Formáty fontů v typografických systémech a jejich převod

Petr Olšák

Abstrakt. Úvod do problematiky formátu fontů. Jak aplikace na zpracování textu pracují s fonty. Kódování dokumentů a návaznost na kódování fontů. Metrické údaje fontů. Podrobnější popis formátu Type-1. Problémy při interpretaci licenčních textů ke komerčně šířeným fontům.

Zavedení pojmů

Nejprve bychom si měli říci, co budeme rozumět pod slovem font. Jednou větou by se dal tento pojem vyjádřit jako implementace nějakého písma v počítači. Zkusíme se na to podívat podrobněji.

Tušíme, že font musí obsahovat informace o jednotlivých znacích písma a tyto informace mohou být v různých formátech fontů implementovány rozličným způsobem. Bez závislosti na formátu fontů obsahuje každý font informace dvojího druhu. Jednak jsou to *metrické údaje* a za druhé pak samotné *kresby znaků*. V metrických údajích jsou pokyny pro sázecí stroj, jak vypadají základní rozměry znaků, jaké je doporučeno dodatečné mezerování mezi některými dvojicemi znaků (tzv. *kerning*) a které skupiny znaků se mají nahradit slitkem neboli *ligaturou*. Některé formáty mohou být navrženy poněkud nevhodně tak, že v nich některé z uvedených informací nelze implementovat a jiné naopak mohou obsahovat mnoho dalších podobných pokynů pro sázecí stroj.

Sázecím strojem rozumím takový algoritmus, který má za úkol zpracovat text ve zvoleném fontu a vypočítat z toho plynoucí rozměry sazby. Obvykle sázecí stroj obsahuje také algoritmy na dělení slov, což je součást problematiky optimálního rozvržení sazby v zadaných rozměrech. Ideální sázecí stroj vůbec nepracuje s vlastními kresbami znaků, ale pro výpočet sazby mu naprosto stačí metrické údaje fontu. Příkladem takového sázecího stroje je TEX.

Výstup sázecího stroje (tj. na definitivní návrh rozložení sazby) je obvykle potřeba prohlédnout na monitoru počítače, aby si uživatel udělal představu, jak ta sazba vlastně vypadá. O to se starají *algoritmy náhledu*, které potřebují pracovat s tvary jednotlivých znaků. Na druhé straně tyto algoritmy už nemusejí číst metrické informace, protože pracují s definitivním rozložením sazby a doplňují do jednotlivých míst sazby podle pokynů sázecího stroje kresby požadovaných znaků. Příkladem algoritmů náhledu jsou algoritmy, které zobrazují výstup TEXu (tzv. dvi) na monitoru. Ty jsou implementovány třeba v programu `xdvi`.

Sázecí stroj a algoritmy náhledu jsou součástí každého interaktivního textového procesoru nebo DTP programu. Hranice mezi sázecím strojem a algoritmy náhledu je zde ovšem jen teoretická, protože obojí bývá součástí jediné aplikace. Kvalita a možnosti sázecího stroje v těchto aplikacích většinou výrazně pokulhávají za specializovanými sázecími stroji, jakým je třeba TEX. Ilustruji to na konkrétním příkladě. V posledních verzích InDesignu se objevily novinky, které konečně zahrnují některé již 20 let používané vlastnosti z TEXu. Mám na mysli například automatické nahrazování ligatur podle metrických údajů ve fontu nebo algoritmus výpočtu optimálního zlomu odstavce. Na některé běžné vlastnosti TEXu si uživatelé komerčních DTP programů budou asi muset počkat, až se možná objeví v budoucích verzích (např. algoritmy pro výpočet sofistikované matematické sazby), a některé vlastnosti jsou pro tyto uživatele z principu nedostupné (např. důkladná dokumentace sázecího stroje včetně přiložených podrobně dokumentovaných zdrojových textů).

Algoritmy náhledu musejí úzce spolupracovat s *rasterizérem fontů*, což je algoritmus starající se o převedení popisu tvaru jednotlivých znaků z interního formátu fontu do bitové mapy v konkrétním rozsahu. Tento rozsah se vypočítá ze dvou parametrů: z velikosti písma v sazbě a z rozlišení výstupního zařízení. S rostoucí velikostí písma nebo s rostoucím rozlišením výstupního zařízení roste i rozsah bitové mapy jednotlivého znaku.

