výsledný font lze použít všude tam, kde výchozí, včetně případů, kdy by jiné řešení ani nebylo možné (např. zobrazo-

vání v X Window). Do úvahy však místo technických problémů přichází problémy právní [7, strany 5,6]. Mezi činnosti pokryté copyrightem patří kromě šíření i úpravy programu (jak je Type1 font chápán) a bez souhlasu držitele copyrightu jsou tyto aktivity ilegální. Což nemusí příliš vadit při osobním používání (pro které jsou u některých fontů úpravy povoleny), ale pokud byste je chtěli zahrnout třeba do dokumentu vystaveného na Webu nebo je zaslat s dokumentem k dalšímu zpracování, můžete se dočkat nemilých překvapení. Různé fonty mají ovšem různé licence, takže překvapeni můžete být i příjemně.

K převodu zakódovaného fontu do čitelné podoby slouží program `t1disasm` z balíku `t1utils` [11]. (Jeho dvojče `t1asm` pak zařídí převod upraveného čitelného textu fontu do zakódované podoby.) Nad takto získaným programem lze provádět různé úpravy. Fakt, že se nevytváří nový program, ale upravuje již existující a s dodatečnými úpravami většinou nepočítající zdrojový kód, s sebou nese nebezpečí nepředpokládaného formátu vstupního souboru. Vypořádat se se všemi možnostmi by totiž vyžadovalo implementovat poměrně věrně některé části PostScriptového RIPu.

Pro praktické použití sice nevhodný, ale zmínku si jistě zasluhující program je `mkt1font` z balíku `accfonts` [12] indologa Johna D. Smithe. `mkt1font` je těžko konfigurovatelný (akcenty umisťuje na střed, většina parametrů by se musela upravovat v Perlovém zdrojáku, složení znaku určuje z jeho názvu), poměrně málo odolný vůči změně formátu vstupního souboru (například doplnění fontu od BitStreamu používajícího jiné konvence než Adobe je do značné míry záležitostí ručních oprav) a složené znaky vytváří kopírováním příkazů základního znaku a akcentu do nové procedury. Nicméně tento program měl primárně posloužit svému autorovi k vytvoření fontů vhodných pro sazbu indických jazyků a svůj účel patrně plní.

Po hraní si se skládáním kompozitních znaků, kdy na vykreslení celého znaku stačily v podstatě tři příkazy (základní znak, posun, akcent), může někomu doslovné kopírování všech příkazů pro vykreslení potřebných tahů připadat neefektivní. Bude-li hledat jinou cestu, najde možná operátor `seac` — Standard Encoding Accented Character. Na první pohled se jeví jako to pravé řešení pro vytváření kompozitních znaků v Type1 fontu. Podle definice ovšem vyžaduje, aby jím spojované znaky (základní a akcent) měly stejný kód, jaký jim byl přidělen ve vektoru Adobe Standard Encoding — což bohužel o akcentech, chceme-li font počeštit v souladu s ISO-Latin-2 normou, neplatí. Samotná firma Adobe dnes již používání tohoto operátoru nedoporučuje [9]. Nicméně při bližším prozkoumání kupříkladu Ghostscriptu [13] lze zjistit, že při použití číselných kódů akcentů podle Adobe StandardEncoding, bez ohledu na jejich pozici v právě vytvářeném fontu, operátor `seac` pracuje tak, jak bychom chtěli. Experimentálně bylo ověřeno, že takto vytvořené fonty fungují správně i v X serveru a na HP LaserJet 4 a 5. Pokud ale někdo upravuje font jen pro vlastní potřebu, může pro snazší práci a úsporu paměti zkusit porušit definici Type1 formátu. `;-)`. Operátor `seac`

má poměrně jednoduchou syntaxi vhodnou pro snadné vytváření kompozitních znaků přesně podle metriky vytvořené např. programem `a2ac`. Rozhodne-li se někdo nedbat dobrých rad a použít ho, měl by si dát pozor na správný význam argumentů [8].

