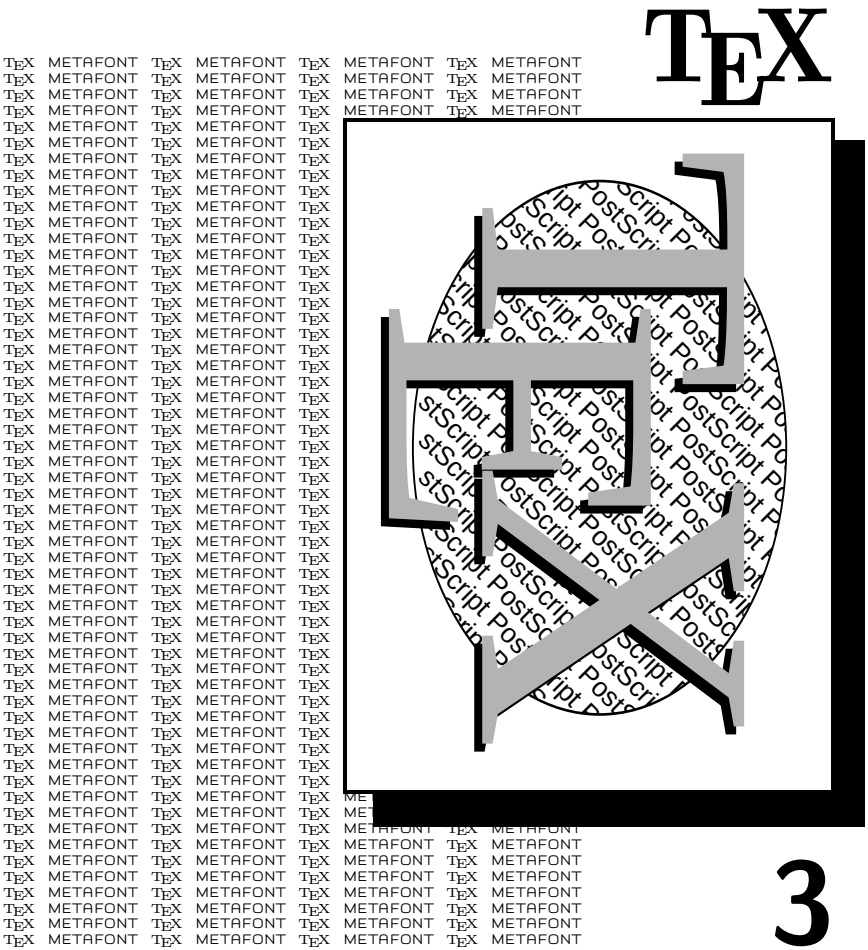


3

ZPRAVODAJ 93

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

OBSAH

Petr Olšák: Jak TEX pracuje s PostScriptem 101

Jiří Rybička: Novou instalaci TEXu? 116

Petr Olšák: Úvaha o fontech v CSTEXu 121

Josef Krob: Několik poznámek ke spuštění TEXu (tex386) pod Windows (DPMI). 131

Zdeněk Wagner: Konverze Word Perfect ⟹ LATEX 135

Jiří Tlach: Několik poznámek z tiskařské praxe 138

Pavel Sekanina: Pozdrav z Aljašky 140

Philip Taylor: Report on the Inaugural Meeting of the NTS Core Group: September, 1993 143

Literatura 147

Část archivu CSTUGu v `ftp.muni.cz` pod `/pub/tex` byla z kapacitních důvodů přesunuta na jiný (rychlejší) počítač a jiný (rychlejší) disk. (Alias `ftp.muni.cz` zůstává, ale ten kdo by snad používal přímo číselnou IP adresu by mohl být časem překvapen). OS je SunOS (unix). Z uživatelského hlediska všechny funkce zůstávají zachovány, včetně struktury adresářů [nebo by alespoň měly `:-)`]. Používaný komprimační program je `INFO-zip` (tj. NE `PKzip`). Najdete ho v `pub/msdos/simtel/zip/unz*` (`.exe` pro MS DOS) či v `pub/muni.cz/msdos/zip*` (zdroje). Případné problémy s TEXovou částí archivu ventilujte na adrese `tex-support@muni.cz`, s funkčností `ftp` se obracejte na `ftp-admin@dcs.muni.cz`.

The preparation of TEX Version 3 and MF Version 2 has taken me much longer than expected, but at last I've been able to look closely at the concept of virtual fonts. The need for such fonts is indeed much greater now than it was before, because TEX's new multilingual capabilities are significantly more powerful only when suitable fonts are available.

Donald Ervin Knuth: *Virtual Fonts: More Fun for Grand Wizards.* TUGboat, Volume 11 (1990), No. 1

Jak TEX pracuje s PostScriptem

Petr Olšák

PostScript je dnes jeden z nejpoužívanějších jazyků pro popis stránek a je jednou z mála alternativ, jak získat sazbu na profesionální úrovni. Většina kvalitních tiskáren a osvitových jednotek pracuje pouze v tomto formátu. V tomto článku probereme vazby a softwarové možnosti mezi tímto jazykem a TEXem.

Zmíníme se též o jedné velmi silné možnosti práce s fonty v TEXu – o tak zvaných virtuálních fontech, které autor TEXu uvedl svým známým článkem v TUGboatu z roku 1990 (viz citát v záhlaví článku).

PostScript

Protože použití TEXu s výstupem do PostScriptu sčítá možnosti obou softwarových produktů, které se vzájemně nevylučují, bude užitečné se nejdříve věnovat podrobněji vlastnostem PostScriptu. Odborníci mi snad prominou mé laické vysvětlování.

Snad každé výstupní tiskové zařízení z počítače s výjimkou plotterů je založeno na myšlence obarvit nějak výstupní plochu obdélníkového tvaru (většinou stránku papíru), přičemž toto zařízení si danou plochu rozdělí

na síť bodů (rastr) a redukuje svůj úkol na vyplňování některých zvolených bodů barvou. Toto vyplňování lze na některých zařízeních přímo řídit počítačem. Například když posíláme na jehličkovou tiskárnu grafický obraz stránky, pak software v našem počítači sám řídí, který bod se má vyčernit a který ne. Pokud ale posíláme na tutéž tiskárnu soubor v textovém režimu, tiskárna přečte jednu řádku textu a rozhodne sama, jaké body vybarví.

Na podobném principu, ale podstatně vyšší úrovni jsou založena tisková zařízení vybavená PostScriptem. V počítači připravíme textový soubor – tak zvaný PostScriptový popis stránek, nebo též PostScriptový program. Takový soubor pošleme tiskovému zařízení ke zpracování. Zařízení je vybaveno interpretem, který čte náš textový soubor a na základě přečtených informací zpracovává pomyslný obraz stránky (image) v dané hustotě bodů (rastr), a v okamžiku, kdy narazí v PostScriptovém programu na příkaz konce stránky, většinou spustí vlastní tisk stránky podle vytvořeného obrazu. Tento interpret je součástí softwarového vybavení výstupního zařízení, takže výpočet obrazu stránky probíhá až na místě, kde se stránka tiskne, a nikoli v našem počítači. Příslušnému interpretu se říká RIP (Raster Image Processor).

Jazyk PostScriptu definovala firma Adobe. Popis jazyka najdeme v knize

PostScript Language Reference Manual, "The Red Book", Adobe Systems Incorporated, 2nd ed., Addison Wesley 1990.

Tato kniha stojí zhruba $ 28 a její druhé vydání popisuje tzv. PostScript Level 2, který je výrazným rozšířením původního návrhu (především v oblasti práce s barvou). Kromě této referenční knihy existují knihy dalších „barev", např. *Orange Book* je určena pro profesionální použití, zatímco *Blue Book* je spíš úvodem a *The Black Book* specifikuje smluvený formát fontů Type 1. Samozřejmě, mnoho bylo napsáno i v jiných knihách.

Jazyk PostScriptu je (zhruba řečeno) nezávislý na výstupním zařízení. Znamená to, že v našem PostScriptovém popisu stránek se nemusíme starat o technické záležitosti výstupního zařízení (např. rozlišovací schopnost). To je věcí konkrétní implementace PostScriptového RIPu. PostScript tedy vytváří jakési rozhraní mezi hardwarovým prostředím tiskového zařízení a softwarovým prostředím našeho počítače.

Uvedená nezávislost ovšem nemůže být absolutní. PostScriptový program je schopen např. testovat konkrétní rozlišení a na základě toho vět-

vit svůj výpočet. Nebo jiná záludnost: napíšeme-li v PostScriptu třeba `0.5 setgray`, dali jsme tím najevo, že další objekty budou kladeny do obrazu stránky v šedé barvě někde uprostřed mezi černou a bílou, ale naprosto nevíme, co s tímto požadavkem udělá konkrétní RIP. Na nízkém rozlišení dostaneme třeba puntíkované plochy typu „co puntík, to deset mil“ a může to naprosto zhatit náš výtvarný záměr, zatímco na vysokém rozlišení to dopadne většinou dobře. Kdo tedy mluví o nezávislosti PostScriptu na zařízení, má na mysli nezávislost z jakéhosi subjektivního optického hlediska (a to jen tehdy, když se jeho PostScriptový program nevětví podle hardwarových parametrů výstupního zařízení). Přitom matematicky nelze snadno definovat, co v této věci znamená nezávislost na zařízení.

Zastavme se u možností, jaké nám programování v PostScriptu nabízí. V této souvislosti se přiznám ke své jisté profesionální deformaci. Když čtu reklamy, většinou nečtu jejich obsah, ale hodnotím typografickou úroveň a v pozadí jakéhokoli grafického triku vidím konkrétní PostScriptový příkaz, kterým to bylo zařízeno. Nicméně návrháři těchto „písmeno-obrázků“ většinou nepracují s PostScriptem přímo, ale prostřednictvím nějakého uživatelsky přítulného rozhraní, jakým je třeba CorelDraw!, či podobné programy. Jinými slovy, aby uživatel vytvořil PostScriptový program, vůbec nemusí umět PostScript. A nyní už k možnostem PostScriptu.

- Pracuje se v souřadnicích s absolutním měřítkem (např. centimetry, tiskařské body apod.), které jsou nezávislé na výstupním rastru. Naše měřítko lze navíc lineárně transformovat (rotace, posunutí a zvětšení). PostScript totiž pracuje s tzv. transformační maticí, pomocí níž vše, co my formulujeme v našich pracovních souřadnicích, transformuje do skutečných souřadnic rastru. Tím lze dosáhnout různé rotace a deformace písmen.
- V každém okamžiku se kromě změny podle transformační matice všechny objekty předzpracovávají na „barvu“ podle tzv. grafického stavu. Můžeme nastavit například červenou a pak vše, co bude dále kladeno do obrazu stránky bude červené. Jak to je technicky zařízeno, to je záležitostí konkrétního RIPu (např. se provede barevná separace nebo se objekt vytiskne v jistém stupni šedi).
- Jsme schopni formulovat jisté abstraktní tahy podél úseček, částí kružnic nebo Beziérových křivek, tedy objektů, které jsou matematicky popsány několika málo souřadnicemi či reálnými čísly. Je

to podobné příkazu „path“ v METAFONTu. Tento tah může tvořit hranici abstraktního objektu, který dostane konkrétní podobu až po přepočítání do rastrového obrazu RIPem. Také lze definovat tloušťku čáry a požadovat, aby RIP vytvořil objekt typu „čára dané tloušťky, jejímž středem vede zadaný abstraktní tah“. Na rozdíl od METAFONTu nelze definovat tvar pera. Konce čar jsou buď hranaté nebo zaoblené.

- Lze požádat o vykreslení tvaru, který je definovaný bitmapově. Zde ovšem pozor! Věrný obraz bitmapy dostaneme pouze tehdy, je-li zrovna transformační matice rovna jednotkové matici. Za jiných okolností RIP konvertuje podle transformační matice (a celého grafického stavu) příslušnou bitmapu na jinou bitmapu, což nedopadá většinou dobře. Proto se bitmapové obrázky s nízkým rozlišením skládají z takových legračních čtverečků, které jsou viditelné pouhým okem a nepomůže ani vysoké rozlišení finálního výstupu.
- PostScript je vybaven podobným (ovšem poněkud chudším) makrojazykem jako TEX. O vlastnostech PostScriptu se v této věci nebudu pro nedostatek místa rozepisovat.
- Jazyk PostScriptu je vybaven mechanismem práce s fonty. Tyto fonty mohou být definovány jakkoli tzv. rutinou fontu, která je popsána PostScriptovými příkazy. Jde o to, že pokud se např. napíše do PostScriptu string `(AHOJ) show`, je PostScript schopen postupně zpracovat písmena A, H atd. rutinou zvoleného fontu, jejíž výsledkem je (obvykle) grafický objekt na stránce. Tyto rutiny většinou pracují podle principu abstraktních tahů (viz výše), ale mohou též existovat fonty, jejichž rutiny pracují způsobem přenesení předem daných bitmap. První případ je častější – jedná se o tzv. „outline“ PostScriptové fonty, ovšem druhý případ nelze přehlédnout, protože to je jedna z možností, jak dostat TEXovské fonty, které nejsou přístupné v PostScriptové podobě, do PostScriptu.
- Absolutní překrývání. RIP si před tiskem drží obsah stránky v bitmapové podobě v paměti a jeho úkolem je zcela překrýt předchozí grafické objekty novými, třebaže mají světlejší barvu. Toho například využívá program *Mathematica* při kresbě svých prostorových obrázků. Program vůbec neřeší problém viditelnosti, ale prostě kreslí PostScriptem „odzadu dopředu“ a tím přední plošky zakryjí ty zadní. Také velmi hezké stínování obrázků je z 90% zásluhou PostScriptu, který míchá barvy podle úhlu dopadu světelného paprsku na plošku sám. Tady vidíme odpověď na otázku, proč pro-

gram *Mathematica* nepodporuje jiný grafický formát. Obrázky z tohoto programu si mohl čtenář prohlédnout v prvním čísle bulletinu z r. 1993.

Neuvedl jsem zdaleka všechny možnosti, ale je na čase přejít k věci.

TEX a dvips

Ačkoli je TEX a METAFONT podstatně starší než PostScript, existuje velké množství možností, jak z hlediska TEXu pracovat s PostScriptem. Přitom vlastní program TEX nebyl kvůli tomu vůbec měněn. I v tom je možno spatřovat obrovskou životnost programu a genialitu autora. Navíc pro uživatele TEXu je PostScript pouze jednou z dalších možností výstupu z TEXu, ale sazba v TEXu vůbec není na této možnosti životně závislá.