Příkladem rasterizéru fontů je třeba METAFONT. Ten převádí tvary znaků implementované ve formátu METAFONT (pomocí matematických křivek a pokynů pro tahy pera přesně definovaného tvaru) do bitových map, se kterými si poradí například již zmíněný prohlížeč sazby `xdvi`. Jiným příkladem rasterizéru je Adobe Type Manager (ATM), který rastruje fonty formátu PostScript Type-1 pro potřeby zobrazení znaků na monitoru počítače. K tomuto rasterizéru se vrátíme později. Ještě jiným příkladem rasterizéru může být PostScriptový RIP (Raster Image Procesor) v tiskárně nebo osvitové jednotce. Ten se stará nejen o rastrování znaků použitých fontů, ale též o rastrování obrázků. Obrázky se zde nebudeme zabývat, protože to přesahuje téma tohoto příspěvku.

Algoritmy náhledu musejí při kresbě znaku řešit navíc jeden problém, který se obvykle při konečném tisku dokumentu nevyskytuje. Tímto problémem je otázka, jak vykreslit jednotlivé znaky na monitoru počítače tak, aby to i přes nízkou rozlišovací schopnost tohoto zařízení bylo aspoň čitelné. K tomuto účelu se ke kresbám znaků do některých formátů fontů přibalují ještě tzv. *hintovací informace*, což jsou pokyny pro rasterizér, jak se má vyrovnat s rastrováním pro nízká rozlišení. Jinou možnost řešení tohoto problému najdeme v náhledových programech výstupu TEXu (např. už dvakrát zmíněný program `xdvi`). Tyto programy požádají rasterizér o bitmapu fontu typicky v rozlišení výstupního zařízení 600 dpi (to je samozřejmě možné konfigurovat i jinak). Tyto bitmapy pak mohou být zobrazeny do výstupní bitmapy monitoru v různých velikostech náhledu tak, aby se $n \times n$ pixelů bitmapy znaku promítlo na jeden pixel monitoru, kde n je celé číslo. Velikosti náhledu lze tedy měnit pouze skokem. Na druhé straně lze při tomto postupu používat pro danou velikost náhledu až $n \times n$ různých úrovní šedi pro výsledný pixel monitoru. Úroveň šedi lze počítat podle poměru černých a bílých pixelů na odpovídající části vstupní bitmapy znaku. Tento trik známý pod názvem *antialiasing* výrazně zlepší čitelnost písma na monitoru.

Někdy se stává, že nám nestačí náhled dokumentu na obrazovce počítače, ale potřebujeme dokument vytisknout. V takovém případě přichází ke slovu znovu rasterizér fontu, který nyní rastruje písmo do rozlišení podle parametrů tiskárny nebo jiného výstupního zařízení. Rozdíly mohou být pouze v tom, kde tento rasterizér pracuje. Často se stává, že rasterizér pracuje přímo na našem počítači při přípravě tiskové úlohy. Tisková úloha pak obsahuje bitové mapy jednotlivých znaků, které si tiskárna sestaví do bitových map celých stránek a vytiskne. S tímto přístupem se setkáváme u jednoduchých tiskáren, které nemají implementován vlastní rasterizér, a dále u písem, která jsou přítomna v počítači jen v takovém formátu, který je rasterizéru tiskárny nesrozumitelný. Příkladem může být PostScriptový RIP v tiskárně a font implementovaný jen ve formátu METAFONT. Pak musí program METAFONT připravit ještě v počítači bitové mapy znaků, tiskový ovladač generující PostScriptový kód pro tiskárnu tyto bitmapy do výstupního kódu zařadí a RIP tiskárny tyto bitmapy použije (takový bitmapový font je z pohledu PostScriptového RIPu implementován jako font Type-3).

Jinou možností je případ, kdy rasterizér pracuje ve výstupním zařízení. Typickým příkladem takového rasterizéru je PostScriptový RIP. Použitý font v počítači by měl mít v ideálním případě formát PostScript Type-1. Ovladač generující PostScriptový kód pro tiskárnu pak pouze zařadí (mírně modifikovanou) kopii tohoto fontu do tiskové úlohy. Tomu říkáme, že font je v PostScriptovém kódu *downloadovaný*. RIP tiskárny pak podle tohoto fontu bude rastrovat znaky do výstupních stránek dokumentu určených k tisku.

