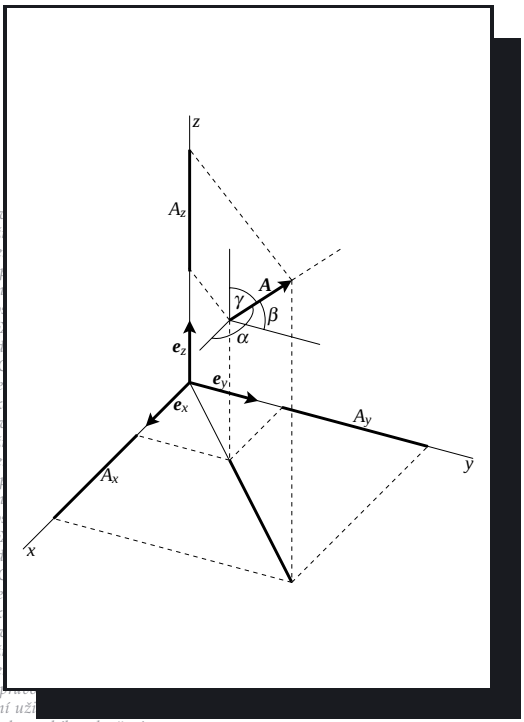
CSTUG



ZPRAVODAJ

Československého sdružení uživatelů TEXu

1

2004

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 14

OBSAH

Toto číslo obsahuje elektronické přílohy zveřejněné na
`http://bulletin.cstug.cz/bul041.shtml`

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes `http://www.cstug.cz/` .

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Postupujte podle instrukcí, které najdete na stránce `http://bulletin.cstug.cz/` . Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

Disketu formátujte nejlépe pro DOS, formáty Macintosh 1.44 MB a EXT2 jsou též přijatelné. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí CSTEXu), zejména v případě, kdy vás nelze kontaktovat e-mailem.

ISSN 1211-6661 (tištěná verze)
ISSN 1213-8185 (online verze)

Úvodníček

Petr Olšák

Omlouvám se za půlroční zpoždění prvního čísla Zpravodaje pro letošní rok. Přiznám se, že jsem měl původně jinou představu o vycházení časopisu: že bude vycházet se železnou pravidelností, že v každém časopise bude zveřejněna uzávěrka pro příští číslo a že ihned po uzávěrce udělá šéfreaktor potřebné redakční práce a časopis se dostane během pár dnů do rukou čtenářů.

Podle mého názoru existují asi tři možné překážky kladené našemu Zpravodaji do cesty. První překážkou je nedostatek článků od autorů. Když už je dostatek článků, pak se může stát, že šéfredaktor Zpravodaje nemá zrovna čas články zpracovat a připravit Zpravodaj k tisku. Dělá totiž také na zakázkách, které ho živí. Zpravodaj mezi takové zakázky nepatří. Když už se najde skulinka mezi zakázkami pro obživu a vejde se tam příprava Zpravodaje, pak mohou být zrovna prázdniny a pan Hála, který nechává v Brně Zpravodaj tisknout a distribuovat, není k zastižení, neboť je na svých prázdninových aktivitách.

Číslu 1/2004 byla dlouho stavěna do cesty překážka první, pak (rovněž bohužel velmi dlouho) překážka druhá. Nyní snad obě překážky pominuly, ale začínají prázdniny, a tudíž to vypadá na překážku třetí. Rozhodli jsme se tuto poslední překážku obejít a nechat výjimečně vytisknout Zpravodaj v Praze. O distribuci se postará paní Holovská, které za tuto nabídnutou pomoc děkuji.

Dostal jsem od některých našich členů dopisy, ve kterých shodně vyjadřují lítost na tím, že Zpravodaj v poslední době obsahuje jen vysoce odborné články, které nic nepřinášejí „průměrnému TEXistovi“. Ano, souhlasím a taky mě to mrzí. Bohužel, článků nemáme tolik, abychom si mohli vybírat a zaměřit tak časopis více na průměrné nebo začínající TEXisty. Když autoři takové články napíší, rádi je zveřejníme. Věřím ale, že aspoň v tomto čísle se objeví některé přístupnější články.

Pokusím se v této souvislosti našim čtenářům umožnit nahlédnout do kuchyně, ve které se obsah Zpravodaje připravuje. Pokud má článek zaslaný na `zpravodaj@cstug.cz` aspoň trochu hlavu a patu, pak jej zařadíme. Vůbec nemáme možnost si vybírat nějak tématicky. Často jsme rádi, že časopis zaplníme aspoň příspěvky z konference SLT. To jsme udělali v případě SLT 2002 a chystáme se obeslat autory se žádostí o možnost zařazení jejich příspěvku i ze SLT 2004. Odtud samozřejmě proudí odborné články, neboť konference SLT je zaměřena odborně. V neposlední řadě rádi zaplníme Zpravodaj CéDéčkem nebo DéVéDéčkem společně se základním manuálem. Zrovna na konci loňského roku byl takový manuál přeložen do češtiny a tak jsme využili možnost dát do Zpravodaje tento překlad.