Postupně si naznačíme nejčastější úkoly, které je nutno řešit v TEXu v souvislosti s PostScriptem. Začněme tím nejběžnějším požadavkem. Máme sazbu připravenou v TEXu (za použití klasických fontů používaných v TEXu) a chtěli bychom ji tisknout na zařízení, které se s námi baví pouze v PostScriptu. V takové situaci použijeme ovladač `dvips`, který transformuje výstup z TEXu (`dvi`) do PostScriptového popisu stránek. Jedná se o volně šířený program, vyrobený Tomem Rokickim (Stanford University). Část tohoto balíku (bez virtuálních fontů) je zařazena do CSTEXu. Ovladač čte `dvi` soubor a bitmapy fontů ve tvaru `pk` nebo `fli` a vytváří PostScriptový soubor, případně výstup rovnou posílá na PostScriptovou tiskárnu. V tomto PostScriptu jsou fonty zapsány v bitmapové podobě a fontový mechanismus PostScriptu pracuje metodou přepisování bitmap. Ve výstupním PostScriptovém souboru proto najdete nejprve hlavičky maker, dále hexadecimální popis bitmap použitých fontů a nakonec vlastní text, který je víceméně nečitelný, protože jsou často mezi jednotlivá písmena vkládány kerningové informace. Taky akcentované znaky jsou přepsány do čísla kódu.

Tady určitě tušíte zradu. Programu `dvips` je nutno v parametrech říci, v jakém rozlišení má soubor `dvi` transformovat. Program podle tohoto údaje použije bitmapy fontů daného rozlišení. Výsledný PostScriptový soubor tedy není zcela 100 % nezávislý na výstupním zařízení. Nejvýhodnější by totiž bylo, kdyby příslušná transformační matice byla jednotková. Bitmapy fontů by pak nebyly RIPem následně znova přepočítávány a nedocházelo by k dalším chybám typu „plus minus pixel“. Pro dosa-

žení nejvyšší kvality je proto velice důležité vědět, v jakém rozlišení bude pracovat výstupní zařízení.

Program `dvips` tedy předpokládá, že dané rozlišení je definitivní a proto změní nejprve pracovní souřadnice PostScriptu tak, aby jedna jednotka odpovídala jednomu pixelu. Transformační matice pak bude (v případě, že bude rozlišení dodrženo) jednotková. Navíc program `dvips` zaokrouhluje rozměrové údaje z absolutně přesného `dvi` souboru na jednotky pixelů výstupního zařízení. Takže v PostScriptovém výstupu jsou pro rozměry použita pouze celá čísla (násobky pixelů). Desetinná čísla jsou sice v PostScriptu také možná, ale těm program `dvips` nevěří, protože například „Red Book“ praví, že reprezentace těchto čísel je závislá na implementaci. (To je další argument proti zastáncům názoru, že PostScript je nezávislý na zařízení.)

Když jsem se dozvěděl o možnosti sazby svého seriálu „Kapitoly o TEXu“ (Softwarové noviny) na osvitové jednotce, musel jsem nejprve zjistit, v jakém rozlišení mi to budou dělat. Jednalo se o rozlišení 1 270 dpi. Pak jsem doma METAFONTem připravil soubory `pk` s tímto rozlišením a programem `dvips` jsem *v tomto rozlišení* připravil soubor `ps`, který jsem zaslal redakci jako soubor pro osvit. Paradoxní na věci je, že takto kvalitní sazbu lze připravit na hloupém AT-čku (a teoreticky i na XT-čku), takže není zcela pravidlem, že k přípravě kvalitní sazby potřebujeme mocné stroje. Navíc takto připravené fonty jsou předem v počítači digitalizovány METAFONTem, a proto se RIP v osvitce už nezdržuje digitalizováním žádných outline fontů a osvit probíhá svižně. Kdo platí za strojový čas osvitové jednotky, pro toho je tento argument velice důležitý. Digitalizace „outline“ fontů RIPem totiž trvá dosti dlouho. Navíc se zákazník nemusí starat, zda požadovaný font v osvitce mají či nikoli.

Ještě jedna zajímavost, která není zanedbatelná. Jsou schopni uživatelé komerčních DTP programů vidět na obrazovce přesně a nefalšovaně to, co bude sázeno v osvitce? Pokud se programy opírají o „outline“ fonty, pak to technicky není možné. Takže WYSIWYG je vlastně lež. Pokud ale pracujeme s TEXem, pak jsme schopni vidět na obrazovce *pixel per pixel* totéž, co se bude svítit. A zas nám k tomu stačí obyčejné AT-čko. Bitové mapy připravené METAFONTem do tohoto rozlišení lze prostě nabídnout ke čtení prohlížeči `dviscr`. Pak například při zvětšení jedna ku jedné a rozlišení 1 270 dpi jsou na obrazovce VGA vidět zhruba dva řádky obrovského textu, který na šířku zabírá asi šest písmen. S výřezem lze samozřejmě posunovat a pixel na obrazovce se skutečně rovná

pixelu v osvitce. Při zmenšení 10×10 pixelů osvitky na jeden pixel obrazovky se už text dá na obrazovce docela dobře číst. Je to zajímavý zážitek (vynikne krása Computer Modern fontů), ale prakticky to asi není příliš použitelné.

Virtuální fonty

Věnujme se nyní trochu jiné úloze. Chceme zařadit do TEXu běžná PostScriptová písma jako například Times, Helvetica apod. Výstup chceme realizovat v PostScriptu, přičemž chceme použít pravé „outline“ fonty. Máme-li kompletní balík programu `dvips`, najdeme tam TEXovské metriky pro nejběžnější písma už připravené. Například `ptmr.tfm` je font Times-Roman (`p`: PostScriptový, `tm`: Times-Roman, `r`: Regular weight). V TEXu nám tedy stačí napsat

```
\font\muj=ptmr at 10pt
\muj Zde je text ve fontu Times-Roman
\bye
```

a TEX začne sázet v tomto fontu. Uvědomme si, že program TEX pracuje pouze s pomyslnými boxy. Jeho předmětem zájmu vůbec není způsob, jakým dostanou nakonec jednotlivé znaky ve fontu svůj tvar. V `dvi` souboru jsou tedy odkazy na font `ptmr`. Nyní použijeme program `dvips`, který nenajde font `ptmr` v souborech typu `pk` ani `fli`, ale najde soubor `ptmr.vf`, který je také součástí balíku programu. Z tohoto souboru si přečte, co má s uvedeným odkazem na font dělat. Zjistí z něj, že se jedná o font `Times-Roman`, provede případné překódování (například ligatury jsou ve fontu `Times-Roman` na poněkud jiné pozici, než s nimi pracuje TEX) a do PostScriptového výstupu vloží příkaz na zavedení klasického PostScriptového (outline) fontu, jehož znaky jsou pak digitalizovány RIPem až v okamžiku tisku.

Na našem jednoduchém příkladě to funguje, ale holá skutečnost je poněkud komplikovanější. Zatím nebylo nikde řečeno, jak takové soubory `tfm` a `vf` vyrobit v případě, že máme k dispozici nějaký méně běžný PostScriptový font, pro který zmíněné soubory v balíku `dvips` nenajdeme. Taky jsem neprozradil, že to nebude fungovat s háčky a čárkami, pouze jsem schválně zvolil takovou ukázku, kde tyto akcenty nejsou.

K tomu, abychom mohli mluvit o možnostech, jak tyto problémy řešit, musím aspoň zhruba nastínit myšlenku virtuálních fontů. Protože se ale jedná o „Další radosti pro velké kouzelníky“ (viz citát v záhlaví

článku), a přitom bych chtěl být stručný, bude se mi tato problematika vysvětlovat velice těžko.

Soubor `vf` je tak zvaný „virtuální font“. Dnes již téměř každý ovladač `dvi` souboru umí v případě, že nenajde font v souborech `pk` nebo `fli`, hledat i v souborech `vf`.[1]) Tam jsou uloženy informace o tom, jakým způsobem se má každá pozice fontu (neboli každý znak) realizovat. Nejběžnějším případem je vytáhnout znak z jiného `pk` fontu, třeba i z jiné pozice. Nebo lze znak realizovat složením více znaků k sobě, přitom každý z těchto elementárních znaků může být „vytažen“ z jiného `pk` souboru, nebo může být realizován jiným způsobem. Takto lze například zařídit sazbu akcentovaných písmen. V neposlední řadě může být znak realizován jako posloupnost příkazů nějakého jazyka, který umí příslušný ovladač zpracovat (podobně jako v TEXu příkaz `\special`). V případě ovladače `dvips` může tedy být takovým virtuálním znakem posloupnost PostScriptových příkazů.

Tato myšlenka virtuálních fontů umožňuje nejen využít plně jazyka výstupního zařízení (v našem případě tedy PostScriptu), ale umožňuje překódovat font do libovolného kódování srozumitelného pro výstupní zařízení a sestavovat písmena z dílčích elementů jako třeba akcenty. Uvědomme si, že tento mechanismus se nemusí použít jenom při práci s PostScriptem, ale je obecný.

Základní myšlenka je formulována a dále už jsou zapotřebí jen jednoduché konverzní rutiny. Především formát `vf` je binární, a tedy lidsky nečitelný. Proto Knuth definoval ve svém článku konverzní rutiny `VFtoVP` a `VPtoVF`, které konvertují soubor `vf` do formátu `VPL` (virtual property list) a zpět. Tento formát je čitelný textový soubor s přesně definovanou syntaxí. Zde se může TEXový kouzelník vyřádit. V souboru `VPL` jsou kromě informací potřebných pro `vf` formát uloženy také informace pro metriky `tfm`. Dají se zde tedy definovat nejen akce, co provést s každým písmenem na úrovni `vf`, ale také lze zadat kerningové páry, tabulky ligatur, velikosti boxů pro jednotlivá písmena a plno dalších věcí. Rutina `VPtoVF` pak nevytváří jen soubor `vf`, ale též `tfm`.

Dále existuje rutina `afm2tfm`, která konvertuje PostScriptové metriky `afm` (Adobe font metric) do TEXových metrik `tfm` a navíc vytváří „virtual property list“ pro možné další změny a úpravy. Tyto úpravy jsou

[1]) Nevím, jaký je obecný postup při čtení `vf` souboru, ale z praktické zkušenosti vím, že Mattesovy ovladače nejdříve ověří, zda k danému fontu neexistuje virtuální popis. (Pozn. K. Horák.)

v případě české sazby s akcenty nutné. Po těchto úpravách se teprve rutinou `VPtoVF` vytvoří definitivní soubory `tfm` a `vf`, vhodné pro českou sazbu PostScriptovými fonty.

Rutina `ps2pk` dokáže číst fonty v PostScriptovém formátu a digitalizovat je do bitmap tvaru `pk`. Tím pádem lze využít tisíce dnes nabízených PostScriptových fontů v čisté TEXovské sazbě bez použití výstupu do PostScriptu! Tato rutina je též vhodná pro vytvoření bitmap pro prohlížeč, které jsou většinou nutné i v případě, že zamýšlíme finální výstup do PostScriptu. Samozřejmě lze pomocí virtuálních fontů provizorně směrovat prohlížeč na jiné fonty běžně dosažitelné ve formátu `pk`. Této možnosti lze využít v případě, že drahá PostScriptová písma nemáme v počítači k dispozici, ale plánujeme finální výstup na zařízení, kde k dispozici budou.

Poznamenejme, že s virtuálními fonty dokážou dnes pracovat téměř všechny DVI-ovladače, tedy například i Mattesovy programy `dviscr`, `dvihplj` a `dvidot`. Všechny zmíněné utility jsou samozřejmě volně šířeny, tedy zadarmo. Většinu konverzních utilit lze navíc získat ve zdrojové podobě (jazyk C nebo WEB).

Bohužel uživatelé, kteří chtějí sázet v TEXu PostScriptovými fonty třeba v češtině, nemají dodnes jinou možnost, než stát se kouzelníky a formát `VPL` si sami připravit.[2]) CSTUG zatím tento formát ani pro běžné PostScriptové fonty pro národní sazbu nepodporuje a veřejně nešíří. Proto také virtuální fonty nenajdete v CSTEXu. Doufám, že tento stav je pouze dočasný (viz článek o CS-fontech, odstavec „co dělat“). V této oblasti tedy zbývá ještě hodně práce z hlediska standardizace národní sazby v TEXu.

PostScriptový `\special`

Prostřednictvím příkazu `\special` lze přímo ve zdrojovém textu TEXu formulovat některý speciální požadavek pro pozdější zpracování konkrétním ovladačem. Tento požadavek se přepíše do souboru `dvi` bez interpretace TEXem a teprve ovladač s ním nějak naloží. V případě ovladače `dvips` lze zařazovat PostScriptové příkazy. Náš zdrojový text pak může být napůl TEX, napůl PostScript. Tím přebíráme možnosti obou systémů zcela do svých rukou.

---

[2]) Předchozí tvrzení autora neodpovídá zcela skutečnosti, viz poznámku na konci Petrova článku.

Například obrázek ve čtvrté kapitole o TEXu v Softwarových novinách jsem nejprve zkoušel udělat METAFONTem. To fungovalo krásně, ovšem jen pro malá rozlišení. Při 1 270 dpi jsem narazil na limity METAFONTu a také ovladače `dvips`, který není schopen zpracovat bitmapu větší než 64 kB.[3]) Rozhodl jsem se tedy pro čistý PostScriptový obrázek. Ve zdrojovém textu jsem zařídil vynechání místa pomocí prostředků TEXu a pak jsem napsal `\special{` a dále se to v mém textu hemžilo PostScriptovými slovíčky typu `moveto`, `lineto`, `curveto` a jinými, pomocí nichž jsem tento jednoduchý obrázek vykreslil. Po ukončovací závorce `}` jsem pak dál pokračoval v psaní textu v TEXu. Digitalizace obrázku pak probíhala až uvnitř osvitové jednotky. Pro „ladění“ obrázku jsem použil Ghostscript s výstupem na obrazovku.

Když si znovu uvědomíme možnosti PostScriptu, zjistíme, že příkaz `\special` a znalost PostScriptu nám otevírá veliké možnosti. Bohužel nevhodným kombinováním těchto možností většinou dokráčíme až k typografickému zmetku (viz níže), ale přesto bude asi čtenáře zajímat, co lze dělat.

Lze psát do kopce nebo z kopce nebo ɘvolɒɔɪxi jinak. příliš vážně.

Článek „Jak jsem se sázel svisle“ z bulletinu č. 4, r. 1992 od O. Ulrycha není nutné při vhodném použití PostScriptu brát

Písmo je možné všelijak deformovat, různě naklánět, ale při všech těchto úpravách mějme na paměti, že při použití TEXovských fontů bude RIP přepočítávat bitmapu na bitmapu a dojde ke zkreslení. V ukázkách jsem sice mohl použít „outline“ fonty, ale pro větší názornost jsem zůstal u bitmapových CM fontů, aby bylo toto zkreslení více patrné (rozlišení 300 dpi).

Všechny výše uvedené ukázky se opíraly o jedinou vlastnost PostScriptu – lineární konverze jakéhokoli objektu podle transformační matice. Tato matice se dá měnit nejen přímo, ale i skládáním elementárních geometrických operací, které mají v PostScriptu jména `translate`,

[3]) To je nemilé omezení na PC. Naštěstí existuje jednoduchý trik, jak uvedený handicap obejít. Program `dvips` má problémy s převodem větší bitové mapy ve formátu `pk`, ale převod formátu `pcx` mu problémy nečiní. Mattesův program `dvidot` s `pcx.dot` zas umí převést `dvi` soubor do `pcx`. (Pozn. K. Horák.)