Adobe doporučovanou [9] metodou je uložení kresby akcentů (a v podstatě i základních znaků) do pomocných procedur, kde budou uloženy jen jednou, a při tvorbě konkrétních znaků se na ně odvolat. Tohoto postupu se poměrně věrně drží program `t1accent` [4] Petra Olšáka použitý autorem při tvorbě PostScriptové varianty $\mathcal{CS}$-fontů [5]. `t1accent` je velmi obecný program umožňující mnohem víc, než jen prosté sesazení základního znaku a akcentu (příkladem budiž možnost různých tvarů akcentů pro minusky a verzálky). Není patrně primárně určen pro zběžné počeštění fontu na laickému uživateli dostačující úrovni, ale dovoluje (vzhledem k šíři svých možností) relativně snadno aplikovat typografické speciality. Za to ovšem platí poněkud neoperativním ovládáním. Osobní poznámka — jinak velice robustně napsaný program se odmítne zabývat fontem, který nezavedl zkratku pro často používané operátory (`noaccess put`), a pokusy o nápravu v případě Céčkového programu mne osobně dovedly ke `core` souboru, což se mi s Perlovými skripty dosud nepodařilo`;-)`. Přesto se zdá, že chce-li někdo počešťovat kvalitně font na úrovni `.pfa/.pfb` souboru, je `t1accent` nejdokonalejší dostupný nástroj.

Poznámky

Něco málo o kódování

Laskavý čtenář už patrně pochopil, že vlastní kresba znaků je na kódování fontu nezávislá (pomiňme nepodporovaný operátor `seac`). Přesto je třeba věnovat kódování jistou péči. Podprogram daného jména *musí* vykreslovat znak tímto jménem označený. To často není splněno u fontů převedených do Type1 třeba z TrueType pomocí různých okenních přípravků. V původním fontu samozřejmě nebyly jednotlivé znaky pojmenované a při převodu jsou znakům přiřazena nějaká implicitní jména, buď podle Adobe StandardEncoding nebo i hůře. Takový font pak není rozumně použitelný.

Uznávané standardy znakových sad jsou obsahem RFC1345, trochu horší je to s Encoding vektory pro Type1 fonty — kanonický Adobe StandardEncoding je všude, ISOLatin1Encoding též, ale oficiální vektor pro kódování PostScriptových fontů podle ISO-8859-2 jsem nikde nenašel. Téměř každý z výše zmíněných balíků obsahuje nějaký přepis popisu znaků do PostScriptových názvů. Otázka, zda se třeba znak 187, `SMALL LETTER t WITH CARON` má skutečně jmenovat `tcaron` kvůli kompatibilitě s `Tcaron`, nebo `tquoteright`, aby odpovídal název grafickému provedení, se nicméně jeví býti rozhodnuta ve prospěch `tcaron`. V prostředí TEXu lze najít i rozšířené schéma obsahující na pozicích neobsazených v ISO-8859-2 pro sazbu užitečné znaky.

Jak určit sesazení znaků s akcenty

Umístění akcentů ke znakům lze řešit buď individuálně pro každý případ, nebo se lze pokusit o zautomatizování této činnosti. Extrémní přístupy (například vycentrování všech akcentů nad základním znakem) však natropí více škody než užitku. Vhodným nástrojem je dobře konfigurovatelný program `a2ac`, vytvářející AFM soubor s (nejen) popisem sesazení jednotlivých kompozitních znaků. Excesy lze pak snadno opravit ruční editací přehledného AFM souboru. Z informací obsažených v AFM je pak možno vyjít při vlastním sestavení výsledného fontu.

Jaký přístup k řešení zvolit

Nejpoužitelnějšího výsledku lze bezpochyby dosáhnout počeštěním Type1 fontu. Na druhou stranu, pokud vám jde třeba jen o sazbu v TEXu, je zbytečné odvolávat se až do výsledného PostScriptového souboru na svůj nový font a muset ho případně dodávat v něm. Navíc, snažíme-li se používat pro češtinu některé oblíbené leč nepříliš volně šířené fonty, mohli bychom se při příliš hlubokých zásazích dostat do sporu s licenčními podmínkami.

Pokud se rozhodneme řešit počeštění nějakého písma kompozitními znaky sestavenými z již hotových znaků a akcentů, je patrně nejvhodnější vytvořit si jeden soubor popisující složení — nabízí se AFM metrika — a z něj pak vycházet při dalším upravování. Ať už jde o virtuální fonty, Type3 nebo Type1 fonty. To nám umožní kupříkladu mít vytvořený Type1 font pro soukromou potřebu bez ohledu na právní aspekty a využívat ho třeba v grafických nadstavbách (LyX), a vyskytne-li se potřeba šířit vlastní PostScriptový dokument obsahující tento font, jednoduše nahradit Type1 font Type3 fontem stejného vzhledu.