V souvislosti s tím mě napadlo, že záležitosti kolem počítačů jsou stále komplikovanější a že se asi podobný trend odehrává se softwarem kolem TEXu. Nelze tedy jednoduše srovnávat články ze Zpravodaje před deseti lety, kdy byl, aspoň podle mého názoru, život s počítači výrazně jednodušší. Když například sleduji pozadí vzniku distribucí TEXlive, připadá mi, že je potřeba rok od roku vynaložit větší úsilí, aby to vůbec vyšlo. Chtěl bych ale zdůraznit, že se nikdy nestalo, abychom článek do Zpravodaje odmítli jen proto, že se nám zdál málo odborný. Těšíme se, že po tomto vysvětlení bude adresa `zpravodaj@cstug.cz` zahlcena desítkami článků, které jsou určeny pro „průměrné TEXisty“...

Přejdu na jiné téma. Začátkem roku se konečně Olaf Weber rozhodl zařadit encTEX podporující UTF8 kódování do standardní web2c TEXové distribuce. Tím se toto rozšíření dostane do teTEXu a distribuce TEXlive. Snad to bude uživatelům k něčemu užitečné.

Kromě toho jsem se letos pokusil nabídnout na CTAN a do TEXlive své makro OFS. Zpočátku jsem s tím měl poměrně velké problémy. Zařazení do CTANu trvalo asi 14 dní a vedly se při tom diskuse o vhodnosti licence. Naléhali na mě, ať balíku OFS změním licenci. Rozhodl jsem se totiž dát balíku OFS podobnou licenci, jako má Knuthův `plain.tex`. To znamená, že si s balíkem může každý dělat co chce, jen nikdo nesmí distribuovat soubor `ofs.tex` pozměněný se stejným názvem. Taková licence ale není v souladu s doporučeními OSI a tudíž zařazení OFS na TEXlive bylo Sebastianem Rahtzem jednoznačně odmítnuto. Navíc Sebastian zjistil, že `csplain` má podobnou licenci, a rozhodl se ho vymazat. Až teprve intervence pana Kasala na `tex-live@tug.org` zabrala a nakonec bylo OFS přijato i se svou původní licencí a `csplain` zachráněn před vymazáním. Panu Kasalovi velice děkuji. Problém Knuthova pojetí volného softwaru kontra dnešního pojetí by asi vydal na samostatný článek. Snad někdy příště.

V červnu jsme společně se sdružením CZLUG uspořádali už tradiční konferenci SLT — seminář o Linuxu a TEXu, tentokrát ve Znojmě. Program byl bohatý a pestrý. Svědčí o tom abstrakty přednášek na `www.cstug.cz/slt/04` a také sborník, který se dá koupit v knihkupectví Mareček, Botanická 68a, Brno za 200 Kč. Mrzelo mě, že se SLT 2004 zúčastnilo velmi málo lidí. V TEXové sekci bylo přítomno kolem desíti lidí. To už opravdu není o toto téma zájem? Přitom účastníci, kteří přijeli, byli vesměs spokojeni. Velmi vysoko byl hodnocen i doprovodný program spojený s návštěvou vinného sklípku, degustací a prohlídkou historické části Znojma s průvodcem. Děkuji Petrovi Adámkovi za zprostředkování tohoto programu. Myslím si, že ti, kteří na SLT 2004 nebyli, mohou jenom litovat.

6. 7. 2004

Anatomie virtuálních fontů

Zdeněk Wagner

Článek představuje úvod do problematiky virtuálních fontů. Neklade si za cíl úplnost. Zaměřuje se na vysvětlení základních vlastností virtuálních fontů. Předvádí jednoduché metody, jak virtuální font vytvořit a předvádí vybrané možnosti, jak virtuální font využít.

Úvod

Základní stavební jednotkou, kterou potřebuje každý sázecí program, je font. Právě font obsahuje uspořádané množiny znaků, jimiž vytváříme texty. Fonty jsou dostupné v různých formátech a ne všechny sázecí systémy dovedou pracovat se všemi z nich. Nás bude v tomto článku zajímat pouze jedna třída, jíž jsou virtuální fonty. Jak název napovídá, nemají daleko k virtuální realitě. Virtuální font je popis objektu, který neexistuje. Až ve vhodném okamžiku je podložen fontem skutečným.

V české TEXové literatuře dosud nebyla problematika virtuálních fontů dostatečně pokryta. Přitom virtuální fonty mohou být velmi užitečné. V tomto článku si tedy vysvětlíme, co virtuální font je a jak se dá vytvořit a využít.