`rotate` a `scale`. Takže například text otočený o 90 stupňů byl do tohoto článku zanesen takto:

```
... není nutné při vhodném použití \ps{}u brát \hskip2ex
\hbox to0pt{\special{ps: gsave -90 rotate}
   příliš vážně.\hss\special{ps: grestore}}
```

Všimněte si, že je zde nutné pracovat mírně schizofrenicky. Aktuální bod sazby se nám totiž „rozbíhá“. Něco jiného si o něm myslí TEX a něco jiného vykonává PostScript. Z PostScriptu toho moc na pochopení příkladu nepotřebujete. Příkaz `gsave` zkopíruje do zásobníku grafický stav včetně transformační matice a pracuje s kopií, `-90 rotate` mění transformační matici příslušným způsobem a `grestore` snižuje zásobník, a tudíž se vrací k původnímu grafickému stavu. Protože grafický stav obsahuje i informaci o aktuálním bodu sazby, naše ztracené body se nám znova „sejdou“ na stejném místě, kde se rozešly. Z příkladu je rovněž patrné, co dostaneme (nebo spíš nedostaneme) na výstupu v obyčejném prohlížeči bez použití PostScriptu, tj. při ignorování příkazů `\special`. Proto je nutné na konečné doladění textu použít například Ghostscript.

Možná vás ještě překvapí syntaxe PostScriptu: píšeme `-90 rotate` a nikoli `rotate(-90)`. Zde totiž panuje (podobně, jako v jazyce pro navrhování stylů v BibTEXu) zásobníkový způsob uvažování. Parametr `-90` se nejprve vloží do zásobníku a poté jej funkce `rotate` ze zásobníku sejme.

Další možností je práce s barvou (`setcolor`) a s rastrovanými stupni šedi (`setgray`) s využitím vlastnosti absolutního překrývání nakreslených objektů novými. V manuálu k programu `dvips` najdete příklad podobný tomuto:

Tento nápis byl do článku zařazen pomocí následujících příkazů:

```
\font\veliky=csbx10 scaled\magstep5
\setbox0=\hbox{\veliky\TeX\ a~PostScript}\dp0=0pt \ht0=0pt
{\offinterlineskip
\special{ps: 0.25 setgray}\hrule height2cm   % << šedý pás
```

```
\special{ps: 0 setgray}\vglue-20pt
\centerline{\hskip4pt \copy0}               % << černý stín
\special{ps: 1 setgray}\vglue-2pt
\centerline{\box0}                          % << bílé písmo
\special{ps: 0 setgray}\vglue22pt}
```

Myslím si, že tato ukázka nepotřebuje dalšího komentáře. Pouze je vhodné zdůraznit, že parametr pro `setgray` v intervalu $\langle 0, 1\rangle$ označuje množství světla a nikoli černé barvy. Proto 0 znamená černou a 1 bílou.

Čtenář zřejmě tuší, jak by vypadala naše ukázka bez použití PostScriptu. Odpověď: dostaneme jen velký černý obdélník

Komu se tyto konstrukce zdají příliš složité, ten může využít již hotových PostScriptových a TEXových maker, která jsou veřejně k dispozici v balíku zvaném `pstricks`. Bohužel, o tomto balíku nemohu podat podrobnější zprávu, protože nejsem sběratelem cizích maker.

PostScriptové obrázky v TEXu

V balíku `dvips` je makro `epsf.tex`, které umožňuje pohodlně vložit PostScriptový obrázek do TEXu. Obrázek by měl být v tzv. „encapsulated PostScript form“, což lidově znamená, že splňuje jistou konvenci pro použití jen jistých PostScriptových příkazů. Například v tomto formátu není dovoleno „natvrdo“ nastavovat transformační matici, ale pouze je možné relativní nastavení (vzhledem ke stávajícímu stavu matice). Také tento formát nemůže obsahovat například `showpage`, příkaz na „vyjetí“ stránky. Jsou-li tyto konvence splněny, pak lze pomocí „obkladových“ příkazů PostScriptu zařídit umístění a velikost obrázku na stránce, případně i obrázek lineárně deformovat. „Encapsulated PostScript form“ tedy představuje jakousi konvenci bezkonfliktního vkládání PostScriptu do PostScriptu.

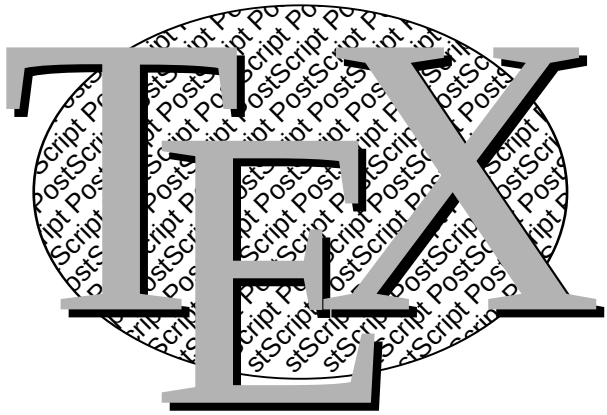
V bulletinu č.1 z r. 1993 jsem se zařazováním takových obrázků pomocí makra `epsf.tex` zabýval (v článku „Jak zařadit obrázky z *Mathematiky* do TEXu“), takže tím bych považoval tuto problematiku za probranou.

Ghostscript

Nakonec pár slov o volně šířeném emulátoru PostScriptu, který čte PostScriptový soubor a výstupem je buď obrazovka počítače, nebo tiskárna bez PostScriptu (jehličková i laserová). Jedná se o Ghostscript, šířený společností GNU. Již jsme se zmínili, že pokud si chcete prohlédnout dokument včetně všech PostScriptových efektů, které nejsou záležitostí

TEXu samotného (například barvy, zařazení PostScriptových obrázků, deformace písma apod.), je možné použít Ghostscript. Není to úplně plnokrevný PostScript (to by nebyl v „public domain“), ale v podstatě vše, co dokáží PostScriptové RIPy, zvládne taky. Někdy Ghostscript zhavaruje na nedostatek paměti počítače nebo HP tiskárny (zvláště, pokud jde o rozsáhlé obrázky). Ovšem buďme rádi, že ho máme.

V balíku je též obsažena knihovna nejběžnějších PostScriptových fontů v souborech s příponou `gsf`, ovšem tyto fonty jsou (z důvodu copyrightu) uloženy pouze v bitmapovém formátu. Budete-li je tedy moc zvětšovat, dočkáte se nepěkných zubatých tahů. Například pro šedý text v závěrečné ukázce tohoto článku bylo použito písmo Times-Roman. Pokud finální tisk článku proběhne pomocí `gsf` verzí PostScriptových fontů šířených v Ghostscriptu (odvozených z nepříliš kvalitních bitových map), pak to poznáme podle toho, že písmo bude značně hranaté. Obrázek ilustruje programovací jazyk PostScriptu.



```
\special{" gsave  /Helvetica findfont 10 scalefont setfont
/ps {(PostScript ) show} def /rad {ps ps ps ps ps} def
gsave 30 30 translate 1 .7 scale
newpath 110 110 100 0 360 arc gsave stroke grestore clip
1 1.43 scale 45 rotate -70 70 moveto
{10 -10 rmoveto gsave rad grestore currentpoint pop 100 ge{exit}if}
loop grestore
/Times-Roman findfont 150 scalefont setfont .7 setgray
/tex {(T) show -35 -35 rmoveto (E) show -30 35 rmoveto (X) show} def
30 60 moveto 0 setgray tex  27 63 moveto .7 setgray tex grestore }
```

18.9.1993 *Petr Olšák*

Několik poznámek na téma PostScriptu a PostScriptových písem v češtině

Při přípravě tohoto článku do tisku jsem zjistil zajímavý úkaz. Zatímco Petr odladil PostScriptové příkazy ke své spokojenosti, mně právě předvedené efekty poněkud ulétávaly (dokonce i ze stránky). Viníka jsme odhalili v různých verzích překladače `dvips`. Verze 5.516, kterou jsem právě používal, patrně vztahuje používané transformace k referenčnímu bodu stránky. Až dosud jsem se s tímto problémem nesetkal, protože používám buď osvědčená makra jako `rotate.tex` nebo `pstricks.tex`, nebo z nich přinejmenším opisuji. Chtěje tvrdohlavě dosáhnout žádaného výsledku bez návratu k starší verzi `dvips`, přidal jsem ke každé Petrem použité transformaci další dva řádky

```
\special{ps: gsave
             currentpoint currentpoint translate
             -90 rotate
             neg exch neg exch translate
             }%
```

Nicméně jsem byl nakonec nucen uvedenou verzi `dvips` zavrhnout ze zcela jiných, mnohem závažnějších důvodů. Ukázalo se totiž, že při převodu bitových map písmen ve tvaru `*.pk` do PostScriptu v případě, kdy se nedostává operační paměti (známá hláška `I cannot allocate more than 64 kB!`), nefunguje tento mechanismus příliš spolehlivě, a to tehdy, chybí-li paměť až na dalších stranách. Díky tomu se mi podařilo nasvítit ze 70 stran 17 takových, kde občas chyběla nějaká písmenka. Draze zaplacená zkušenost.

Jak si mi Petr postěžoval, neodpovídá skloněný nápis jeho představě. Ta byla tuším následující:

nebo *naklánět*

V Petrově `\special{ps: gsave -45 rotate 1 1.5 scale 28.1255 rotate}` jsem zkusil nahradit záhadnou hodnotu úhlu 28.1255° hodnotou $\operatorname{arctg}\frac{2}{3}$ a prohodit pořadí uvedených operací: `33.69 rotate 1 1.5 scale -45 rotate`. Proč to ale takto funguje, se mě radši neptejte.

Není tak docela pravda, že bychom k použití PostScriptových fontů pro českou sazbu v TEXu museli tolik kouzlit, jak by se z příslušného odstavce

o virtuálních fontech mohlo zdát. Jednak existuje program `accents` kolegy Zlatušky, o němž autor referoval naposledy na letní valné hromadě a který vcelku snadno umožňuje získat osmibitový virtuální popis ze sedmibitového fontu i pro standardně kódované PostScriptové fonty. Navíc součástí programů jako Adobe Type Manager nebo CorelDraw! 3.0 pro Windows je bohatá kolekce PostScriptových písem ve formátu Adobe Type 1. Protože národní lokalizace těchto „okenních“ programů obsahují obvykle písma v tzv. východoevropském kódování, které zahrnuje všechna naše akcentovaná písmena, stačí převést adobovské metriky `afm` na `tfm` do téhož kódování, v němž máme ostatní metriky. To umožňují novější verze programu `afm2tfm` z balíku `dvips`. Formálně sice generujeme virtuální popis, ten lze ale s lehkým srdcem vymazat a použít jen odpovídající výsledný `tfm` soubor. Jediným problémem snad je, že ne vždy jsou k dispozici jak `pfb`, tak i `afm` soubory. Místo nich často nalezneme jen metriky `pfm` pro tiskárnu, jejichž konverze na `afm` je zatím pro mne netriviální (pokud chceme kromě rozměrů jednotlivých boxů znát i kerningové vztahy, a to při používání TEXu jistě chceme!). Zatím nejuspokojivější výsledek dává jiný „okenní“ (komerční) program FontMonger.

Ještě stručnou poznámku ke kvalitě fontů šířených se současnými verzemi Ghostscriptu. Ghostscript jako *public domain* program obsahuje jen volně šiřitelná písma. Mezi nimi však najdeme i řadu kvalitních PostScriptových písem jako např. čtyři řezy Utopia (věnovaných firmou Adobe) anebo pěkný grotesk od firmy URW (viz obálku chystané reedice Doobova *Jemného úvodu*). Kromě toho jsou mezi Ghostscriptovými fonty varianty všech základních PostScriptových fontů s extenzí `gsf`, což je obvykle ekvivalent ASCII formátu `pfa`, ale může to být i čistě PostScriptový program. Tato extenze se používá jen pro odlišení souborů s vektorovými popisy písem. Protože tyto programy jsou v případě kvalitních písem renomovaných firem chráněny copyrightem, jsou Ghostscriptové popisy získány konverzí z bitových map a nepůsobí tedy příliš kvalitním dojmem, jsou hranaté a hrbaté. Jak to ovšem je s copyrightem v našich zemích, mi není přesně známo. Důležité však je, že Ghostscript umožňuje zobrazit libovolný PostScriptový font, takže není problém nahradit Ghostscriptová písma řezy např. z Adobe Type Manageru (máme-li je k dispozici).

Karel Horák

Novou instalaci TEXu?

Jiří Rybička

Někdy na jaře roku 1993 jsem dostal jako řádný člen sdružení pozvánku na červencové shromáždění, na němž měla být připravena nová instalace TEXu. Již delší dobu předtím jsem některé své problémy po poradě s kolegy odsouval, že prý budou v nové instalaci už řešeny. To vše jen stupňovalo mou nedočkavost.

Hned po příjezdu domů s dvěma balíčky disket jsem začal vytvářet na svém disku potřebné místo, abych mohl novou instalaci hned vyzkoušet. Pro jistotu jsem ani jeden z dosavadních souborů nesmazal, abych se mohl kdykoliv ke staré instalaci vrátit. Opatrnosti v případě tohoto téměř životně důležitého systému nikdy není nazbyt.

Průběh instalace

Vyždímal jsem skoro 20 MB prostoru a spustil instalaci. Vzhledem k tomu, že s TEXem (přesněji LATEXem) pracuji již delší dobu a že jsem byl již ze shromáždění poněkud instruován, nevyvedly mne úvodní dotazy instalačního programu z míry. Bez problémů jsem odpověděl na všechny a instalace se rozběhla. I když se na obrazovce objevila zpráva, že si mohu dát kávu nebo čaj, v mém případě (386/40) byla instalace hotova poměrně rychle (první plus). Pomyslel jsem si něco o univerzalitě systému a o počítačích řady XT, na kterých jistě také vše funguje, ale pouze pomaleji.

Po úspěšném konci instalace jsem chtěl začít pracovat, ale objevil jsem problém. Adresáře vypadaly poněkud jinak než v instalaci předchozí, mé dosud používané dávky se vůbec nedaly použít, některé důležité soubory jsem vůbec nenašel. V této chvíli jsem poznal, že jsem se přecenil, když jsem nenechal instalovat i systém menu. Označil jsem se za plážového frajera a opět sáhl po instalačních disketách. Představoval jsem si, že to všechno zase bude muset probíhat znova. O to větší bylo mé překvapení, když se instalovalo jen přesně to, co jsem potřeboval! (druhé plus).