Ideální případ popsaný v předchozím odstavci může mít výjimky. Buď je font v jiném než Type-1 formátu nebo není downloadovaný vůbec. RIP tiskárny totiž disponuje vlastní sadou fontů, které může použít. Ve standardní sadě fontů, která musí být instalována v každém PostScriptovém RIPu (jinak se na ten RIP díváme jako na méněcenný), je skupina 35 fontů od Adobe z rodin Times, Helvetica, Courier, AvantGarde, Bookman, NewCentury, Palatino, ZapfChancery, ZapfDingbats a Symbol. Pokud v dokumentu používáme tato písma, pak není potřeba je ovladačem downloadovat, protože RIP je musí znát. Tím se sníží velikost PostScriptového kódu, který nyní neobsahuje použitý font, ale jen pokyn pro RIP, aby daný font realizoval ze svých vlastních zdrojů. Tento postup mimo jiné šetří naši peněženku. Nemusíme totiž uvedené fonty od Adobe kupovat dvakrát. Jednou jsme je zaplatili v ceně RIPu tiskárny a v počítači tento font principiálně nepotřebujeme. Pravda, toto „spoření“ nám může komplikovat život při náhledech dokumentu na monitoru. Algoritmům náhledu musíme sdělit prostřednictvím nějaké substituční tabulky, že originální fonty od Adobe v počítači nemáme a že je chceme pro účely náhledu nahradit podobnými třeba volně šířenými fonty. To se dá v inteligentních systémech pro přípravu sazby konfigurovat (mezi takové systémy patří samozřejmě distribuce TEXu). Přitom metrické údaje fontů jsou zadarmo a Adobe je zveřejňuje na svých WWW stránkách. Sázecí stroj necháme pracovat s originálními metrickými údaji, protože naším cílem je dokument nakonec vytisknout s originálními písmy od Adobe.

Další výhodou tohoto postupu je skutečnost, že výsledný PostScriptový kód našeho dokumentu, který neobsahuje downloadované fonty, můžeme s klidným svědomím vystavit na Internetu a posílat všem svým kamarádům a známým, aniž bychom tím porušili licenci k použitým fontům. Navíc při použití jen vyjmenovaných 35 fontů budeme mít jistotu, že RIP našich kamarádů se s chybějícími fonty správně vyrovná. O problému konfliktu s licencí k fontu při zasílání dokumentů se ještě zmíním později v souvislosti s tzv. *embeddingem* fontů.

Společnost Apple společně s Microsoftem vyvinula formát TrueType, který se v implementaci tvaru znaků výrazně liší. Rasterizér pro tento formát písma zařadily tyto společnosti přímo do jádra svého operačního systému a tím prosadily rozšíření tohoto formátu, se kterým si původně nerozuměl PostScript od Adobe. Dnešní tiskové ovladače generující PostScript používají na uvedený problém nekompatibility zvláštní fintu. Odkazuje-li dokument na font, který je v počítači přítomen jen ve formátu TrueType, „zabalí" tento font do obálky se speciálními pokyny pro PostScriptový RIP a tím vzniká tzv. font Type-42. V takovém stavu jej ovladače downloadují do výstupního PostScriptového kódu. Novější verze PostScriptových RIPů se s tím dokáží vyrovnat a fontům Type-42 rozumějí. Přesto se mnohdy stává, že dochází k nepříjemným problémům, které jsou důsledkem původní nekompatibility obou formátů.

Formáty dokumentů

S formátem fontů úzce souvisí i formát dokumentu. Nebudeme zde přesně vyjmenovávat a popisovat jednotlivé formáty, ale uvedeme jen základní principy a požadavky na formáty dokumentů.

Abychom byli přesnější, nebudeme od této chvíle mluvit o formátu dokumentu, ale o formátu konkrétní aplikace, která je určena na manipulaci s textem. Tato aplikace ukládá na disk informace o textu v domluvené datové struktuře případně si vyměňuje informace s jinými aplikacemi. Této struktuře říkáme *formát aplikace*.

Při návrhu formátu aplikace se většinou bere v úvahu účel a vlastnosti, které se od aplikace očekávají. Kdyby měl být tento formát používán jedinou aplikací, asi by nás ani moc její formát nezajímal. Středobodem veškerých problémů je ovšem skutečnost, že aplikace potřebuje kooperovat s ostatními aplikacemi zaměřenými na podobnou nebo návaznou činnost a přenášet informace mezi sebou a třeba i mezi různými platformami a architekturami. V takovém případě nás samozřejmě formát, ve kterém jsou tyto informace přenášeny, velmi zajímá.