Článek obsahuje příklady maker a programů. Abyste si je nemuseli opisovat, jsou k dispozici jako elektronická příloha časopisu na `http://bulletin.cstug.cz/bul041.shtml`.

Jak TEX pracuje s fonty

TEX je dávkově orientovaný sázecí stroj, který na vstupu očekává zdrojový soubor, a výstupem je hotová sazba ve formátu nezávislém na zařízení (DVI – Device Independent). Ve výstupním souboru jsou tedy uloženy informace, kam se mají umístit jednotlivé znaky, ale jejich kresba tam není. V průběhu sazby tedy vůbec nemusíme vědět, jak bude který znak vypadat. Potřebujeme pouze znát rozměry znaků a vlastnosti ovlivňující dvojice sousedících znaků: selektivní prostrkání (kerning) a slitky (ligatury). Zmíněné informace jsou uloženy v metrickém souboru, který se tradičně označuje zkratkou TFM (Text Font Metrics). Skutečnou kresbu znaků vyžaduje až ovladač výstupního zařízení, který ze souboru DVI tvoří tištěnou stránku nebo jinou vizuální reprezentaci dokumentu. Zdánlivou výjimkou je pouze pdfTEX. Při výstupu do PDF totiž musí znát kresbu znaků

již během sazby. Je to dáno tím, že sázecí stroj i výstupní ovladač pro PDF jsou integrovány v jediném programu.

TEXový sázecí stroj se nezajímá o formát fontů, vyžaduje pouze metrické soubory. Pokud si tedy vymyslíme Private Nonsense Font Specification, k fontům vytvoříme odpovídající soubory TFM a napíšeme si ovladač, který s těmito fonty bude umět pracovat, můžeme je v TEXu použít.

Každý znak má čtyři rozměry: šířku, výšku, hloubku a kurzívní korekci. Při sazbě TEX umístí referenční bod znaku na aktuální pozici sazby. Poté posune aktuální bod sazby doprava o šířku znaku a pokračuje ve stejné činnosti. Ke změně dochází pouze v případě, že mezi danou dvojicí znaků má být nenulový kerning. V takovém případě se aktuální bod sazby posune o požadovanou hodnotu. Výška a hloubka jsou nutné pro sestavování řádků. Jakmile je odstavec nalámán, zjistí TEX výšku a hloubku každého řádku. Pokud by se řádky překrývaly, zvětší TEX řádkový proklad[1]. Kurzívní korekci vkládáme příkazem `\/`.

V TEXu používáme nejčastěji dva druhy fontů: bitmapové generované METAFONTem a PostScriptové ve formátu Type1. V následujícím odstavci si je stručně popíšeme.

METAFONT je program, který z matematického popisu fontu vytváří metrický soubor a bitovou mapu určenou pro konkrétní výstupní zařízení[2]. Některé metrické informace jsou uloženy i v bitmapě. Ovladač tedy nemusí číst soubor TFM, protože informace o rozměrech znaků jsou přítomny ve fontu samotném.

PostScriptové fonty Type1 jsou specializované programy, které vykreslují znaky na zařízeních, kde je tento jazyk podporován. Obrysy znaků jsou definovány jejich obrysy, jež jsou popsány matematickými rovnicemi. Závěrečným příkazem v popisu každého znaku je posun aktuálního bodu sazby. Při tisku tedy nemusíme znát metriku, ovladač potřebuje pouze posloupnost znaků, mezislovní mezery a kerningy a povely pro přechod na nový řádek. Rozměr znaku lze získat z něho samotného. PostScript obsahuje operátor `pathbbox`, který zjistí ohraničovací rámeček (bounding box) aktuální cesty tvořené sadou křivek. Takto zjištěný ohraničovací rámeček však může být příliš veliký. Proto nejprve použijeme operátor `flattenpath`. Šířka znaku v TEXovém smyslu však nemusí (a často také není) shodná s šířkou ohraničovacího rámečku. V souboru TFM potřebujeme údaj o posunu aktuálního bodu sazby. Ten zjistíme operátorem `stringwidth`. Metrické údaje lze tedy zjistit přímo z fontu. Lze k tomu použít makro `printafm.ps` z distribuce GhostScriptu, jímž vytvoříme soubor AFM (Adobe font Metrics). Většinou však soubor AFM dostaneme současně s fontem. To je lepší alternativa, protože soubor může obsahovat informace o kerningových párech. Tento údaj ve fontu není. Metriku z formátu AFM převedeme do TFM buď programem `afm2tfm` nebo makrem `fontinst`.

[1] Toto chování lze ovlivnit hodnotou parametru `\lineskiplimit` a dalších.