Poté bylo potřebné, jak jsem se dověděl ze souboru CODELAT.TXT, generovat formátové soubory. Když jsem si vzpomněl, jak jsem si kdysi vytvářel formátový soubor, obcházela mne hrůza. Dávka INITEX mne však velmi rychle přesvědčila, že mám v ruce profesionální systém. Bylo

možné vytvořit snadno a rychle požadovaný formátový soubor i s volbou kódování češtiny. Okamžitě mě napadlo, že nejenom vytvoření, ale i modifikace formátového souboru bude velmi jednoduchá (třetí plus) a později jsem se o tom přesvědčil při instalaci fontů Times Roman.

Konečně jsem mohl spustit dávku LATEX, která zobrazila menu, z něhož bylo možné použít základní komponenty. **Vše fungovalo!** (čtvrté plus).

Později jsem se dověděl ještě páté plus, a to možnost přerušení instalace. Není-li například dost místa na disku, instalace nemůže dál pokračovat. Je možnost zrušit nebo odsunout některé soubory, které nejsou v instalaci nutné (například dokumentace), a postupovat dále. Instalační procedura dokáže pokračovat přesně v tom místě, kde předchozí běh skončil.

Co lze nalézt uvnitř instalace?

Editor

Klíčovou částí při práci s TEXem je bezesporu editor. Na jeho vlastnostech do značné míry závisí i celkový způsob práce se systémem. Instalace nabízí tři editory: TE, QEDIT a CSED.

Je pravděpodobné, že editor TE, který byl součástí i předchozích verzí, zvolí jako hlavní editor málokdo. Je vhodný pro malé úpravy malých textů. Jeho výhodou může být jeho malá velikost. Hodnotnými kandidáty tedy zůstávají další dva.

Novinkou instalace je editor QEDIT firmy SemWare. Používám jej na úplně všechny editační práce již velmi dlouho, proto se u jeho popisu poněkud zastavím. Jsem s jeho parametry maximálně spokojen, na drobné nevýhody jsem si zvykl a naučil jsem se je obcházet. Jedná se o sharewareový produkt, jehož licenční poplatek činí pro zámořského uživatele 65 $. Za tento poplatek dostanete tištěný manuál, který je ve velikosti přes 200 kB i na disku, ostrou verzi editoru a jeho případné verze další. Mezi nejzajímavější parametry jistě patří velikost editoru — pouhých 46 160 B!!

Je i neobyčejně rychlý. Spuštění a zavedení editovaného souboru je prakticky okamžité (například spuštění se souborem o 8 000 řádcích trvá necelé 2 sekundy — AT 386/40). Umí zpracovat libovolný počet souborů (pokud stačí operační paměť), jejichž obsah je možné zobrazit v několika vodorovných oknech na obrazovce. Kromě běžných editačních příkazů

pro pohyby kurzorem, vkládání a mazání textu, obnovení smazaných řádků, vyhledání a nahrazování umí pracovat s bloky (i sloupcovými), libovolný blok umí i vyplnit zvoleným znakem nebo seřadit podle abecedy, celý text nebo jen blok může vyslat na tiskárnu, umí kreslit rámečky ze semigrafických symbolů. Kromě toho všeho ovšem hlavně dokáže pracovat s makropříkazy. Ty jsou dvojího druhu: předdefinované instalací nebo definované v souboru maker. Obojí si může uživatel libovolně přizpůsobit.

Předdefinované makropříkazy jsou zapsány v textovém souboru, v němž jsou na úvod popsány konvence tohoto zápisu. Konfiguračním programem QCONFIG je obsah tohoto textového souboru „zabudován" do editoru. Nevýhodou této definice je omezený paměťový prostor pro makropříkazy.

Tuto nevýhodu lze kompenzovat pomocí makropříkazů zapsaných v souboru maker. Těchto souborů může být neomezené množství, je však možné v jednom okamžiku použít jen jeden. Zde jsem na omezení velikosti paměti nenarazil. Soubor makropříkazů lze z disku přečíst do paměti editoru buď v okamžiku startu, kdy je jméno tohoto souboru zadáno z příkazového řádku, nebo kdykoliv jindy volbou služby `Read macro` z nabídek. Službou `Write macro` lze soubor makropříkazů uložit na disk.

V kterémkoliv okamžiku lze elegantně vytvořit nový makropříkaz v paměti a jeho obsah přiřadit téměř libovolné klávese (například i F11 a F12 včetně kombinací s Ctrl, Shift i Alt). Vytvoření makropříkazu probíhá tak, že uživatel provádí požadovanou činnost a klávesy se zaznamenávají.

Je samozřejmé, že tímto způsobem lze vytvořit pěkné přizpůsobení pro TEX (viz novou instalaci), kdy jsou do příslušného souboru maker vloženy často psané sekvence příkazů, např. `\begin{`, `\item`, `\section{}`, `$_{}$` a podobně. Tento soubor pak při spuštění editoru uvedeme v příkazovém řádku a makropříkazy budou okamžitě k dispozici.

QEDIT akceptuje systémové nastavení národního prostředí — nezasahuje ani do definice kláves, ani do zobrazení na obrazovce a na tiskárně.

Editor CSED je podle slov zasvěcených kolegů stejně dobrý. Nemohu to z vlastní zkušenosti potvrdit. Existují však nejméně dva důvody pro jeho použití: je to český produkt a je pro členy sdružení již zakoupen.

Fonty

Jednou z nejradikálnějších změn, kterou nová instalace přináší, jsou nové české fonty. Zatímco ve staré instalaci se písmena s diakritikou různým způsobem skládala, zde jsou vytvořeny nové kompletní znaky (šesté plus). To sice s sebou přináší poněkud větší nároky na diskový prostor, tištěný výsledek je však profesionální, a to by měl být vždy cíl každého, kdo pracuje s TEXem. Kromě písmen s diakritikou nalezneme v nových fontech ještě znak promile a dva znaky pro horní a dolní české uvozovky.

Pro odlišení starších fontů `cm*` jsou nově upravené fonty označeny názvy `cs*`. Tím sice mohou nastat jisté problémy, ty jsou však v instalaci rovněž řešeny. Pokud například používáme čisté LATEXovské příkazy, tuto změnu nepoznáme. Soubory `.dvi`, které máme z dřívějška, budou zobrazitelné i nyní.

Při instalaci se všechny požadované fonty „sbalí“ do knihovních souborů `*.fli`, jejichž obsah je v textové formě uveden v příslušných souborech `*.lst` (sedmé plus). Toto sbalení má svůj význam zejména v úspoře místa na disku. Je všeobecně známo, že například dva soubory o velikosti 432 B a 228 B nezabírají na pevném disku 660 B, ale 4 096 B. Nejmenší alokační jednotkou je totiž jeden sektor o velikosti 2 kB, žádný soubor tedy nemůže zabírat méně. Zbylé části — fragmenty sektorů nelze nijak využít. Soubory s fonty se vyznačují tím, že je jich hodně a že nejsou příliš velké. Při jejich sbalení tedy vynikne výhoda eliminace zmíněných fragmentů.

Novější verze programů

V nové instalaci jsou zahrnuty pokud možno nejnovější verze všech programů (osmé plus), což poznáme patrně nejčastěji a nejmarkantněji u prohlížeče `dviscr`. Má některé nové parametry, je rychlejší a patrně lépe hospodaří s pamětí, protože se mi nepodařilo jej vyvést z míry souborem s mnoha fonty (u starší instalace to bylo na denním pořádku).

Pokud používáme menu, je velmi příjemné, že před tiskem na kterémkoliv ze tří typů tiskáren máme možnost prohlížečem sledovat tvar dokumentu s těmi fonty, které budou použity při tisku (tedy i s řadami 180 dpi, 240 dpi, 360 dpi).

V programu TEXCAD se objevila možnost kreslení Bèzierových křivek, avšak s možností zadat pouze 3 vztažné body. Chceme-li získat křivku vyššího stupně, musíme ji sestavit ze dvou částí. Trvalým nedostatkem

tohoto programu však zůstává, že není ochoten akceptovat národní znaky při zadávání textů z klávesnice.

Za přínosnou lze označit i novou verzi programu pro doplnění nezlomitelných mezer za jednopísmenné předložky a spojky. Je schopna doplnění nezlomitelné mezery i tehdy, je-li před příslušnou předložkou například ještě závorka. Kromě toho je možné činnost „vlnkovače“ ve zdrojovém textu vypnout nebo zapnout, což umožňuje vynechat zejména úseky v prostředí `verbatim`.

Dokumentace

Při rozbalení každé části instalace se na disku objeví i příslušná dokumentace, částečně i v češtině. Její celková délka přesahuje 1 MB, což samo o sobě odrazuje od jejího studia. Není však vůbec na škodu najít chvilku času a dokumentaci alespoň zběžně „prolistovat“. Často zde nalezneme řešení některých svých problémů.

Vnitřek instalace

Při své činnosti nová instalace hojně využívá schopností operačního systému, z něhož lze při troše fantazie vymámit fantastické věci. Veškerý chod menu[1]) je řízen několika programy a příkazovými dávkami, které si předávají informace v systémových proměnných. Žádný z těchto obslužných programů nezabírá v okamžiku chodu překladače nebo ovladačů operační paměť (deváté plus). Vše je velmi rychlé a pružné. Při určitém proniknutí do obsahu a významu jednotlivých systémových proměnných lze snadno vytvořit další vlastní příkazové dávky.

Instalace nabízí rovněž vytvoření národního prostředí v kódování Kamenických (desáté plus), a to na klávesnici i na obrazovce. Obojí je provedeno způsobem, který považuji za optimální. Ovladač klávesnice je modifikací ovladače KBD Milana Kureše, jehož zdrojový text v assembleru byl uveden před časem v časopisu Bajt. Jeho významné vlastnosti jsou: 1) zabírá v operační paměti pouhých 704 B, 2) lze v jeho přítomnosti používat jak národní, tak i původní znaky klávesnice bez složitého přepínání.

Pro zavedení českých znaků na obrazovku je k dispozici soubor FONTCS.DPI, který lze použít místo souboru EGA.CPI při instalaci národního prostředí v operačním systému. Podrobný návod lze přečíst

[1]) Popis tohoto systému nabídek byl již v TEX-bulletinu uveden.

v souboru FONTCS.DOC. Skutečnost, že se jedná o instalaci národního prostředí v operačním systému, i když v kódování Kamenických, je patrně příčinou toho, že nastavení je schopno se po grafických programech automaticky obnovit a není překážkou chodu jiných aplikací.

Závěr

Z mé pozice uživatele LaTeXu se nová instalace jeví jako čistá nadmnožina instalace starší. Je provedena inteligentně, pečlivě a svědomitě a je vidět, že to také dalo spoustu práce. Za to jistě patří jejím tvůrcům dík všech uživatelů.

Sečteno, podtrženo — 10× plus — na otázku v nadpisu lze tedy určitě odpovědět ANO!!!

Úvaha o fontech v CSTeXu

Petr Olšák

Při rozhovorech s nejrůznějšími příznivci TeXu o nové instalaci CSTeXu se často zavede řeč na otázku fontů, jejich názvů, kódování a mechanismů, s nimiž CSTeX s těmito fonty pracuje. Myslím si, že to je záležitost velice důležitá a proto jsem se rozhodl uvést čtenáře do „zákulisí“, v němž nová koncepce fontů vznikala a zmínit mnohá pro a proti, se kterými jsme se potýkali při rozhodování o tom, jak to udělat.

Sám nepatřím mezi ty tvůrce CSTeXu, kteří v otázce fontů prosazovali jednoznačně jednu určitou myšlenku – většinou jsem se při těchto diskusích zdržel hlasování, protože jsem se nepovažoval v této věci za odborníka, který má znalosti o tom, jak se to dělá v jiných zemích. Nevím tedy, co je v této oblasti nejlepší. Tím samozřejmě nechci ze sebe smést veškerou vinu za nedostatky, které CS fonty mají. Na druhé straně si myslím, že bych mohl právě proto podat možná nejobjektivnější pohled na věc. Přitom asi pohled dostatečně zasvěcený, protože jsem „byl při tom“, tj. zúčastnil jsem se všech schůzek, na nichž nová instalace vznikala.

Původní verze

Mnohého čtenáře zajímají především změny, které s sebou přechod z původní instalace na novou přináší. Podívejme se na to podrobněji z technického hlediska. Začneme tím, jak to bylo zařízeno v původním CSTEXu. Pomineme úplně pravěkou verzi, kde

```
jedin\'y mo\v{z}n\'y zp\accent23usob,
jak n\v{e}co napsat \v{c}esky
```

vypadal takto. Prvním krůčkem k pohodlnějšímu psaní bylo vytvoření preprocesoru, který text napsaný například v kódování Kamenických konvertoval do textu obsahujícího příslušné `\accent23` a podobné klenoty, a takto předpracovaný soubor se teprve nabízel ke čtení TEXem. Ani toto řešení nebylo pro národní jazyk dostačující, protože neumožňovalo implementovat vzory dělení slov.

Zastavme se až u osmibitové verze TEXu a u docela nedávné verze CSTEXu, která byla ještě před rokem šířena naším sdružením a která už velmi dobře umožňovala pořizovat české a slovenské dokumenty. Mattesův emTEX je schopen pomocí tzv. `tcp` tabulek řídit činnost preprocesoru, který je přímo zabudovaný do TEXu. Tyto tabulky se načítají jednou provždy při vytváření formátů. Byly vytvořeny tak, aby byl emTEX schopen číst například zdrojový text v „kameničtině“ a přitom vnitřně pracoval v poněkud jiném kódování – v případě původní verze CSTEXu šlo o kódování „skoro“ podle Corku (o tomto kódování se rozepíšu za chvíli, protože kolem něho se to všechno točí). Tím stará verze vytvořila soubor `dvi`, který v sobě obsahoval odvolávky na akcentovaná písmena (říkejme mu pracovně osmibitový `dvi` soubor). Přitom tyto odvolávky neměly svůj protějšek v bitmapách (`pk`), takže se uvedený soubor `dvi` nedal použít přímo pro zpracování ovladači. Proto Oldřich Ulrych vytvořil konverzní rutinu `dvi2dvi`, která z osmibitového `dvi` souboru udělala jiný a delší soubor stejného jména, kde všechny odkazy na akcentovaná písmena byly nahrazeny řadou DVI-příkazů stejných, jako kdyby byl použit příkaz `\accent`. Tato rutina byla zařazena do dávky hned za volání TEXu a „průměrný“ uživatel o její činnosti nic nemusel vědět. Výsledkem byl nakonec sedmibitový `dvi` soubor, který se dal zpracovat jakýmikoli ovladači, pracujícími s Computer Modern fonty. To byla veliká přednost z hlediska kompatibility.