Pokud má aplikace předávat informace jiným aplikacím třeba i od jiných výrobců, je nutností, aby existovala veřejná specifikace použitého formátu. Často jsme, bohužel, svědky právě opačného trendu, kdy veřejná specifikace formátu chybí. Asi je to pro výrobce aplikace marketingově výhodné. Uživatelé, kteří pak používají takové aplikace, se stávají dobrovolně zajatci svého softwarového dodavatele, protože veškeré jejich informace (například archivy firmy) jsou uloženy ve formátu, na jehož dešifrování může poskytnout software jediný dodavatel. Takové praktiky jsou odsouzeníhodné. Účastníci této konference, která má v názvu slovo „open", na to jistě budou mít podobný názor. Nadále tedy budeme uvažovat jen formáty s veřejnou specifikací.

V zásadě můžeme formáty aplikací pro zpracování textu rozdělit do dvou skupin: formáty, které obsahují informace, jež se objeví na vstupu do sázecího stroje a za druhé formát, který je výstupem sázecího stroje a čtou jej algoritmy náhledu nebo tiskové ovladače.

Formát vstupu sázecího stroje lze jazykově analyzovat (členění na slova, věty, odstavce) a v ideálně navrženém formátu lze rozpoznat i další logickou strukturu dokumentu (členění na kapitoly, podkapitoly, odkazy na obrázky, popisky obrázků, poznámky pod čarou apod.). V takovém formátu lze dodatečně měnit text dokumentu, což je například pro interaktivní textové procesory a DTP programy velmi důležité. Formát by neměl obsahovat informace týkající se konkrétního vzhledu sazby, tj. například jak velkou mezeru použít nad nadpisem a pod nadpisem, jaký použít font pro text a jaký pro nadpisy, jaké jsou rozměry zrcadla sazby apod. Tyto informace nesoucí požadavky na sazbu by v ideálním případě měly být druhým nezávislým vstupem do sázecího stroje.

Ideál se ale často liší od skutečnosti, protože často potřebujeme pracovat s jediným souborem, který po načtení do aplikace (a po spuštění sázecího stroje v aplikaci) zrealizuje náhled dokumentu ve stejné podobě, v jaké jsme jej viděli před uložením do souboru. Reálné formáty proto míchají logickou strukturu dokumentu s požadavky na vzhled sazby. Mnohdy je formát aplikace tak nevhodně navržen, že z něj už nelze jednoduše získat zpět logické a jazykové členění dokumentu, protože se spíše blíží formátu výstupu ze sázecího stroje než formátu vstupu. Jediné, co tyto nevhodně navržené aplikace dodržují, je možnost dodatečné modifikace textu. V dnešní době se naštěstí pomalu ustupuje od takto nevhodně navržených formátů. Mezi vhodné formáty patří například datové struktury založené na XML. Za jistých okolností je v tomto smyslu vhodným formátem i vstup pro TEX.

Formát vstupu sázecího stroje není načítán přímo sázecím strojem, ale nejprve scannerem tohoto formátu. V něm se interpretují konkrétní značky použité ve formátu, jako například značka pro přechod na další kapitolu nebo značky obklopující text určený pro poznámku pod čarou. V případě TEXu se takový scanner obvykle implementuje na úrovni jeho makrojazyka. Uživatel TEXu si tak může pro každý dokument definovat svůj vlastní specializovaný formát vstupu sázecího stroje. Konfigurace scanneru se obvykle připojují do doprovodného souboru společně s nastavením parametrů sazby, ale není to podmínkou.

Formát výstupu sázecího stroje obsahuje definitivní rozmístění jednotlivých elementů sazby na jednotlivých stránkách a není pro interaktivní aplikaci, která dovoluje modifikovat text, dále použitelný. Skvělým příkladem takového formátu může být `dvi` (výstup z TEXu) nebo PostScript. I interaktivní aplikace používají formát výstupu sázecího stroje při předání informace o dokumentu tiskovému ovladači. V některých špatně navržených aplikacích se ale setkáváme s tím, že určitou činnost, která je výsadou sázecího stroje, dělá tiskový ovladač znovu. Pak se dočkáme takových nepochopitelných záhad, jako je zcela jinak naformátovaná sazba stejného dokumentu na dvou různých tiskárnách. Takové příhody dokládají neprofesionalitu programátorů aplikace a ovladačů, kteří nebyli schopni implementovat sázecí stroj jen do aplikace a škrtnout vše, co se podobá sázecímu stroji, z algoritmů tiskového ovladače.