[2] Výstupní zařízení se od sebe liší nejen rozlišením, ale i tvarem a velikostí tiskového bodu. Proto dvě různá zařízení se stejným rozlišením vyžadují různé korekce. Parametry pro řadu zařízení najdete v souboru `modes.mf`, který byste ve své instalaci TEXu měli mít.

Vidíme, že u dvou nejběžnějších typů fontů jsou metrické informace uloženy na dvou místech: v souboru TFM pro TEX a v samotném fontu pro výstupní zařízení. Tyto informace musí být identické. Výstupní zařízení totiž nemůže tušit, že jsme TEXu podstrčili upravenou metriku. Pokud to uděláme, dostaneme na výstupu zmatek. Jestliže potřebujeme font se stejnou kresbou znaků, ale s jinými metrickými údaji, musíme použít metodu, kterou si ukážeme v následující kapitole.

Co je virtuální font

Ukázali jsme si, že TEX při sazbě používá pouze metrické údaje. Můžeme tedy TEXu podstrčit metriku fontu, který vůbec neexistuje. K čemu se to hodí? Je to užitečné tehdy, když vlastní font je vytvořen ve výstupním zařízení. Pojmem *virtuální font* však rozumíme předpis, kterým vytváříme kresbu znaků poskládáním fragmentů z jiných fontů. Virtuální font smí obsahovat vše, co může být v souboru DVI, včetně příkazů `\special`. Pokud však do virtuálního fontu vložíme příkaz `\special` fungující pouze na určitém zařízení, bude výsledný font na tomto zařízení závislý. Při tvorbě a použití virtuálních fontů proto musíme být obezřetní.

Virtuální font sám o sobě nedefinuje kresbu znaků. Výjimkou jsou pouze případy, kdy by daný znak byl tvořen pouze linkami, jež reprezentují TEXové primitivy `\hrule` a `\vrule`, nebo byl generován výhradně pomocí příkazů `\special`. Nejčastěji virtuální znak odkazuje na skutečný znak nebo znaky v nějakém jiném fontu nebo několika různých fontech. Tyto instrukce jsou uloženy v souboru VF a ovladač DVI je při zobrazování interpretuje. V době, kdy koncept virtuálních fontů vznikl, si ovladače DVI s nimi neuměly poradit. Proto byl napsán program `dvicopy`, který při kopii souboru DVI provede devirtualizaci. Instrukce ze souborů VF interpretuje a výsledný soubor pak obsahuje pouze odkazy na normální fonty. V dnešní době byste program `dvicopy` již neměli potřebovat, protože virtuální fonty jsou podporovány ve všech ovladačích, ale je dobré o této možnosti vědět.

Tvoříme jednoduchý virtuální font

Virtuální font je tvořen binárním souborem VF. Programem `vftovp` jej převedeme na textový soubor VPL (Virtual Property List). Ke zpětné konverzi slouží program `vptovf`. Jeho výstupem je jak virtuální font VF, tak metrický soubor TFM, který vyžaduje TEX. Virtuální font můžeme tedy vytvořit tak, že si jej sami napíšeme v textové podobě a přeložíme programem `vptovf`. To však není příliš pohodlné, je výhodnější využít nějaké nástroje.

Jedním z jednoduchých nástrojů, jímž lze virtuální fonty vytvářet, je balíček qdTEXvpl od Eherharda Mattese. Hlavní program je psán v jazyce C (jako jeden z mála programů E. Mattese je dodáván ve zdrojovém kódu). Součástí balíčku je TEXové makro a fiktivní font. Princip spočívá v tom, že se požadované znaky vygenerují TEXovými prostředky. Dodaným makrem je označkujeme, přeložíme do DVI a z něj se následně generuje virtual property list. Písmena *qd* v názvu programu znamenají *Quick & Draft*. Je tedy téměř jisté, že před přeložením VPL na VF budeme muset udělat další zásahy.

Celý postup si předvedeme na jednoduchém příkladu. Nejprve si napíšeme soubor, který nazveme `qdmftex.tex`:

```
\input qdtexvpl
\font\T cmr10
\font\M mplogo10
\texvpl{t}{\T \TeX}
\texvpl{m}{\M METAFONT}
\texvpl{c}{\kern2pt\vrule height5pt depth-1pt width4pt\kern2pt}
\bye
```