Toto řešení mělo ale i nedostatky. Především háček v CM fontech jako samostatný znak, který se příkazem `\accent` posazuje nad písmena, vůbec neodpovídá českým typografickým představám a navíc jej TEX

umisťuje na geometrickou osu písmene a nikoli podle „výtvarného citu“. Také zde nebyla dodržena Knuthova zásada o neměnnosti CM fontů. Soubory `tfm` se totiž jmenovaly CM a přitom byly osmibitové – tedy TEX byl schopen například font zavedený jako

```
\font\titul=cmbx10 scaled\magstep4
```

zpracovat i s háčky a čárkami! Pro uživatele $\mathcal{AMS}$TEXu a plainu, kde se bez příkazu `\font` nelze obejít, to bylo pohodlné řešení. Ovšem tito uživatelé si možná neuvědomovali, že pak jejich zdrojové texty nejsou mezinárodně přenositelné, protože standardní instalace TEXu s CM fonty má správně na výskyt akcentovaného znaku v takto zavedeném fontu reagovat v `log`u zprávou

```
Missing character .. in font..
```

a znak zcela ignorovat. Je sice pravda, že české texty asi nikdo nebude chtít mít mezinárodně přenositelné, ale nezasahovat do CM fontů je požadavek samotného autora TEXu a CM fontů a má k tomu asi dobré důvody. Ze staré verze si tedy uživatelé osvojili určitý zlozvyk v používání příkazu `\font`, jehož odnaučení bude činit značné potíže. Já sám patřím mezi takové uživatele.

Je třeba si uvědomit, že tato věc *vůbec nesouvisí se zvoleným typem kódování fontu*, takže kritici použitého kódování, kteří argumentují, že není možno použít příkaz `\font` „přirozeným způsobem“, jsou vedle jak ta jedle.

Mechanismus práce s fonty v nové verzi

Uveďme nejprve základní cíle, které byly kladeny na nově vytvořenou instalaci. Jsou tři:

- 100% kompatibilita s CM světem
- Zařadit osmibitové bitmapy fontů
- Nemít bitmapu žádné abecedy zbytečně dvakrát, tj. šetřit diskovou kapacitu

K dispozici byly osmibitové CS-fonty Petra Nováka, které vycházely z CM fontů, ale navíc obsahovaly další znaky odpovídající písmenům s pevnými akcenty, používanými v české a slovenské sazbě. METAFONTové zdrojové texty těchto fontů dal Petr Novák plně k dispozici $\mathcal{CS}$TUGu a to byl vlastně hlavní impuls k vytvoření naší nové instalace. Než se tyto fonty dostaly do nového balíku, byly ještě upraveny Láďou Lhotkou a Karlem Horákem.

Pro jména fontů jsme použili původní schéma Petra Nováka, tj. jejich názvy se shodují s CM fonty, ovšem s tou výjimkou, že místo prvních dvou písmen `cm` jsou použita písmena `cs`. V instalaci jsou na úrovni metrik vedle sebe obě skupiny fontů, jak CM, tak CS. Najdete zde tedy jak soubor `cmr10.tfm`, tak `csr10.tfm`. Pouze pro matematické fonty alternativa CS neexistuje. CM metriky jsou nyní *originální* Knuthovy metriky nikterak neupravované. CS metriky obsahují i akcentované znaky a přitom v „dolní“ polovině tabulky (to je ta část s kódy menšími než 128) se CS fonty shodují s CM fonty.

Formát, který se automaticky generuje v CSTEXu při inicializaci, má předponu CS, tedy např. `csplain, cslatex` apod. Při inicializaci `csplain`u a `csamstex`u je na chvíli předefinován primitivní příkaz `\font` (viz soubory `csplain.ini`, `csamstex.ini` a `csfonts.tex`). Tím pádem všechny formátem definované příkazy pro práci s fonty (např. `\bf`, `\it` apod.) automaticky pracují s fonty CS místo CM fontů. Protože na konci inicializace je příkazu `\font` vrácen původní význam (nedělalo by to dobrotu, kdyby tomu tak nebylo), je nutno pro zavedení uživatelského fontu při sazbě češtiny či slovenštiny použít *výslovně* CS font, tedy například

```
\font\titul=csbx10 scaled\magstep4
```

F ormáty LATEXu a AMSLATEXu jsou řešeny trochu jinak. Zde není příkaz `\font` vůbec v průběhu inicializace předefinován, ale jsou načteny pozměněné definiční soubory pro fonty (viz soubory `makefmt.bat`, `lfontscs.tex`, `fontdef.cs` a další). Výsledný efekt je stejný: napíše-li uživatel LATEXu třeba příkaz `\huge\bf`, zavede se a použije se místo fontu CM font CS. Takový uživatel se nemusí o nic starat, protože většinou nemusí pracovat explicite s příkazem `\font`.

Výsledek po zpracování TEXem tedy obsahuje v `dvi` souboru odkazy na akcentovaná písmena ve fontech s označením CS. Ovladače nyní pracují s osmibitovými bitmapami CS fontů a nemají žádný problém. Nepoužívá se žádný postprocesor.

Co se stane, zpracováváme-li CS formátem anglický dokument? Pokud v něm je použit příkaz `\font`, pak určitě se jím zavádí CM font a nikoli CS. Ve výsledném `dvi` souboru pak máme jakýsi „mix“. Texty zpracované implicitními příkazy z formátu (`\bf`, `\rm` atd.) jsou vysázeny v CS fontech a explicite definované příkazy způsobí sazbu v CM fontech. Nyní záleží na tom, co s takovým `dvi` souborem chceme dělat. Zpracujeme-li jej na naší instalaci, pak ovladače budou číst pouze osmibitové bitmapy CS a použijí z nich jen dolní polovinu tabulky. Narazí li

na požadavek sazby v CM fontu, ovladač místo toho načte odpovídající CS font, jak praví substituční tabulka (viz `subst.drv`). Takže se zpracováním pomocí Mattesových ovladačů znovu nejsou žádné problémy. Navíc nemáme bitmapu žádné abecedy v instalaci zbytečně dvakrát.

Chceme-li naše „mixované“ `dvi` poslat do ciziny, musíme jej nejprve transformovat programem `dviout`, který zkontroluje, zda v celém dokumentu není jediný odkaz na akcentované písmeno a přejmenuje v `dvi` souboru všechny CS fonty na odpovídající CM fonty a upraví kontrolní součty. Takto zpracovaný `dvi` soubor je shodný se souborem, který by mohl být teoreticky vytvořen jinde standardním formátem pracujícím pouze s CM fonty. Nikdo nepozná, že jsme pro vytvoření tohoto souboru použili náš lokální formát s předponou CS. Tím je dosažena 100% kompatibilita s CM světem.

Druhá věc, kterou může chtít uživatel udělat s „mixovaným“ `dvi` souborem, je jeho transformace do PostScriptového popisu stránek pomocí programu `dvips`. Tento program v naší instalaci také pracuje pouze s CS bitmapami, ale bohužel nemá schopnost načítat substituční tabulky, jak tomu bylo u emTEXových ovladačů. Proto je do dávky zařazen před volání programu `dvips` konvertor `dviout`, který tentokrát pracuje obráceně: všechny CM názvy v `dvi` souboru přepíše CS názvy (viz soubor `others.bat`).

Je vidět, že uvedenými mechanismy jsme dosáhli tří vytýčených cílů, které jsou zmíněny na začátku tohoto odstavce.

Kódování fontů

Všimněte si, že jsem se zatím nezmínil o kódování CS fontů. Chtěl jsem, aby bylo patrné, že všechny výše zmíněné problémy jsou *nezávislé* na volbě kódování horní poloviny tabulky, takže k tomu, abychom splnili zmíněné tři cíle, jsme mohli zvolit jakékoli kódování fontů, například Pišvejcovo kódování. Na druhé straně žádné super skvělé kódování by žádný výše zmíněný problém neumožnilo řešit elegantněji.

Připomeňme si, že *vnitřní* kódování TEXu (a tedy kódování fontů) není v emTEXu závislé na *vstupním* kódování zdrojových textů, protože mezi těmito dvěma světy stojí `tcp` tabulka preprocesoru, který je zabudovaný v emTEXu. Podle této tabulky se tedy konvertuje kódování vstupního textu (Kameničtí, PC Latin2, KOI8 či jiné) do předem daného vnitřního kódování. V instalaci CSTEXu umožňujeme při inicializaci formátu vybrat ze tří `tcp` tabulek, které odpovídají třem nejběžnějším vstupním

kódováním na PC strojích. Není přitom problém vyrobit `tcp` tabulku pro jiná kódování. Po zpracování preprocesorem se tedy stává *vstupní* kódování nepodstatným pro naše další úvahy, takže se jím dále nebudeme zabývat.

Původní Novákovy fonty byly v KOI8. Je zřejmé, že toto kódování není příliš perspektivní, a proto je také vyloučíme z našich úvah. Navíc Láďa s Karlem zařadili do METAFONTových textů jakési „meta-kódování“, takže změna rozložení znaků je záležitostí celkem komfortního editování *jediného* souboru.

Prvním kandidátem na vhodné vnitřní kódování TEXu by se mohlo jevit kódování podle Corku. Tento standard byl smluven na schůzce představitelů uživatelů TEXu pro latinkou píšící země v Evropě. Na základě tohoto kódování vznikly tzv. DC fonty, které nabízejí sjednocení toho, co potřebují evropské země. Najdete tam třeba å, ç, ù, Ê nebo ñ. Samozřejmě tam najdete i akcentovaná písmena používaná u nás.

Důležitým a nezanedbatelným argumentem pro standard podle Corku je skutečnost, že se podle tohoto standardu a na základě existence DC fontů začínají šířit nejrůznější makra, která s těmito fonty pracují. Taky vzory dělení slov pro evropské jazyky se zřejmě budou šířit v kódování Cork. Dále je třeba si uvědomit, že nelze míchat různá kódování v TEXu. Tj. že bych třeba přepínal mezi DC fontem a CS fontem a přitom by se samo přepnulo i kódování. To lze řešit jenom virtuálním popisem fontu. Vnitřní kódování TEXu je jediné. Poznamenejme ale, že přepínání mezi CS a DC fonty je dosti nesmyslný požadavek, protože z hlediska sazby českých nebo slovenských textů oba fonty nabízejí stejné možnosti. DC fonty mají znatelně horší typografickou kvalitu, ale zato obsahují některé „zahraniční“ symboly, které ovšem v případě potřeby lze poskládat i z diakritiky CM fontu (viz ñ).

Bohužel standard z Corku má jednu podstatnou nevýhodu: diktuje, jak má vypadat rozložení znaků v dolní polovině tabulky. Přitom se na mnohých místech liší od CM standardu! Například tam, kde v CM fontu najdeme řecká písmena (pozice nula, jedna, dvě, ...) jsou v DC fontech umístěna různá interpunkční znaménka.

Pokud bychom tedy chtěli použít „čistý Cork“ (jako je tomu v DC fontech), museli bychom v CS fontech zasáhnout do dolní poloviny tabulky, čímž ovšem nelze splnit požadavek o nejvýše jedné bitmapě pro každou abecedu. Museli bychom totiž mít vedle sebe i originální CM fonty už třeba jen kvůli tomu, abychom nepřišli o řecká písmena. Vyznavači DC fontů si tedy plní disk nejen neužitečnými znaky, které skoro

nikdy nepoužijí a které se dají poskládat pomocí diakritiky z CM fontu, ale navíc musí mít CM fonty extra zvlášť. Tento nápor na diskovou kapacitu průměrného uživatele, který požádá o instalační diskety CSTEXu naše sdružení, jsme nechtěli připustit.

Mohli jsme tedy zůstat u „polovičního Corku“, tj. použít smíšené kódování Cork-CM tak, jako tomu bylo i u předchozí verze CSTEXu. Toto řešení taky zvolili naši polští kolegové. Zde bychom se ale museli samostatně rozhodnout, kam umístit některá interpunkční znaménka, která podle Corku kolidují s CM. Taky bychom asi neobsadili všechny pozice ve fontu, ale jen ty, které obsahují znaky používané v české a slovenské sazbě. Tento přístup mohl být jistým kompromisem, který by ale nakonec stejně nic neřešil. Zřejmě by toto poloviční řešení ani neumožnilo snadno použít makra vyvinutá pro DC fonty. Vyhovovalo by pouze uživatelům, kteří si nějak vylepšili původní instalaci, odstranili postprocesor `dvi2dvi` a na tomto starém smíšeném kódování začali dále stavět (např. si sestavili virtuální popisy k méně dostupným fontům, nebo si vytvořili některá makra závislá na kódování).

Do našich úvah vstoupil další argument, který se jmenuje ISO Latin2. Je to obecná norma pro „zpracování dat, textové aplikace a výměnu informací v albánštině, češtině, angličtině, němčině, maďarštině, polštině, rumunštině, srbochorvatštině, slovenštině a slovinštině“. Norma ISO má velkou váhu zvláště mezi počítačovými odborníky, kteří se neomezili jen na hračičky typu DOS a WINDOWS. Například v UNIXovém světě je jedinou alternativou (tam nepanuje taková džungle jako na PC). Ještě informace pro ty, co něco vědí o kódování Latin2 na PC: ISO je něco jiného. Ačkoli jde spíš o definici rozšíření ASCII kódu pro horní polovinu tabulky než o kódování fontu (tj. nehledejte tam ligatury, samostatné akcenty apod.), ukázalo se, že pokud se přizpůsobíme tomuto kódování, mají UNIXoví administrátoři ulehčenou práci s instalací češtiny v TEXu. Jde totiž o to, že standardní Knuthův TEX, který je možno instalovat pod UNIXem, není emTEX. Tedy není vybaven zabudovaným preprocesorem a mechanismem `tcp` tabulek. Tím pádem musí být vstupní kódování textu shodné s vnitřním kódováním TEXu (pokud nechceme programovat filtry, které by z důvodu zpětného výstupu chyb na terminál a do `log`u musely být oboustranné). Jak už bylo řečeno, ISO Latin2 je přitom standard pro kódování českých textů na výkonných systémech a zdá se, že zde například bratři Kameničtí neprorazí. Proč tedy neudělat CS fonty v tomto kódování? UNIXoví administrátoři nebudou muset programovat ani filtry, ani ne-

budou muset sestavovat virtuální popisy fontů. Stačí jim, když použijí naše CS fonty.

Tak se nakonec stalo, že CS fonty jsou v horní polovině tabulky podmnožinou ISO Latin2. Jsou totiž vybrány jen některé pozice, které odpovídají písmenům z češtiny a slovenštiny. Jedná se zhruba o padesát pozic; zbylých 80 zůstalo neobsazeno. Z mého laického pohledu je toto řešení asi stejně dobré i špatné jako použití „polovičního Corku“. Jsou lidé, kteří se přiklánějí spíš k jedné či druhé variantě, ale zatím žádný názor nepřevažuje. To zřejmě ukáže až čas. Rozhodně ale CS fonty jsou tímto rozhodnutím jednou provždy vázané k tomuto kódování a instalace CSTEXu je na nich nyní založena. Pokud se ukáže někdy v budoucnu vhodné použít jiné kódování, bude se muset změnit název fontů. Bude-li toto zaručeno, pak může u nás vzniknout solidní přenositelnost české sazby na úrovni `dvi` souborů. Nezdá se mi nereálné, aby například některé firmy vlastnící osvitovou jednotku nabízely osvit z `dvi` souboru. METAFONTování bitmap by pak mohla řešit firma sama, protože pouze ona přesně ví, jaké rozlišení je třeba použít. Zákazník by se o to nemusel starat. Pouze by muselo být řečeno, v jakém standardu je `dvi` soubor připraven. Zda se jedná o CM, CS nebo DC fonty. Proto upozorňuji všechny rýpaly, kteří chtějí měnit METAFONTové zdroje CS fontů, že ačkoli je změna rozložení velmi snadná, nedělejte to. Rozložení fontu, jehož výpis byl ostatně přiložen ke každé instalační krabici CSTEXu, se měnit nesmí.