Klademe-li si vysoké typografické nároky, nemůžeme se spokojit s kompozitními znaky (tvar akcentu bude například muset záviset na tvaru základového znaku). V tom případě již není možné vyjít jen z metriky, ale musíme nějak ovlivnit přímo kresbu znaků. To ze zde uvedených utilit podporuje pouze `t1accent`. Nemělo by být ovšem příliš těžké podle definičního souboru případně opět vytvořit Type3 font, nastaly-li by nějaké problémy s používáním Type1 fontů.

Přestože asi neexistuje jednoznačný návod, vyplatí se vždy alespoň tušit, co se při použití toho kterého programu děje a jak věci fungují.

Závěr

Nezbývá než přiznat, že přečtení tohoto TEXtu vás jistě nenaučilo bez dalšího úsilí snadno počešťovat PostScriptové fonty. Snažil jsem se ale poskytnout jistý vhled do problematiky a souhrn dostupných pomůcek včetně svého, ne nutně správného, názoru na ně.

Pokud se vám úspěšně podaří počeštit nějaké písmo, můžete se o svou radost zkusit podělit se správcem stránky Fontanasia [14]. Budete-li si chtít otestovat svůj PostScriptový font, můžete velmi dobře použít Ghostscript (např. soubor `prfont.ps`), pokud chcete přidat PostScriptový font v ISO-8859-2 kódování do Xserveru, bývá často potřeba uvést font v souboru `fonts.dir` pod označením končícím kódováním `-adobe-fontspecific`, a teprve v souboru `fonts.alias` mu vytvořit přezdívku se správným označením kódování `-iso8859-2`.

Elektronická verze tohoto článku, jakož i některé pokusy autora o hrátky s PostScriptovými fonty, lze najít na URL `http://www.fi.muni.cz/~xmachac/slt98/`.

Zdroje informací

[1] Sojka, Petr: *Virtuální fonty, accents a přátelé*, `ftp://ftp.fi.muni.cz/pub/tex/local/cstug/sojka/aboutacc`

[2] Zlatuška, Jiří: `accents, l2accents`, `ftp://ftp.fi.muni.cz/pub/tex/local/fontware/accents`

[3] Olšák, Petr: `a2ac`, `ftp://math.feld.cvut.cz/pub/olsak/a2ac/`, `CTAN:fonts/utilities/a2ac`

[4] Olšák, Petr: `t1accent`, `ftp://math.feld.cvut.cz/pub/olsak/t1accent/`

[5] Olšák, Petr: `t1csfont`, `ftp://math.feld.cvut.cz/pub/olsak/t1csfont/`

[6] Adobe: *PostScript language reference manual (RedBook)*, ISBN 0-201-18127-4

[7] Adobe: *The Adobe Type 1 Font Format (BlackBook)*, ISBN 0-201-57044-0, `http://www.adobe.com/supportservice/devrelations/PDFS/TN/T1_SPEC.PDF`

[8] Adobe: *Adobe Technical Note 5015, Type 1 Font Format Supplement*, `http://www.adobe.com/supportservice/devrelations/PDFS/TN/5015.Type1_Supp.pdf`

[9] Adobe: *Composite Characters*, `http://www.adobe.com/supportservice/devrelations/typeforum/composit.html`

[10] Chroboczek, Juliusz: `ogonkify`, `http://www.dcs.ed.ac.uk/home/jec/programs/ogonkify/`

[11] Hetherington, I. Lee: `t1utils`, `CTAN:fonts/utilities/t1utils`

[12] Smith, John D.: `accfonts`, `ftp://bombay.oriental.cam.ac.uk/pub/john/software/fonts/accfonts/`

[13] `Ghostscript`, `http://www.cs.wisc.edu/~ghost/`

[14] Kvasnička, Michal: *Fontanasia*, `http://www.econ.muni.cz/~qasar/fontanasia/`

Petr Macháček, `xmachac@fi.muni.cz`