Formát výstupu sázecího stroje je velice obtížně použitelný ke zpětné jazykové analýze, mezi kterou v tuto chvíli řadím například i vyhledávání slov v textu nebo třeba přenos textu pomocí myšoidní operace „sejmi a vypusť" do jiné interaktivní aplikace. Některá slova jsou totiž rozdělena spojovníkem, protože sázecí stroj při výpočtu sazby takto rozhodl. Mezi dvojicemi vybraných znaků je vložena informace o dodatečném mezerování, protože tuto informaci přečetl sázecí stroj z metrických údajů fontu. Konečně některé skupiny znaků jsou nahrazeny jinými znaky podle tabulky ligatur.

Vidíme tedy, že je velice obtížné navrhnout ideální formát, který by udržoval jednak informaci o definitivním rozložení sazby a jednak dovoloval aspoň omezené manipulace s textem. Přitom na takový formát aspiruje PDF (Portable Document Format) navržený a prosazovaný firmou Adobe. Má se jednat o formát vstupu nebo výstupu sázecího stroje? Ideální pro potřeby dodatečné manipulace by bylo, kdyby se jednalo o formát vstupu sázecího stroje. K tomu ovšem by musel být sázecí stroj všech aplikací pracujících s textem jednou pro vždy normalizován a musel by se chovat naprosto stejně ve všech platformách. Tuto vlastnost má, pokud vím, jedině TEX. Nicméně, zatím stále nebyl komerčními subjekty vybrán jako normalizovaný sázecí stroj do všech aplikací. Když se navíc rozhlédneme kolem sebe a všimneme si různých, možná i záměrných, nekompatibilit produktů jednotlivých výrobců, dojdeme snadno k závěru, že tento ideál je zřejmě nesplnitelný.

Proto se u moderních formátů přistupuje k určitému kompromisu, který vede ke zdvojení některé informace ve formátu. Například rozdělené slovo je ve formátu zapsáno jednak ve formátu výstupu sázecího stroje a hned vedle jej najdeme (například pro potřeby vyhledávání) zapsáno dohromady a nerozděleně. Stejný trik je použit u slov, která obsahují ligatury. Dále je potřeba ve formátu rozlišit mezery, které jsou mezislovní, od mezer, které jsou důsledkem automatického vyrovnání podle tabulky kerningových párů. Ty druhé se při vyhledávání slov musejí ignorovat.

Kódování dokumentů

Kódováním dokumentu rozumíme úmluvu, jak budeme ve formátu dokumentu zanášet informaci o jednotlivých znacích textu. Aby měly algoritmy pro náhled a tiskové ovladače zjednodušenou práci, je tato datová reprezentace většinou volena tak, aby to nějak navazovalo na datovou reprezentaci znaku ve fontu. Jak ukážeme později, není to ovšem podmínkou. Jednotlivé znaky použité v dokumentu můžeme zanášet do formátu pomocí slovních výrazů (např. Ecaron, ocircumflex apod.), pomocí speciálních sekvencí (např. TEXové sekvence pro akcentované znaky), ale zdaleka nejčastější způsob kódování je zanesení znaků do dokumentu jako čísla, tzv. *kódy znaků*. Je to totiž nejúspornější a aspoň pro písmena anglické abecedy se to považuje za samozřejmé. Za dobu vývoje počítačů se ustálilo ASCII kódování 92 viditelných znaků obsahujících písmena latinky bez akcentů a několik diakritických znaků. Kódy těchto znaků jsou vždy menší než 128, takže k uložení do paměti vyžadují jen 7 bitů. Osmý bit byl kdysi využit jako kontrolní nebo rezervní. Konkurenční kódování EBDIC se naštěstí neujalo.

Dnes bohužel existuje mnoho návrhů na kódování české abecedy. Tyto různé a nekompatibilní návrhy jsou vesměs založeny na jednobajtovém kódování (všechny kódy jsou menší než 256). Jejich vzájemná nekompatibilita uživatelům stále komplikuje život při přenosu dokumentu z jedné aplikace do druhé nebo dokonce při přenosu dokumentu mezi platformami. Dnes se stále častěji v aplikacích používá kódování, které obsahuje kódy větší než 256, takže aspoň některé znaky jsou zaneseny ve dvou nebo i více bajtech. Mezi těmito kódováními stojí za zmínku tzv. Unicode, který se snaží deklarovat jednotné kódy pro písmena abeced všech jazyků světa a je navržen důsledně jako dvoubajtové kódování.