Přiznáváme, že nové řešení není zdaleka ideální. Chtěli jsme jen ukázat, že v tuto chvíli zřejmě žádné řešení nebude ideální. Kdyby se pánové v Corku dohodli jen trochu rozumněji (nepožadovali změnu v rozložení dolní poloviny tabulky), vypadala by dnes situace podstatně jinak. Z dnešního hlediska se jeví, že tehdy mohlo být výhodnější, kdyby se tito pánové vůbec nedohodli. Vytvořit rozumný standard lze totiž pouze v případě, že zde nefiguruje žádný špatný standard, na němž už část TEXovské obce začala stavět.

Co dělat?

Závěrem bych chtěl upozornit na některé bolístky nového CSTEXu a při té příležitosti nabídnout aktivním a šikovným kolegům možnost se programátorsky realizovat.

První věc se může zdát celkem jednoduchá, ale zdání klame. Jde o makro `csfonts.tex`, které redefinuje primitiv `\font` tak, že pokud napíšeme

```
\font\muj=cmcokoli
```

bude to interpretováno jako

```
\font\muj=cscokoli.
```

přičemž tato změna významu proběhne pouze tehdy, jestliže `cmcokoli` není matematický font. V tomto článku jsem se zmínil, že jsem takové makro použil při generování CS formátů, ale že jej nedoporučuji používat globálně i v dokumentu. Znamená to, že makro není dokonalé. Znám tři chyby zmíněného makra, které způsobují, že nelze ponechat redefinovaný příkaz `\font` bez možných kolizí. Za prvé, délka názvu fontu se musí skládat aspoň ze čtyř písmen, jinak makro způsobí chybu. Za druhé, primitiv `\font` nevyžaduje povinné použití znaku `=` před názvem fontu, zatímco redefinovaný `\font` ano. Za třetí, primitiv `\font` má druhotný význam v konstrukci `\the\font`, přičemž při použití makra tento význam ztrácí. Šikovný makro-programátor by možná mohl tyto problémy vyřešit. Mám ale dojem, že to je natolik komplikované, že to nestojí za to a že se budeme muset všichni zlozvyk s používáním příkazu `\font` odnaučit. Nebo snad někoho napadá nějaké snadné a elegantní řešení, které by nebylo v rozporu s požadavkem autora CM fontů?

Další věcí je problém přenosu národního textu na zařízení, které není vybaveno CS fonty. Určitě by mnohý uživatel rád využil možnosti poslat třeba český text do zahraničí v souboru `dvi` v CM standardu, tj. ve formátu zpracovatelném na jakékoli instalaci TEXu ovšem za cenu nižší typografické kvality provedení háčků a čárek. To totiž v předchozí verzi CSTEXu šlo. Tam byla tato nižší typografická kvalita háčků jedinou možnou alternativou. V nové verzi nelze zatím tohoto návratu k CM dosáhnout. Dá se sice před zpracováním TEXem použít program `cstocs`, který ztransformuje všechny výskyty akcentovaných znaků do kontrolních TEX-sekvencí, ale pak přestane fungovat dělící algoritmus TEXu pro český (případně slovenský) jazyk! Použití `cstocs` je tedy možné (a nutné)

pouze v případě anglického dokumentu, kde figuruje jen několik málo českých slov (například jména autorů, citace apod.).

Co tedy udělat pro možnost zasílat plné české/slovenské texty v `dvi` souboru v CM standardu do ciziny? Zdá se mi poněkud velký luxus kvůli tomu uchovávat starou verzi CSTEXu, která navíc pracuje naprosto odlišným způsobem. Chtělo by to pro tyto účely udělat něco podobného, jako byl v původní verzi konvertor `dvi2dvi`. Olin se ovšem k tomuto programu už nechce vracet. Jeho verze byla totiž víceméně dost komplikovaná. Měla ve svém `exe` kódu pevně stanovenou množinu fontů, se kterou uměla pracovat. Tabulku akcí, co má s daným písmenem daného fontu dělat, se program `dvi2dvi` „naučil“ od TEXu v okamžiku své kompilace (v Turbo Pascalu). Matně si vzpomínám, že tato kompilace byla natolik komplikovaná (v různých fázích kompilace bylo nutno volat TEX na různé soubory a pak pokračovat), že jsem kdysi snahu naučit program `dvi2dvi` pracovat s novým fontem nakonec vzdal. Nabyl jsem dojmu, že to může pochopit jen autor programu.

Ideální by bylo, kdyby existoval takový konvertor, který by obsahoval některé algoritmy TEXu přímo v sobě a nemusel se pro každý nový font znova učit, co má dělat. Jedná se většinou o algoritmy, které se spustí primitivem `\accent`. Našel by se někdo, kdo by se odvážil do toho pustit?

Další věc, která by byla obecně užitečná, je podpora nejběžnějších PostScriptových fontů v českém jazyce. Určitě by nebylo od věci, kdyby CSTUG mohl nabízet virtuální popisy těchto fontů (`VPL` – virtual property list) k veřejnému použití v TEXu. Je sice pravda, že se jedná o komerční fonty, ale ty nejběžnější z nich jsou součástí v podstatě každého PostScriptového zařízení (včetně WINDOWS) a požadavky na jejich použití v TEXu budou určitě stále častější. Navíc tyto popisy, které se nutně opírají o vnitřní kódování TEXu, upevní pozici ISO Latin2 v našich luzích.

Nakonec jedna programátorsky možná atraktivní záležitost, ale nikterak nesouvisející s fonty. Možná už také existuje v „public domain“, jen o tom nevím. Jde o to, že by se CSTEX stal atraktivnějším pro začínající uživatele, kdyby byl vybaven inteligentním hypertextovým helpem. Představme si, že máme konfigurovatelný editor, kde lze nakonfigurovat funkční klávesu tak, že se na její stisknutí vyvolá příkaz DOSu a navíc se do tohoto příkazu obkreslí slovo, na němž stojí kurzor. Například QEdit to umí a CSEd pravděpodobně taky. Jde tedy o to získat program, který nabízí po vyvolání z řádky DOSu nápovědu ke slovu, které je na řádce DOSu uvedené jako parametr. Ideální by bylo, kdyby byl pro-

gram schopen najít i slova příbuzná a byl vybaven silnými hypertextovými možnostmi. Samozřejmostí je čtený text v komprimované podobě a s rychlým přístupem, protože takový LaTeXbook či TeXbook zabere jistě dosti místa i ve formě nápověd. Taky možnost snadného vytváření a doplňování nápověd je nutná.

Rozhodně by se našly i další nevyřešené problémy, co by se dalo dělat. Máte-li do něčeho chuť, doporučuji nejprve kontaktovat výbor CSTUGu. Podmínkou by totiž v každém případě měla být možnost zařazení softwaru do „public domain“. Určitě nebude CSTUG kupovat komerční výrobky s komerčními chybami a za komerční ceny. Taková věc je v příkrém rozporu s myšlenkou samotného TeXu.

Několik poznámek ke spuštění TeXu (tex386) pod Windows (DPMI).

Josef Krob

Popis (viz příloha) spuštění TeXu v protect modu je použitelný až na jednu maličkost. Abych ostatním ušetřil možná zdlouhavého hledání chyby (alespoň mně to trvalo dlouho, než mi došlo, o co jde), nabízím svou zkušenost s popsanou praktikou.

Nejdříve popis prostředí, ve kterém se vše odehrávalo: 386DX/40, 4 MB, DOS 6.0 + 4DOS v.4.02, Windows 3.1, QEMM v. 6.03. Příkazový interpret 4DOS je vhodné nahradit klasickým COMMANDem, protože se musí nastavovat proměnné (SET xxx = ...), a to pod 4DOSem neprobíhá vždy nejčistěji.

Vlastní úpravu tex386.exe je nutno provést mimo DPMI, a to podle popisu.

Z balíku dpmigcc1.zip je vhodné si spolu s rsx.exe zkopírovat i soubor rsx387. Může se totiž stát, jako tomu bylo v mém případě, že jakmile se pokusíte upravený tex386 spustit pod DPMI, skončí běh programu po vypsání hlavičky hláškou „Can't find emu“. Po delším hledání a zkoušení jsem zjistil, že nejde o pštrosa, ale pravděpodobně o emulátor matematického koprocesoru, jehož použití je na zvážení rsx.exe, a v tako-

vémto případě je potřeba do systémových proměnných ještě zařadit `SET RSX387=C:\BIN\RSX387`, což v uvedeném popisu nenajdete. Poté již nic nebrání tomu, aby se tex386 rozeběhl v jednom z woken.

Ještě je vhodné mít na paměti, že varianta s rsx.exe běží pouze pod DPMI. Chcete-li se tedy vrátit k verzi DOS, je třeba přestavit systémové proměnné na `SET EMX=C:\BIN\EMX.EXE` (bez těchto nastavení je tex386.exe po provedených úpravách nespustitelný).

A jak to vypadá s rychlostí? K testu jsem použil 159 stran (výsledných) bezproblémového textu (formálně), knižně členěný — kapitoly (24), obsah, větší množství poznámek, žádná matematika, několik obrázků (bm2font), výsledný `*.dvi` o velikosti 691 304 bytů.

Tabulka napoví:

	Windows		DOS
	Full screen	Windowed	
tex386 rsx	$6'15''$	$6'30''$	–
tex386 emx	–	–	$3'11''$

Během měření jsem samozřejmě nespouštěl pod Windows žádné jiné aplikace, veškerou kapacitu jsem přenechal TEXu. Je jistě možné smést nepříliš povzbudivé (pro Windows) výsledky měření rychlosti (žádné překvapení) se stolu poukazem na to, že můžeme během běhu TEXu spouštět jiné aplikace, tisknout apod., čímž se ztracené minuty vrátí. Snad je to námitka oprávněná, ale pouze pokud jste vybaveni zřejmě podstatně větším kusem paměti, než jsou mé (strojové) čtyři MB. Já, jsa takto vybaven, jsem nedokázal spustit současně kompilaci a v druhém okně mít třeba prohlížeč (dviwin 2.7). Posléze se ukázalo, že nejen prohlížeč, který sám o sobě není nikterak skromný, ale v podstatě vůbec nic. Ani kapitána Nortona. Pokaždé když se někdo pokusil texu(rsx) uždibnout paměti, skončil svůj běh krátce po startu vypsáním hlavičky a hláškou DYN: out of memory.

Když jsem se ho pokusil ošálit tím, že jsem jej (tex-rsx) nejdříve spustil, nechal jej běžet na pozadí a teprve poté odstartoval jinou aplikaci ($\mathcal{CS}$ED), fungovalo to jen do té doby, než jsem $\mathcal{CS}$ED opustil (abych se podíval jak si tex-rsx vede). Zpátky se vrátit už byl problém opět pro nedostatek paměti, kterou tex-rsx pohotově zabral.

Veškeré spouštění TEXu (DPMI i DOS) se odehrávalo z prostředí TEXShellu, protože i další teoreticky možná výhoda — použít ke zpracování zdrojového textu pro TEX výhod prostředí GUI — se ukázala

na dané konfiguraci jako nerealizovatelná opět pro nedostatek paměti. Pokud celé okno nezatuhlo, skončilo to známou hláškou out of memory. To v případě, že jsem TEX zavolal z prostředí Programmer's File Editor (PFE). Náročný a mocný EMACS v úpravě pro TEX a Windows (MEWIN, MEWLaTEX) rovněž uzavřel dosovské okno, ve kterém ve většině případů prokmitne konec nadějí s již otravným „out of memory". Situace je o to beznadějnější, že mnohdy nestačí prostě ostatní aplikace uzavřít a znovu se pokusit spustit tex-rsx, ale je nutné zcela ukončit Windows, znovu je spustit a s čistou pamětí … Málo praktické.

Naprosto tak nebylo možné při přepínaní (celoobrazovkovém) mezi prohlížečem a editorem spouštět kompilace, jak jsem to běžně praktikoval s tex286.exe.

Tolik tedy mé dílčí zkušenosti, zhodnocení ponechávám na laskavém čtenáři.

Potřebné soubory, o kterých se píše v příloze, jsou k nalezení na

`ftp.muni.cz v adresáři pub/tex/emtex/emx,rsx`

```
Running emTeX386 under DPMI
^^^^^^^^^^^^^^^^^^^^^^^^^^^

To run emTeX 386, you need do the following

   1. Remove the DOS extender attached to tex386.exe (& mf386.exe)
      of emTeX 386 distribution.

   2. Use another DOS extender that runs under DPMI.
      Such a~DOS extender is available in the network.

[0] Get Required Files via FTP

0.1 Get Rainer Schnitker's DOS Extender

   Rainer Schnitker's DOS extender (RSX) runs under DPMI (only!).

   It is available from:

      omingate.clarkson.edu
              pub/msdos/djgpp/pub/dpmigcc1.zip
      ftp.mcc.ac.uk
              pub/djgpp/pub/dpmigcc1.zip

   (I~assume that you have C:\BIN and it is included in your PATH.)
   Unzip "dpmigcc1.zip" and copy RSX.EXE in the \RSX\BIN directory
   to C:\BIN. All other files are not required for this specific
   purpose.

0.2 Get Eberhard Mattes's DOS Extender and EMX Binding Tool
```

```
   Eberhard Mattes's DOS Extender (EMX), which is attached
   to the original emTeX 386 executables, does not run under
   DPMI. Since RSX only runs under DPMI, if you want to use
   emTeX 386 under plain DOS as well as MS Windows, you need
   EMX, too. EMX is also required to run EMXBIND.EXE

   Also, you need the EMX Binding tool (EMXBIND.EXE) to remove
   Eberhard Mattes's DOS Extender from emTeX 386 executables
   and attach a~DOS Extender Loader (EMXL).

   They are available from:

      ftp-os2.nmsu.edu
              os2/2_x/unix/gnu/emx08f/emxdev.zip
      (and many other places)

   Unzip emxdev.zip and copy EMX.EXE, EMXL.EXE, and EMXBIND.EXE
   in \EMX\BIN to C:\BIN. All other files are not required for
   this specific purpose.

[1] Remove the DOS Extender Attached to TEX386.EXE (& MF386.EXE)

  Again, I~assume that you have copied RSX.EXE, EMX.EXE, EMXL.EXE,
  and EMXBIND.EXE to C:\BIN and C:\BIN is included in the PATH.
  Run the following:

     set emx=c:\bin\emx.exe
     emxbind -x tex386.exe tex386
     emxbind -x mf386.exe mf386

  The above procedure removes the DOS extender permanently attached
  to tex386.exe and mf386.exe.

[2] Attach DOS Extender Loader

   In the previous section, we removed the actual DOS extender.
   Now, we attach a~DOS extender loader:

      emxbind tex386
      emxbind mf386
      del tex386
      del mf386

   Now, to tex386.exe and mf386.exe, a~DOS extender loader
   (i.e. emxl.exe) is attached, not the actual DOS extender.
   The DOS extender loader searches for the environment variable
   EMX and loads the DOS extender specified by the variable.

[3] Running emTeX 386
```

```
3.1 Under Plain DOS (without DPMI)

   Set EMX to EMX.EXE:

      set emx=c:\bin\emx.exe

   It would be a~good idea to place the above line in your
   AUTOEXEC.BAT. Also add the following line to your AUTOEXEC.BAT:

      set emxopt=-t
      set emxtmp=c:/temp

   under the assumption that you have the C:\TEMP directory.

3.2 Under DPMI (including Windows' DOS Box)

   You have to set EMX to RSX.EXE:

      set emx=c:\bin\rsx.exe

[4] Summary

   The method presented here is summarized as follows:

  1. Remove the DOS extender attached to tex386.exe and mf386.exe
  2. Attach a~DOS extender *loader* to tex386.exe and mf386.exe
  3. Get DOS extender for plain DOS and for DPMI.
  4. Properly set DOS extenders which the DOS extender loader
     loads when you run tex386.exe or mf386.exe

Good luck.

Young
```