Soubor definuje tři znaky, v nichž se používají dva různé fonty. Prvním parametrem makra `\texvpl` je požadovaný znak, druhým parametrem je požadovaná realizace. Soubor zpracujeme TEXem a na DVI se můžeme podívat. Programem qdTEXvpl pak vytvoříme následující VPL:

```
(MAPFONT D 17
   (FONTNAME cmr10)
   (FONTCHECKSUM O 11374260171)
   (FONTDSIZE R 10.000000))
(MAPFONT D 18
   (FONTNAME mplogo10)
   (FONTCHECKSUM O 37045067476)
   (FONTDSIZE R 10.000000))
(CHARACTER C t
   (CHARWD R 1.861079)
   (CHARHT R 0.683330)
   (CHARDP R 0.215276)
   (MAP
      (SELECTFONT D 17)
      (SETCHAR C T)
      (PUSH)
      (MOVERIGHT R -0.166702)
      (MOVEDOWN R 0.215277)
      (SETCHAR C E)
      (POP)
```

```
      (MOVERIGHT R 0.388855)
      (SETCHAR C X)))
(CHARACTER C m
   (CHARWD R 5.133313)
   (CHARHT R 0.600000)
   (CHARDP R 0.000000)
   (MAP
      (SELECTFONT D 18)
      (SETCHAR C M)
      (SETCHAR C E)
      (SETCHAR C T)
      (MOVERIGHT R -0.022223)
      (SETCHAR C A)
      (SETCHAR C F)
      (MOVERIGHT R -0.044444)
      (SETCHAR C O)
      (SETCHAR C N)
      (SETCHAR C T)))
(CHARACTER C c
   (CHARWD R 0.800000)
   (CHARHT R 0.500000)
   (CHARDP R 0.000000)
   (MAP
      (MOVERIGHT R 0.200000)
      (MOVEDOWN R -0.100000)
      (SETRULE R 0.400000 R 0.400000)))
```

Vlastnost `MAPFONT` definuje použité fonty. Pak následují definice jednotlivých znaků. Každý z nich začíná specifikací rozměrů. Pokud je některý z rozměrů nulový (typicky hloubka), nemusí být uveden. Vlastnost `MAP` popisuje realizaci daného znaku. Pomocí `SELECTFONT` se zvolí font, `SETCHAR` slouží k usazení znaku, `MOVEDOWN` a `MOVERIGHT` k posunům a `SETRULE` k sazbě linky. Rozměry jsou uváděny v násobcích `DESIGNSIZE`, jejíž implicitní hodnota je 10 pt (uvedli jsme ji při volání programu qdTEXvpl). Soubor VPL přeložíme programem `vptovf`. Metriku TFM vložíme na místo, kde TEX očekává metrické soubory, VF vložíme tam, kde ovladač očekává virtuální fonty. V některých TEXových instalacích může být vyžadována obnova databáze, některé instalace jsou schopny číst uvedené soubory z aktuálního adresáře. Přepneme-li se nyní do fontu `qdmftex` a napíšeme tcm, dostaneme:

TEX ■ METAFONT

Takto vytvořený font neobsahuje mezislovní mezeru. Museli bychom do něj ručně doplnit hodnoty parametrů `\fontdimen`, o nichž si povíme později.

Osmý bit schází nám

Písmena s diakritickými znaménky lze vysázet čistě TEXovými prostředky tak, že samotný akcent usadíme nad příslušné písmeno. Proč nás tedy zajímají fonty, kde jsou akcentovaná písmena kreslena jako samostatné znaky? Máme pro to dva důvody. První důvod je estetický. Primitiv `\accent` umisťuje diakritické znaménko na geometrický střed, což nevypadá vždy hezky. Závažnější je však skutečnost, že takto zapsaná slova neumí TEX dělit. Oba problémy lze řešit virtuálním fontem. TEX podle metrického souboru najde všechny znaky a ve virtuálním fontu můžeme usazení diakritických znamének doladit. Více se o tom dočtete v češtině v článku [4].

Podobný přístup, ale z jiného důvodu, byl kdysi použit při zpracování Zpravodaje. Pro jeho sazbu se používají $\mathcal{CS}$ fonty, které jsou vytvořeny METAFONTem. Pro soubor ve formátu PDF se však bitmapové fonty nehodí, je nutno použít PostScriptové fonty Type1. $\mathcal{CS}$ fonty ve formátu Type1 neexistovaly. Proto byl pro tyto účely použit virtuální font, který mapoval znaky z $\mathcal{CS}$ fontů na standardní Computer Modern, jejichž varianty Type1 byly volně dostupné. Virtuální font byl ovšem použit pouze pro účely převodu do PDF. Mohli jsme si to dovolit, protože virtuální fonty měly identickou metriku jako skutečné $\mathcal{CS}$ fonty.