Je pochopitelné, že pokud mají aplikace mít mezi sebou vyměnitelný formát dokumentu, musejí se všechny shodnout na společném kódování. To se, zvláště v případě české a slovenské abecedy, dlouho nedařilo a dosud nedaří. Jediným řešením je asi možnost, že se snad v budoucnu všechny aplikace shodnou na Unicode. Tento standard není žádnou novinkou, ale jeho prosazení je v anglicky mluvících zemích velmi obtížné. Nikdo dobrovolně nezdvojnásobí velikost své firemní databáze, ve které používá skoro výhradně písmena anglické abecedy. Argument, že pak bude možné nekonfliktně používat češtinu s turečtinou v anglicky mluvících zemích moc neobstojí.

Je třeba si uvědomit, že kódování dokumentu souvisí s formátem fontů jen okrajově. Dá se totiž provádět konverze až za výstupem ze sázecího stroje tak, aby bylo vše v souladu s implementací použitého fontu. Příkladem takového postupu mohou být virtuální skripty TEXu, které mohou tvořit (mimo jiné) překódovací rozhraní mezi interním kódováním textu v TEXu a kódováním vybraného fontu. Tyto virtuální skripty jsou samozřejmě respektovány jednak algoritmy pro náhled a jednak tiskovými ovladači, takže vše funguje bez problémů. Jediný TEXový dokument tedy může obsahovat odkazy na fonty rozličně kódované a pokud tyto fonty mají společnou sadu znaků použitou v dokumentu, pak stačí zvolit pro tyto znaky libovolné kódování, se kterým bude pracovat sázecí stroj. Rozdíly v kódováních fontů lze pak kompenzovat na úrovni virtuálních skriptů. Tuto geniální možnost TEXu jsem v jiných systémech na zpracování textu nenašel.

Také je možné použít fonty, které jsou na kódování nezávislé, protože implementují jednotlivé znaky výhradně podle názvů. Příkladem mohou být fonty ve formátu PostScript Type-1. Při práci s těmito fonty stačí „lovit“ jednotlivé znaky do sazby podle jejich názvů.

Rozhraní mezi aplikací a fontem

Většina aplikací se při práci s fonty obrací na služby hostitelského operačního systému. TEX je v tomto smyslu výjimkou, protože se jednak distribuuje společně se sadou Computer Modern fontů v TEXovsky specifickém formátu, které jsou zcela nezávislé na hostitelském operačním systému, a jednak TEXové ovladače dokáží obejít služby systému a pracují přímo s fonty rozličných dalších formátů.

Pokud aplikace využívá při práci s fontem služeb systému, pak „loví“ jednotlivé znaky z fontu tak, že systému sdělí kód použitého znaku a obdrží od něj metrické údaje nebo bitovou mapu znaku. Při komunikaci se systémem není možno vyzvednout z fontu znak podle jména, ačkoli font může být na takovou možnost stavěný. Operační systém striktně vyžaduje kód znaku. Tím se výrazně omezují možnosti například formátu fontu Type-1 v takových aplikacích.

Vysvětlíme si, jakou roli hraje ATM (Adobe Type manager). Operační systémy MS Windows a MAC umějí nativně pracovat jen s formátem TrueType. PostScriptovému formátu Type-1 nerozumějí. Pro tyto systémy existuje berlička v podobě ATM. Pokud se aplikace obrátí na systém s požadavkem na rastr nebo metrický údaj znaku z fontu Type-1, systém se obrátí na ATM se stejným dotazem, ATM znak rasterizuje nebo připraví informace o metrickém údaji a výsledek předá systému. Ten jej pak předá aplikaci.

Uvedu pro srovnání ještě možnosti, které při práci s fonty nabízejí UNIXové systémy. Přesněji bych měl hovořit o Xserveru, který je na těchto systémech pro práci v grafice vesměs používán. I v nejstarších Xserverech byla implementována možnost práce s fonty formátu Type-1 přímo bez použití berličky podobné ATM. Nebyla ovšem možnost pracovat s formáty TrueType. Teprve v novějších systémech existuje vylepšená verze font serveru, která hraje podobnou úlohu jako ATM a rozumí navíc formátu TrueType. Jinak řečeno, při spolupráci s font serverem umí Xserver pracovat i s fonty formátu TrueType. Jeden font server navíc dokáže obsloužit požadavky mnoha Xserverů po TCP/IP síti. To je schopnost, o které jsem v případě MS Windows nebo

MAC neslyšel. Nejnovější verze Xserverů umějí pracovat s Type-1 i TrueType nativně. Kvalitu rasterizéru pro nízká rozlišení jsem osobně neověřoval, protože s fonty TrueType vůbec nepracuji.