Konverze Word Perfect $\Longrightarrow$ LaTeX

Zdeněk Wagner

Když člověk jednou propadne TeXu (LaTeXu, $\mathcal{AMS}$-TeXu, ...), říká si, že konverze jsou zbytečné. Nejjednodušší je napsat vše rovnou v TeXu. S tímto názorem souhlasím, jenže život není žádná humanita. Často se

stává, že získáme soubor, který někdo napsal nějakým textovým editorem, a my jej máme vysázet v TEXu. V tomto článku bych se chtěl podělit o své zkušenosti (a trápení) s konverzí českých textů z Word Perfectu 5.1 do LATEXu.

K této problematice jsem se dostal tehdy, když jsem získal české texty z jistého semináře na VŠE psané ve Word Perfectu. Věděl jsem z FAQ,[1]) že existuje konverzní program `WP2LATEX`. Původně jej napsal v Turbo Pascalu R. C. Houtegen z TUE-Eindhoven. Proto jsou některé komentáře psány holandsky. Později jej pomocí `p2c` zkonvertoval Glenn Geers ze Sydney University do jazyka C a poté jej modifikoval Michael Covington z University of Georgia. Jak se dalo předpokládat, program neumí konvertovat české znaky. Dlouhé samohlásky byly nahrazeny správnými příkazy jako `\'e`, neboť se vyskytují např. ve francouzštině. Většina českých znaků se však zkonvertovala na `\P`, což není moc užitečné. Než ale popíšu, jak jsem problém vyřešil, budu pokračovat v líčení dalších potíží.

Z dokumentu z VŠE jsem si tedy zjistil kódování češtiny ve Word Perfectu 5.1. Dlužno poznamenat, že to nebyl ani kód Kamenických, ani Latin2, ani KOI8ČS. Původně jsem chtěl svoji verzi dát k dispozici na nějaký server, ale text z VŠE neobsahoval všechna česká písmena a konverzní funkce tedy nebyla kompletní. Proto jsem od svého záměru upustil.

Po krátkém čase mi volal známý, že v jistém pražském nakladatelství mu tisknou cosi napsané českým Word Perfectem 5.1[2]) a bohužel se jim porouchala tiskárna. Hledá tedy někoho, kdo by mu to nějak vytiskl. Proto jsem navštívil jinou známou, která pro jiné nakladatelství psala texty českým Word Perfectem 5.1. Napsal jsem si u ní na disketu všechna česká písmenka, abych mohl dokončit konverzní program. K mému velkému zděšení měl její český Word Perfect 5.1 úplně jiné kódování češtiny než Word Perfect 5.1 z VŠE. Nyní vás asi nepřekvapí, že známý, pro kterého jsem si konverzní program připravoval, měl ve své verzi Word Perfectu 5.1 kódování zcela odlišné...

Ideu obecného konverzního programu jsem tedy vzdal. Z programu `WP2LATEX` jsem vytvořil pomocný nástroj, s nímž lze konverzi provést, avšak definici kódování je nutno upravit podle konkrétního dokumentu.

---

[1]) Máte-li přístup na INTERNET, víte, že zkratka FAQ znamená „Frequently Asked Questions“ a příslušný dokument obsahuje nejčastěji kladené otázky týkající se TEXu a samozřejmě též odpovědi.

[2]) Všimněte si, že stále mluvím o „stejné“ verzi Word Perfectu!

Teď už se konečně dostaneme k výkladu, jak je vše uděláno. Word Perfect má řadu znakových sad (např. znaková sada 8 obsahuje řeckou abecedu). Konverzní program tedy musí rozeznat nejenom kód znaku, ale i sadu. Pro tento účel si `WP2LATEX` vytváří tabulku. Ta je definována ve funkci:

```
Local Void Ext_chr_init(LINK)
```

Funkce obsahuje řadu příkazů. Pro ilustraci uvedu následující:

```
LINK->char_set[0x1c] = 0x4;
LINK->char_code[0x1c] = 0xb;
strcpy(LINK->ext_lat[0x1c], "\\pounds ");
```

Hodnota 0x1c nemá žádný hlubší význam, je to pouze index v konverzní tabulce. Uvedené příkazy specifikují, že znak ze sady 0x4 s kódem 0xb se nahradí řetězcem `\\pounds␣`, tj. vytiskne se jako £. Pokud se nějaký znak v tabulce nenajde, v LaTeXovém souboru se objeví `\P`. Ošetření tohoto případu se vyskytuje ve funkci

```
Local Void Character(LINK)
```

v následujících příkazech:

```
if (found)
   strcpy(ch, LINK->ext_lat[j - 0x80]);
else
   strcpy(ch,"\\P");
```

V programu jsem udělal dvě úpravy. Nejprve jsem zrušil všechny příkazy ve funkci `Ext_chr_init`. Abych dostal něco užitečnějšího než strohé `\P`, nahradil jsem poslední příkaz výše uvedeného příkazu `if` příkazem:

```
sprintf(ch, "\\P{%x/%x}", (unsigned)char_set, (unsigned)char_code);
```

Tím získám ve výpisu jak číslo znakové sady, tak kód nepoznaného znaku. Kdyby se v textu vyskytoval £ a nebyl v tabulce, měl bych ve výstupním souboru `\P{0x4/0xb}`. Po dohodě s autorem textu nebo často i intuicí lze kódování rozluštit a doplnit do funkce `Ext_chr_init`. Do konverzní tabulky lze zadat až 256 znaků.

WP2LATEX má svůj vlastní „style option“ pro tisk LaTeXem. Nezkoušel jsem konvertovat matematiku ani tabulky, takže nevím, jak se v těchto případech konverzní program zachová. Při konverzi hladkých textů doplňuje určitá svá makra, ale naštěstí je jich tak málo, že je lze v libovolném editoru snadno ručně odstranit a dokument pak lze snadno vytisknout i v plain TeXu.

Zdeněk Wagner
wagner@csearn.bitnet
wagner@earn.cvut.cz

Několik poznámek z tiskařské praxe

Jiří Tlach

Každý, kdo se TeXem nějaký čas zabývá, dostane dříve či později za úkol vysázet tiskové podklady pro nějakou brožuru, skripta, časopis, manuál... nebo knížku. Protože vždy a všichni šetří – tedy co nejlaciněji a samozřejmě v TeXovské kvalitě.

Pro nejběžnější ofsetový tisk je nutno vyrobit (jako předlohu) na transparentní matérii zrcadlově převrácenou stránku (tzv. špígl). S Mattesovými ovladači to jde poměrně snadno (v `options` použijeme transformaci strany s parametrem `tr4, 5, 6 nebo 7`. Měli bychom však míti na paměti některá technická omezení vyplývající z konstrukce nejčastěji používaných laserových tiskáren a hlavně téměř magickou číslici 300 DPI, charakterizující jejich běžné rozlišení (jehličkové k tomuto účelu nemá smysl používat a užití inkoustových nutno odzkoušet na konkrétních materiálech).

Celý proces poté funguje tak, že se v tiskárně předlohy smontují na větší formát (obvykle A3 nebo A2) a kontaktně překopírují na kovolist (zde je písmo normálně čitelné a po vyvolání přijímá barvu, narozdíl od ostatního povrchu kovolistu, který barvu nepřijímá). Na kovolist se nanáší barva, gumovým válečkem se přenáší na papír – a vesele tiskneme. Samozřejmě trošku zjednodušuji, ale pro pochopení principu mi to budiž prominuto.

Abychom nebyli výsledkem své poctivé a dobře placené práce zaskočeni, je dobré si uvědomit několik omezení vyplývajících z praxe:

1) Snažte se najít spolupracujícího typografa. Kvalita publikace není jen v TEXové technické perfektnosti, ale též v optickém (typografickém) návrhu stránky. Sázet každý řádek jiným druhem písma, jak to předvádí reklamní letáčky v tramvajích, je sice možné, ale svědčí to jen o tom, že tolik písem máte někde k dispozici.
2) Nepoužívejte menší písmo než 10 bodů; sázíte-li složitější matematiku (indexy třetí a vyšší úrovně) použijte raději 12 bodů.
3) Pokud si můžete sami písmo zvolit, volte bezpatkové typu Helvetica, které je nejodolnější a dobře vykreslené i při 300 DPI tiskárny.
4) Pauzák shánějte co nejkvalitnější, bez struktur „oslí kůže“, které charakterizovaly pergamen. Pro fajnovou práci si sežeňte fólii FOLAREX MX nebo HX.*
5) Průchod papíru, fólie tiskárnou volte co nejpřímější. Většina z nich (např. HP 3 a 4) to umožňuje odklopením tajemných zadních dvířek.
6) S fóliemi zacházejte „v rukavičkách“, jsou citlivé na mastnotu před tiskem a na otěr po něm.
7) K transportu a do tiskárny dodávejte fólie proložené např. průklepovým papírem.
8) Musíte-li do textu vkládat obrázky, které nejsou pouze kresby (pérovky), ale obsahují více odstínů šedé (pro ofsetový tisk se musí rastrovat), bývá účelnější si domluvit v tiskárně dodatečnou montáž z fotograficky narastrované předlohy než je zrcadlově obracet pomocí např. `bm2font`. Pro takovéto účely 300 DPI totiž nedostačuje.
9) Snad jediný potěšující bod. Čím horší papír pro konečný tisk použijete v tiskárně (samozřejmě ne úplný h... papír), tím lepší bude výsledek. Nasákavost papíru lehce zaretušuje roztřepané okraje písma z předlohy.
10) Máte-li možnost, vyzkoušejte si celý „technologický proces“ na několika stránkách předem. Vyhnete se nemilým překvapením.

Tolik malé „desatero mladého sazeče“, který musí šetřit aby měl za tři. Máte-li ovšem přístup ke kvalitní tiskárně s vyšším rozlišením (600, 800 ba i 1 200 DPI), bude výsledek daleko přesvědčivější, ale celý postup se vám dost zkomplikuje. Nemůžete si totiž usnadnit práci Mattesovým

* Dá se objednat např. u fy. HŮLA – tel. (02) 62 79 969; cena zhruba 10,– Kč za A4.

ovladačem, musíte stránky „otáčet“ pomocí PostScriptového formátu – tedy `dvips` a vygenerovat si bitmapové fonty v příslušném rozlišení. Na laserových tiskárnách Laser Master nebo QMS se tisknou takovýmto postupem, k veliké spokojenosti, např. některé polské deníky.

K maximální dosažitelné kvalitě – osvitové jednotce máte pak již jenom skok. PostScriptový formát totiž většina z nich dnes umí (pomocí zařízení RIP) číst také. Zde se však nevyhnete alespoň základnímu odzkoušení přímo u stroje. Svítí se obvykle na filmy šířky 20, 25 nebo 30 cm v přepočítaném rozlišení cca 1 000–4 000 DPI (pro běžné písmo naprosto postačí 1 500 DPI; čím větší rozlišení, tím déle se stránka svítí, tím více zákazník zaplatí), které si můžete zvolit o vlastní újmě. Pozor však, musíte mít přístupné fonty s požadovaným rozlišením. Přijdete na to totiž až v okamžiku, kdy budete neúspěšně generovat svůj `mysoubor.ps`. S výhodou můžete použít základní sadu fontů fy. Adobe, kterou většina RIPů umí interpretovat jaksi sama od sebe. PostScriptový formát vám také umožní bezpočet kouzelných triků, které samotný TEX neumí, nebo umí velmi obtížně a nedokonale. Psát do kopce, šikmo, podle křivky… Ne nadarmo bylo PostScriptu věnováno dostatek času i na EUROTEXu vloni na podzim. Spojení s PostScriptem posouvá TEX do další profesionální roviny.

Jiří Tlach

Pozdrav z Aljašky

Pavel Sekanina

Článek obsahuje názory a zkušenosti jediné osoby. Prohlašuji, že jsem zaujatá osoba s velmi příkrými slovy pro vše, s čím nesouhlasím.

Jirka Veselý mne při mém posledním pobytu ve vlasti poprosil, zda bych nenapsal „něco“ o TEXu polárních medvědů. Možná že řekl u tučňáků, ale nerad bych mu křivdil, neboť tučňáci, jak vědí všichni žáci Járy Cimrmana, žijí pouze na jižním pólu.

Takže když jsem před rokem dával slib, že napíšu článek o tom, jak TEXují Eskymáci, netušil jsem, že budu mít tak velké „philosophické“ problémy s obsahem. Neboť jsem byl vychován pod heslem:

„Nemůžeš-li pochválit, raději mlč!“

A ono skutečně není mnoho k chválení, mám výhrady téměř ke všemu, co se týká počítačem podporované sazby.

Vlastně mám výhrady i k mnoha dalším věcem, především k otřesné kvalitě středního školství v USA. Jestli se je pokusíme napodobit, tak způsobíme sobě víc škody než celá Nejedlého jednotná škola za uplynulých čtyřicet let dohromady. Ale to patří do jiných časopisů.

Zpět k TEXu. Můžeme začít pořizováním textu. Náš ‚department of mathematical science‘ má spoustu různých počítačů propojených do sítě (do sítí). PC a Macy jako vzdálené terminály, klasické terminály, workstations (DEC, NEXT, IBM ...), VAX(y) a jeden supercomputer (Cray).

Takže si člověk může vybrat, na čem chce pořizovat základní text. Tenhle článek je psán doma na PC mého spolubydlícího. Editor? CSED. To by mohlo napovědět, jak vysoko si cením editorů na zbylých typech počítačů.