Triky s fonty

Virtuální font umožňuje rozšířit možnosti TEXu. Klasickým případem, s nímž si TEX neumí poradit, je prostrkaný text. Existuje sice několik řešení na úrovni maker, jmenujme např. makro `\prostrkej` od Petra Olšáka [3], ale tyto metody se hodí spíše na samostatná slova, nejlépe v titulcích. Pokud bychom takto měli sázet větší úsek textu, potřebujeme jiný přístup. Vytvoříme si prostrkaný virtuální font. Metrický soubor TFM lze programem `tftopl` převést na textový soubor PL (Property List). Jeho formát se hodně podobá formátu VPL. Prostrkaný virtuální font si můžeme proto vytvořit jednoduchým prográmkem. Je napsán v Perlu, protože programy v tomto jazyce se snadno a rychle ladí a interpret existuje pro všechny platformy. Zde je výpis programu `spaced.pl`:

```
#!/usr/bin/perl
if ($#ARGV == 1) { $spacing = $ARGV[1]; }
else { $spacing = .1 }
if ($ARGV[0] =~ /\.pl$/) {
  $fontname = $‘;
  open(PL, $ARGV[0]) or die ’Cannot open ’ . $ARGV[0];
}
else { die ’Invalid font name’; }
$c = 0;
```

```
print "(VTITLE spaced version of $fontname)\n";
while (<PL>) {
  chop;
  next if /\(LIG C [fli]/;
  $_ = '   (SPACE R .4)' if /\(SPACE/;
  $_ = '   (STRETCH R .1)' if /\(STRETCH/;
  $_ = '   (SHRINK R .06)' if /\(SHRINK/;
  if (/^\((DESIGNSIZE.+)\)$/) {
    $dsize = "(FONT$1)";
  }
  if (/^\(CHARACTER\s+/) {
    $c = $';
  }
  if ($c && /\(CHARWD R (.+)\)$/) {
    $wd = $spacing + $1;
    $_ = "   (CHARWD R $wd)";
  }
  if ($c && $_ eq '   )') {
    print<<"EOMAP";
   (MAP
      (SETCHAR $c)
      )
EOMAP
    $c = 0;
  }
  print "$_\n";
  if (/^\((CHECKSUM.*)\)$/) {
    $checksum = "(FONT$1)";
    print<<"EOMAP";
(MAPFONT D 0
   (FONTNAME $fontname)
   $checksum
   (FONTAT R 1.0)
   $dsize
   )
EOMAP
  }
}
```

Program má dva parametry. První je povinný a obsahuje jméno soouboru PL včetně přípony. Druhý, nepovinný parametr specifikuje požadovaný proklad v jednotkách DESIGNSIZE. Standardně použijeme hodnotu 0.1. Při převodu pro-

vádíme několik činností. Nejprve si zapíšeme `VTITLE`. Je to pouze nepovinný komentář, ale je vhodné si poznačit, co jsme vytvořili. Prostrkaný font nesmí obsahovat ligatury. Proto odstraníme záznamy `LIG`, pokud je druhým znakem některý z množiny f, l, i. Ponecháme ligatury, jimiž se vytvářejí dlouhé pomlčky. Prostrkaný text by měl mít větší mezislovní mezery, než se používají v běžném textu. Změníme proto hodnoty parametrů `SPACE`, `STRETCH` a `SHRINK`, jež určují mezeru, její roztažitelnost a stažitelnost. Tyto hodnoty ovšem lze modifikovat dodatečně TEXovým primitivem `\fontdimen`, případně můžeme velikost mezislovní mezery nastavit pomocí `\spaceskip`. Hodnotu `DESIGNSIZE` si pouze zapamatujeme, abychom ji ve vhodný okamžik mohli zkopírovat do `FONTDESIGNSIZE`. Jakmile přečteme `CHECKSUM`, můžeme zapsat informaci o použitém fontu `MAPFONT`. Zmíníme se o hodnotě `CHECKSUM`. Je to kontrolní součet, který je uložen v metrickém souboru TFM, TEX jej zapíše do DVI a METAFONT jej zapisuje i do vlastního bitmapového fontu. Nezáleží na způsobu, jak je tento kontrolní součet vypočítán. Důležité je pouze to, aby na všech místech byla stejná hodnota. Ovladač podle toho zkontroluje, zda používá font, jehož metriku použil TEX při sazbě. Při tvorbě virtuálního fontu musíme tedy uvést správnou hodnotu `FONTCHECKSUM`, na hodnotě kontrolního součtu virtuálního fontu nezáleží. Můžeme použít stejnou hodnotu nebo třeba nulu. Hlavní činnost provedeme při zpracování vlastnosti `CHARACTER`. Kód znaku si uložíme pro pozdější použití. Vlastnost `CHARWD` obsahuje šířku znaku. Tu musíme zvětšit o požadovaný proklad. Narazíme-li na ukončovací pravou závorku, zapíšeme instrukce, jak se má znak vytvořit. Vysadíme znak se stejným kódem z implicitního fontu. Proto nemusíme uvádět `SELECTFONT` a `SETCHAR` bude obsahovat kód znaku, který jsme si uložili. Program jsme aplikovali na font csr10. Použili jsme příkaz:

```
spaced.pl csr10.pl > scsr10.vpl
```

Nyní již můžeme sázet proloženě. Samozřejmě jsme virtual property list museli překompilovat na virtuální font a soubory nakopírovat do správných adresářů. Protože pracujeme s osmibitovým fontem, funguje i dělení slov. Ve fontu jsme ponechali všechny kerningy a definice ligatur pro pomlčky. Skutečně – pomlčka na půlčtverčík funguje a dokonce — máme i dlouhou čtverčíkovou pomlčku.