Všechny systémy umějí pracovat s různými bitmapovými formáty fontů, které se používají pro texty nabídek, výpisy logů apod. Existuje bohužel velmi mnoho formátů bitmapových fontů, které nejsou snadno přenositelné ze systému do systému. Při srovnávání možností programů pro přípravu elektronické sazby nejsou samozřejmě bitmapové fonty předmětem našeho zájmu.

Popis a rozdíly základních formátů fontů

V této části nastíním základní principy formátů METAFONT, TrueType, Type-1, Multiple-Master a OpenType a pokusím se o srovnání. Podrobněji vyložím formát Type-1 včetně ukázek na počítači.

Pokusím se vyvrátit falešné představy o tom, že formát Type-1 může efektivně pracovat maximálně s 256 znaky, že je závislý na kódování a že tento formát nemá možnost k podchycení záznamu o proměně skupiny znaků v ligaturu.

K této části přednášky není připraven žádný text, aby se účastník konference mohl na něco těšit a neměl vše hned připraveno ve sborníku jako na talíři.

O problematice licencí fontů

Není pochyb o tom, že písma implementovaná do počítače nebudou vždy šířena zdarma, ačkoli i taková písma existují. Na fonty je třeba se spíše dívat jako na umělecká díla. S písmy v počítači se ale neobchoduje stejně jako s obrazy nebo sochami, ale spíše jako se softwarem. Texty licencí k písmům tedy hovoří o software, nebo dokonce o programu, což je zboží, se kterým se v souvislosti s počítači obchoduje. Právě z licenčních důvodů se ve specifikacích formátu fontů hovoří o fontech někdy až nepřirozeně jako o programech.

Většina komerčních licenčních ujednání k softwaru je z technického pohledu na věc obtížně interpretovatelná. Na některé nejasnosti se zde pokusím upozornit. Licence k písmům se navíc potýkají s dalším problémem, který je typický jen pro tento druh software. I tyto problémy se pokusím rozebrat v následujícím textu.

V komerčních licencích poskytovatel software dává zákazníkovi možnost nějakým omezeným způsobem fonty použít. Co to ale znamená „použití" fontů? Je to okamžik, kdy je potřeba použít jejich metrické údaje k výpočtu sazby? Domnívám se, že rozhodně ne. Metrické údaje k fontům jsou volně zveřejněny třeba i u takových firem, jako je Adobe. Kdokoli je může použít a teoreticky vypočítat rozložení sazby na stránkách. Když nemáme vlastní popisy tvaru jednotlivých znaků, pak takový dokument stejně nezrealizujeme do konečné podoby.

Jiná interpretace slova „použití" fontů by mohla tvrdit, že se jedná o okamžik downloadování popisů tvarů znaků do dokumentu. To se asi nejvíce blíží pojetí většiny licencí na fonty. Je ovšem důležité se vyvarovat možnosti volně zveřejnit na Internetu nebo někomu poskytnout dokument s úplnými downloadovanými fonty, protože by je kdokoli trochu více vládnoucí počítačem mohl z tohoto dokumentu vykostit a začít nelegálně „používat". V tom případě by měl podle mého názoru na takovém stavu vinu i ten, kdo dokument zveřejnil, protože on tomu pirátovi umožnil jednoduše krádež. Právníci by se na takovém případě asi vydováděli.

Tiskové ovladače generující PostScriptový kód, který je nejčastějším nositelem downloadovaných fontů, by tedy měly umět vložit font ve tvaru označovaném jako „embedding subset". Znamená to, že font není downloadován celý, ale jen ty znaky, které jsou v dokumentu použity. Některé licence se ale brání i takovému způsobu poskytování dokumentu dalším osobám, protože nemají jistotu, zda autor například nepoužil ve svém dokumentu tabulku všech znaků fontu. V takovém případě je embedding subset dost k ničemu.