A ze všeho nejhorší jsou Macintoshe.

Ty potvory vám vlezou všude. A strašně rychle se množí ...

Jestli si myslíte, že v tak moc vychvalovaných TEXtures pořizujete ASCII text, tak si ho pošlete na jiný počítač a prohlédněte v nějakém slušném editoru. Prvních x bytů souboru jsou informace o pořizovaném souboru (jak velké okno, jaký typ písma, kde kurzor a tak — to vše je součástí souboru `.tex`).

Taky nechápu, proč každý z počítačových systémů volí jinou metodu pro kódování konce řádku (mám na mysli `<CR>` a `<LF>` zmatek). A což teprve způsob který zvolili u „NEXTů“. Jediné „rozumné“ vysvětlení jsou peníze (prachy, love, po americku ‚big bucks‘).

Mimochodem: neviděl jsem hrůznější produkt než `vi`. Tedy alespoň tak mi připadá. Ano, je mi to jasné, teď mne odepsali všichni programátoři UNIXu.

Emacs je o něco lepší. O hodně lepší. Až na to, že je tak plně programovatelný, že neexistuje žádná norma. A pro neprogramátora upravit Emacs pro jeho potřeby je dosti vysilující práce. No dobře, možná jsem úplně blbej. A nebo mi jenom neukázali jak to udělat snadno. Ale pokud se jedná o dobrý software, měl by sám vést ...

Já osobně jsem velmi špatný strojopisec (typ datel), takže pokud nemohu nadefinovat lehce a v průběhu editace okamžitě měnit makra pro klávesnici, tak jsem ztracený. Také z důvodu přenosu souborů z jednoho

systému na druhý potřebuji přesně vědět, co jsem přenesl. To zase vyžaduje editor který poctivě zobrazí i znaky mezi 0–32. Jedině systémy pod DOSem na PC splňují mé požadavky na „rozumný“ TEXovský software.

A to nehovoříme o české diakritice ...

K samotnému TEXu. Valná většina mých kolegů používá čistý Plain TEX. Argumentují tím, že to usnadňuje zasílání článků k publikaci. Překvapuje mne, že redakce časopisů nemají ekvivalentní styly pod $\mathcal{AMS}$-TEXem a LATEXem. Protože pokud se nepletu, tak například brněnské *Archivum Mathematicum* je schopno zpracovat články napsané pod oběma nadstavbami.

Takže čistý Plain TEX dává větší možnost kompatibility, na druhé straně se tím ztrácí možnosti nabízené LATEXem a $\mathcal{AMS}$-TEXem. A to nehovoříme o dalších nadstavbách. Takže se mi rýsuje práce pro jarní semestr: série seminářů o nadstavbách TEXu. Specielně prostředí `picture` a diagramy vůbec. (Teorie kategorií a lidé z computer science.)

Další nepříjemnost jsem zažil se staršími verzemi NEXT workstations. Je na nich od výrobce nainstalován TEX spolu s METAFONTem a postscriptovým driverem pro tiskárnu.

Tak za prvé (a toto zůstalo i u nových počítačů), METAFONT nepracuje, jak se od něj čeká, tedy jak je popsán v METAFONTbook. Neprodukuje `tfm` soubor. Neptejte se mne proč, nedostal jsem kloudnou odpověď od žádného z reprezentantů NEXTů.

Za druhé (u nových chyba odstraněna), to co vidím na obrazovce, `dvips` nevytiskne úplně stejně. Vyráběl jsem makro na popis kazet a perfektní rámečky na obrazovce se mi odmítaly rámovat na papíře. Takže veškerá krása NEXTové laserové tiskárny je ztracena. Rozdíl mezi 300 dpi a 400 dpi je totiž znatelný pouhým okem. Specielně při použití kurzívy.

Takže celkový dojem od polárního kruhu je spíše horší než lepší — to, co jsme docílili doma v Česku a Slovensku (v zemích bývalého Československa) je na mnohem vyšší úrovni. To se týká jak přípravy (úpravy) velkých programových balíků, tak i zkušeností a znalostí a odvahy našich TEXistů. Téměř každý, kdo doma pracuje s TEXem, by tady mohl dělat TEXperta. Pravdu díc, **ta** pozice mi není proti mysli, jen si časem musím pořídit vlastní PC pro editování.

Od polárních medvědů, neboli z nejseverněji položené university v Americe (University of Alaska, Fairbanks) všechny příznivce TEXu zdraví a mnoho úspěchů v roce 1994 přeje

Pavel Sekanina
FTPS@acad3.alaska.edu

Report on the Inaugural Meeting of the NTS Core Group: September, 1993

Philip Taylor

This is a report on the inaugural meeting of the NTS ('New Typesetting System') project group, held during the autumn DANTE meeting at Kaiserslautern (Germany) on 23rd and 25th September, 1993.

Present: Joachim Lammarsch (DANTE President,
and instigator of the NTS project);
Philip Taylor (Technical co-ordinator, NTS project);
Marion Neubauer (minutes secretary);
Prof. Dr. Peter Breitenlohner,
Mariusz Olko,
Bernd Raichle,
Joachim Schrod,
Friedhelm Sowa.

Background: Although the NTS project has been in existence for approximately eighteen months, there has not previously been a face-to-face meeting of members of the core group; at the Spring meeting of DANTE Rainer Schöpf announced his resignation as technical co-ordinator, and Philip Taylor was invited by Rainer and Joachim to take over as co-ordinator, which he agreed to do.

Joachim Lammarsch opened the Autumn meeting by reviewing the history of the project and the rationale which lay behind its creation; each member of the group then briefly reviewed his or her particular area of interest in the project, after which the group received an extended presentation from Joachim Schrod on one possible approach to the realisation of NTS. The members of the group were broadly in support of the approach outlined by Joachim Schrod, and it was **agreed** that this should form the basis for discussions at the meeting.

The approach proposed by Joachim may be summarised as follows: TEX in its present form is not amenable to modification; the code, although highly structured in some ways, is also painfully monolithic in

others, and any attempt to modify the present code in anything other than trivial ways is almost certainly doomed to failure. Accordingly, before attempting to modify TEX in any way it is first necessary to re-implement it, the idea behind such re-implementation being to eliminate the interdependencies of the present version and to replace these with a truly modular structure, allowing various elements of the typesetting process to be easily modified or replaced. This re-implementation should be undertaken in a language suitable for rapid prototyping, such as the Common Lisp Object System ('CLOS'). The primary reason for the re-implementation is to provide modularisation with specified internal interfaces and thereby provide a test bed, firstly to ensure that TEX has been properly re-implemented and subsequently to allow the investigation of new typesetting paradigms.

Once a working test bed has been created, and compatibility with existing TEX demonstrated, a second re-implementation will be undertaken; this re-implementation will have the same modular structure as the test bed but will be implemented with efficiency rather than extensibility in mind, and will be undertaken using a combination of literate programming and a widespread language with a more traditional approach, such as 'C++'. When this second version has also been demonstrated to be compatible with TEX, it will be made available to implementors around the world, the idea being to encourage people to migrate to NTS by demonstrating its complete compatibility with TEX (the test bed will also be made available if there is interest shewn in its use). Thereafter new ideas and proposals will be investigated using the test bed, and if found to be successful these will be re-implemented in the distribution version.

The main problem which the group identified with the approach outlined by Joachim was simply one of resources: in order to accomplish two re-implementations within a reasonable time-scale, it would be essential to use paid labour, it being estimated that each re-implementation requires a minimum of four man-months work to produce a prototype, and eight man-months to reach the production stage. As this is far beyond the ability of members of the group to contribute in the short term, it is clearly necessary to employ a small team (of between two and four members) to carry out the re-implementations under the guidance and supervision of one or more members of the core group. Initial costings suggested that this could not be accomplished within the present financial resources of the group, and accordingly it was **agreed** that Joachim

Lammarsch should seek further financial support. Subsequent investigations shewed that a quite significant reduction in costs could be achieved if the programming team were sited in a central or eastern European country, particularly if the members of the team were also residents of the country; this approach is being investigated.

As it was obvious that no immediate progress could be made with Joachim Schrod's proposal, even though the group agreed that it represented an excellent philosophical approach, it was also **agreed** that the group needed to identify some fallback approaches, which could (a) be commenced immediately, and (b) would be of significant benefit to the TEX community at large. The group identified two such projects, these being (1) the specification of a canonical TEX kit, and (2) the implementation of an extended TEX (to be known as e-TEX) based on the present WEB implementation. It was also **agreed** that Marek Ryćko & Bogusław Jackowski would be asked if they were willing to co-ordinate the first of these activities, and that Peter Breitenlohner would co-ordinate the second.

The ideas behind the two proposals are as follows.

(1) The canonical TEX kit: at the moment, the most that can be assumed of any site offering TEX is (a) iniTEX; (b) plain TEX; (c) LATEX; and (d) at least sixteen Computer Modern fonts. Whilst these are adequate for a restricted range of purposes, it is highly desirable when transferring documents from another site to be able to assume the existence of a far wider range of utilities. For example, it may be necessary to rely on BibTeX, or on MakeIndex; it may be useful to be able to assume the existence of BM2FONT; and so on. Rather than simply say "all of these can be found on the nearest CTAN archive", it would be better if all implementations contained a standard subset of the available tools. It is therefore the aim of this project to identify what the elements of this subset should be, and then to liaise with developers and implementors to ensure that this subset is available for, and distributed with, each TEX implementation.

(2) Extended TEX (e-TEX): whilst the test bed and production system approach is philosophically very sound, the reality at the moment is that the group lacks the resources to bring it to fruition. None the less, there are many areas in which a large group of existing TEX users believe that improvements could be made within the philosophical constraints of the existing TEX implementation. E-TEX is an attempt to satisfy their needs

which could be accomplished without a major investment of resources, and which can pursued without the need for additional paid labour.

Finally the group agreed to individually undertake particular responsibilities; these are to be:

Peter Breitenlohner: Remove any existing incompatibilities between TEX–XET and TEX, with the idea of basing further e-TEX developments on TEX–XET; liaise with Chris Thompson concerning portability of the code; produce a catalogue of proposed extensions to e-TEX.

Joachim Lammarsch: liaise with vendors and publishers in an attempt to raise money for the implementation of NTS proper; arrange a further meeting of interested parties; liaise with Eberhard Mattes concerning the present constraints on the unbundling of emTEX; negotiate with leading academics concerning possible academic involvement in the project.

Mariusz Olko: take responsibility for the multi-lingual aspects of e-TEX and NTS; discuss the possibility of siting the NTS programming team in Poland; discuss the possibility of academic involvement with leading Polish academics.

Bernd Raichle: endeavour to get TEX–XET integrated into the standard UNIX distribution; prepare a list of proposed extensions to e-TEX; lead discussions on NTS-L.

Friedhelm Sowa: primary responsibility for finance; prepare proposals for a unified user interface and for unification of the integration of graphics; liaise with the Czech/Slovak groups concerning possible siting of the NTS programming team in the Czech Republic or Slovakia; discuss possible academic involvement with leading academics.

Philip Taylor: Overall technical responsibility for all aspects of the project; liaise with other potential NTS core group members; prepare and circulate a summary of the decisions of this and future meetings.

Philip Taylor,
09-NOV-1993 14:02:03

Literatura

Na našem knižním trhu je již poměrně dlouhou dobu vážný nedostatek literatury, která by se zabývala počítačovým zpracováním textů. Nejde o manuály těch či oněch programových systémů, ale spíše o učebnice, jež by dokázaly tuto velmi širokou a z praktického hlediska velmi potřebnou oblast pokud možno uceleně a přehledně zmapovat. S cílem alespoň částečně přispět k lepší informovanosti v tomto směru byly vytvořeny dva učební texty určené pro studenty Vysoké školy zemědělské v Brně a Vysokého učení technického v Brně.

První z nich má název **Programové vybavení počítačů — Zpracování textů počítačem**. Obsahem publikace jsou tyto kapitoly:

- *Jak psát text* — několik formálních pravidel pro uspořádání vlastních myšlenek tak, aby výsledný text přinesl čtenáři maximální užitek.
- *Základy typografie a sazby* — velmi stručný průvodce některými typografickými pojmy a zásadami. Je zde poukázáno na hlavní typografické chyby, kterých se dopouštějí amatérští producenti různých textů.
- *Technické vybavení pro zpracování textů* — popis principu činnosti a provedení různých částí výpočetních systémů, které se podílejí na vytváření tiskoviny — procesor, klávesnice, myš, obrazovka, tiskárna, scanner, fotosazba.
- *Zpracování grafické informace* — popis možností tvorby, přeměny, úschovy a použití obrázků, které mohou být vkládány do textu.
- *Systémy pro zpracování textů* — klasifikace prostředků umožňujících vkládat a upravovat text pomocí počítače — textové editory a procesory.
- *Desktop publishing* — přehled systémů pro počítačovou sazbu, a to jak pro počítače typu PC, tak i pro MacIntosh.

V dodatcích textu jsou uvedeny některé informace „manuálového typu“:

A. *Programovací jazyk PostScript* — přehled některých operátorů jazyka PostScript s příklady použití.

B. *Parametrické křivky* — matematický popis křivek užívaných v grafických systémech — Hermitovy křivky, Bèziérovy křivky, B-spline.

C. *Standardní textové formáty* — popis způsobu uložení dokumentů ve formátech WordStar, Microsoft Word a WordPerfect
D. *Slovníček* — pojmy používané v publikaci

Na publikaci se podíleli tito autoři (v abecedním pořadí): Ing. Petr Hanáček, Ing. Vítězslav Obůrka, Ing. Petr Přikryl, Doc. Ing. Zdena Rábová, CSc., Ing. Jiří Rybička, Ing. Miroslav Slezák.

Druhá publikace má titul **Programové vybavení počítačů — Systémy pro počítačovou sazbu**. Obsahuje popis možností dvou různých systémů pro počítačovou sazbu, a to systému LaTeX a systému Ventura Publisher verze 3.0. Popis každého systému tvoří samostatnou část textu, která je odpovídajícím systémem také vysázena. Čtenář má tedy možnost oba systémy velmi jednoduše porovnat.

Bibliografické údaje:

[1] Rybička, J. a kol.: Programové vybavení počítačů — Zpracování textů počítačem. Ediční středisko VŠZ v Brně, 1992. 109 stran, 26,– Kč
[2] Rybička, J., Přikryl, P.: Programové vybavení počítačů — Systémy pro počítačovou sazbu. Ediční středisko VŠZ v Brně, 1992. 98 stran, 20,50 Kč.

Poznámka: Oba tituly zakoupilo sdružení CSTUG a rozešle je svým kolektivním členům. Případní další zájemci je mohou zakoupit v prodejně skript v areálu Vysoké školy zemědělské v Brně.

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	600
Tisk:	HBT-Jet PRESS Neratovice
Adresa:	CSTUG MÚ UK, Sokolovská 83, 186 00 Praha 8