Druhým problémovým případem je podtrhávání. Řešení pomocí maker má svá omezení. Vytvoříme si tedy virtuální font programem `underline.pl`:

```
#!/usr/bin/perl
$overlap = .05;  $ulpos = .24;  $ulheight = .04; # defaults
$overlap = $ARGV[1] if $#ARGV > 0;
$ulpos = $ARGV[2] if $#ARGV > 1;
$ulheight = $ARGV[3] if $#ARGV > 2;
$ovstring = sprintf("%0.4f", -$overlap); # must be fixed real
```

```
$ulht = sprintf("%0.4f", $ulheight);
if ($ARGV[0] =~ /\.pl$/) {
  $fontname = $`;
  open(PL, $ARGV[0]) or die 'Cannot open ' . $ARGV[0];
}
else {
  die 'Invalid font name';
}
$c = $f = 0;
print "(VTITLE underlined version of $fontname)\n";
while (<PL>) {
  chop;
  if (/^\((DESIGNSIZE.+)\)$/) {
    $dsize = "(FONT$1)";
  }
  $f = 1 if /^\(FONTDIMEN/;
  if (/^\(CHARACTER\s+/) {
    $c = $';
  }
  if ($c && /\(CHARWD R (.+)\)$/) {
    $wd = 2 * $overlap + $1;
  }
  if ($c && $_ eq '   )') {
    print<<"EOMAP";
   (MAP
      (PUSH)
      (MOVEDOWN R $ulpos)
      (MOVERIGHT R $ovstring)
      (SETRULE R $ulht R $wd)
      (POP)
      (SETCHAR $c)
      )
EOMAP
    $c = 0;
  }
  if ($f && $_ eq '   )') {
    $f = 0;
    $th = $ulheight - $ulpos;
    print<<"EOMAP";
   (DEFAULTRULETHICKNESS R $th)
   (BIGOPSPACING1 R $ulpos)
EOMAP
```

```
  }
  print "$_\n";
  if (/^\((CHECKSUM.*)\)$/) {
    $checksum = "(FONT$1)";
    print<<"EOMAP";
(MAPFONT D 0
   (FONTNAME $fontname)
   $checksum
   (FONTAT R 1.0)
   $dsize
   )
EOMAP
  }
}
```

Struktura programu je velmi podobná, popíšeme si tedy jen odchylky. Program nyní má čtyři parametry, z nichž pouze první, obsahující jméno původního fontu, je povinný. Dalšími parametry definujeme polohu podtrhávací linky, její tloušťku a přesah. Mezi některé dvojice písmen se totiž vkládá kladný kerning. Přesah slouží k tomu, aby linka nebyla přerušena. Parametry mají standardní hodnoty. Možná vás zaujme použití funkce `sprintf`. Je-li hodnota čísla menší než 0.1, Perl ji zapíše ve vědecké notaci. Takové číslo však program `vptovf` neumí přečíst, proto si proměnné, kde se dá očekávat malá hodnota, naformátujeme sami. Zajímavá jsou dvě místa programu. Při vlastní sazbě znaku si nejprve příkazem `PUSH` uložíme aktuální bod sazby, pak vysadíme linku, jejíž délku jsem vypočetli z šířky znaku, pak se příkazem `POP` vrátíme na původní bod sazby a vysadíme znak. Na konci programu modifikujeme sekci `FONTDIMEN`. Paramery `DEFAULTRULETHICKNESS` a `BIGOPSPACING1` jsou v TEXu dostupné jako `\fontdimen` 8 a 9. Ukládáme si do nich informaci o podtrhávací lince. Analogicky si vytvoříme font ucsr10. Nyní již můžeme sázet podtrženě, ale ještě to není úplně ono. Nepodtrhávají se mezery.

V této části jsme udělali tři změny. Především sázíme text do dvou sloupců a dovolíme TeXu, aby na nový řádek oddělil i jen dvě písmena. Můžeme si tak lépe demonstrovat, že TeX dělí podtržená slova. Druhou změnou je zvětšení řádkového prokladu o 15%, aby text nebyl příliš slitý. Poslední úpravou jsme dosáhli toho, že se podtrhávají i mezery. Ve virtuálním fontu nelze podtrženou mezeru vytvořit, protože TeX mezeru jako znak nepoužívá. Velikost mezislovních mezer se vezme z parametrů, o nichž jsme se zmínili při vytváření prostrkaného fontu. Jedinou možností, jak mezery podtrhnout,

je vytvoření vhodného makra. Postup si popíšeme již v normálním, nepodtrženém textu. Všimněte si, že řešení není dokonalé, logo TEX se podtrhává špatně, protože písmeno E zasahuje pod řádek.