Teoreticky je možné downloadovaný font šifrovat, ale pokud mají existovat veřejné specifikace použitých formátů, musí být veřejně znám i klíč pro rozšifrování takových fontů. Tuto slepou uličku už dávno opustili ve firmě Adobe, když dodatečně zveřejnili specifikaci formátu Type-1 včetně „hesla" šifry, které je dnes všeobecně známé. Tím heslem je číslo 5839.

Pro přenos běžných dokumentů, kde tolik nezáleží na volbě konkrétního písma, je tedy vhodné používat takové formáty dokumentů, ve kterých není nutné fonty downloadovat. Stačí tam ponechat jen odkaz na font. Pokud se výrobci software shodnou na základní skupině fontů podporovaných na všech operačních systémech a ve všech RIPech, přenos aspoň jednoduchých dokumentů by byl vyřešen. Mezi takovou základní skupinu by mohla patřit třeba skupina 35 fontů dostupná v každém PostScriptovém RIPu. Bohužel, výrobci software pro různé platformy se zatím nedohodli.

Důsledkem toho je, že zasílání dokumentů s nedownloadovanými fonty je velmi problematické a při přenosu na jinou platformu skoro jistě dojde k problémům.

Vidíme tedy, že pokud licence k písmům omezuje možnost poskytování písma dalším osobám (což je obvyklé), pak bohužel neexistuje technicky dokonalé řešení, jak jim předávat dokumenty, ve kterých jsou taková písma použita.

Vraťme se znovu k možným výkladům slova „použití“ fontů. Typicky nepočítačový, ale jistě správný, výklad slova „použití“ písma je okamžik, kdy text čte čtenář. Pokud má třeba Vieweghova kniha náklad několika set tisíc výtisků, pak je „použití“ písma v takové knize řekl bych skoro lidové. To licence většinou vůbec neřeší, ale možná by měla.

Dalším výkladem slova „použití“ fontu by mohl být okamžik, kdy je třeba vytvořit tvary znaků podle jejich matematických popisů. To dělá rasterizér fontu v osvitových jednotkách, v tiskárnách a při náhledech též přímo v počítači. Když vytvoříme dokument s downloadovanými fonty a pošleme jej na některé z uvedených zařízení, teprve pak se fonty v tomto smyslu „použijí“. Pokud licence omezuje použití fontů na limitovaný počet zařízení, museli bychom do tohoto počtu zahrnout i všechny tiskárny s RIPem, na kterých někdy budeme chtít dokument vytisknout. Navíc by to vyloučilo i možnost předávání dokumentu dalším osobám s fonty ve tvaru embedded subset.

Na druhé straně by to dovolilo instalovat fonty do UNIXového font serveru. Rastrování by pak probíhalo v procesoru jediného počítače, ale fonty by byly k dispozici na neomezeném počtu UNIXových pracovišť. Každé pracoviště by totiž od font serveru dostalo po síti vždy takovou bitmapu, jakou zrovna pro náhled dokumentu potřebuje.

Otázka možnosti „použití“ jakéhokoli softwaru (nejen fontů) na limitovaném počtu počítačů je vždy problematická, pokud vezmeme v úvahu propojení počítačů do sítě. Tam můžeme software reálně instalovat jen na jediném počítači a ostatní pracoviště mohou sloužit jako grafické terminály k tomuto počítači. Z technického pohledu software pracuje pouze na jediném počítači (protože jedničky a nuly takového software se odehrávají v elektrických obvodech tohoto jediného počítače) a ke koncovému pracovišti je přiveden po síti jen výsledný obraz dané aplikace. Takových pracovišť sloužících jako grafické terminály ovšem může být k centrálnímu počítači připojeno libovolně mnoho, takže na jediném počítači současně může pracovat velmi mnoho uživatelů. Existenci takových multiuživatelských počítačů většina softwarových licencí vůbec neřeší. Asi proto, že právníky, kteří texty licencí navrhují, takové technické řešení vůbec nenapadlo. Oni možná jsou schopni si pod pojmem počítač představit jen osobní PC, na kterém v jednu chvíli může pracovat jen jeden člověk.

Literatura

– Adobe Systems Incorporated. *Adobe Type-1 Font Format—Version 1.1*. Addison-Wesley, Reading, MA, USA, August 1990. ISBN 0-201-57044-0. iii + 103 pp. US$14.95. Dostupné i elektronicky na `www.adobe.com`.
– Adobe Systems Incorporated. *PostScript Language Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1990. ISBN 0-201-18127-4. viii + 764 pp.
– Donald E. Knuth. *The METAFONTbook*, volume C of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13445-4. xi + 361 pp.