Definici provedeme uvnitř prostředí `multicols`, některé změny budou lokální. Prvními kroky v definici je nastavení limitu, aby logo TEX, použité na konci dvousloupcové sazby, nerozhodilo řádkový rejstřík. Dále zvětšíme řádkový proklad, přepneme na podtržený font a povolíme dělení dvě písmena před koncem slova.

```
\lineskiplimit \minus5dd
\linespread{1.15}\selectfont
\underlined \righthyphenmin2
```

Nyní si nadefinujeme makro pro podtrženou mezeru:

```
\def\hrulespace{\leaders\hrule height\fontdimen8\the\font
                                depth\fontdimen9\the\font
    \hskip\fontdimen2\the\font plus \fontdimen3\the\font
                                minus \fontdimen4\the\font}
```

Makro definujeme pomocí `\leaders`. Tento primitiv se chová stejně jako *glue*. Je to odstranitelný element a je v něm povolen zlom, pokud mu nepředchází jiný odstranitelný element. Za řídícím slovem `\leaders` následuje *rule* nebo *box*, druhým objektem je mezera. Začněme vysvětlování odzadu. Velikost mezislovní mezery je uložena v metrice v parametru `\fontdimen2`, `\fontdimen` s čísly 3 a 4 obsahují roztažitelnost a stažitelnost. Použitím těchto hodnot v primitivu `\hskip` dosáhneme stejných výsledků, jako kdybychom použili nepodtržený font, z něhož jsme je do virtuálního fontu převzali. Při popisu programu `underline.pl` jsme si řekli, že polohu a tloušťku podtrhávací linky uložíme do parametrů `\fontdimen` 8 a 9. Ve skutečnosti `\fontdimen8` neobsahuje tloušťku linky, ale tloušťku zmenšenou o polohu linky. Přesně tuto hodnotu očekává `\hrule` za klíčovým slovem `height`. Tím jsme zajistili, že mezera bude podtržena stejnou linkou jako ostatní znaky.

Bylo by nepohodlné, kdybychom `\hrulespace` museli do každé mezislovní mezery zapisovat ručně. Mezeru proto uděláme aktivní a vytvoříme vhodnou definici. Aktivní mezera je však dvojsečná zbraň. Abychom nemuseli dávat při definicích zvlášť velký pozor, použijeme trik. Vykřičníku nastavíme, že odpovídající malé písmeno je mezera. Dále učiníme vykřičník aktivním. Celou definici pak vložíme do `\lowercase`. Tento primitiv převede všechna písmena na malá, ale zachová jejich kategorii. Z vykřičníku se tedy stane aktivní mezera:

```
{\lccode`\!32
```

```
\catcode`\!13
\lowercase{%
\gdef!{\ifhmode\expandafter\checkspace\fi}
\gdef\checkspace{\futurelet\next\maybespace}
\gdef\maybespace{%
  \let\hsp\hrulespace
  \ifx!\next \let\hsp\empty
  \fi
  \hsp}
}}
\def~{\nobreak\hrulespace}
```

Mezery chceme podtrhávat pouze v horizontálním režimu. Jsme-li v jiném režimu, neprovedeme nic. V textu můžeme mít více mezer za sebou. Za normálních okolností je vstupní procesor TEXu nahradí jedinou mezerou, ale neprovede to v okamžiku, až bude mezera aktivní. Proto se pomocí `\futurelet` podíváme na následující token. Je-li jím aktivní mezera, neprovedeme nic, v opačném případě vložíme podtrženou mezeru. Pak si ještě nadefinujeme vlnku tak, aby vkládala nezlomitelnou podtrženou mezeru.

Text odstavce se může nacházet na několika řádcích. Prázdným řádkem je odstavec ukončen. Pokud bychom do tohoto mechanismu nezasáhli, přechod na nový řádek ve zdrojovém souboru by vytvořil nepodtrženou mezeru. Ošetříme to následujícím způsobem:

```
\catcode`\^^M13 %
\def^^M{\ifhmode\expandafter\checkpar\fi}%
\def\checkpar{\futurelet\next\maybepar}%
\def\maybepar{%
  \let\hsp\hrulespace %
  \ifx^^M\next \let\hsp\par %
  \fi %
  \hsp}
```

Znak konce řádku jsme nejprve učinili aktivním. Od tohoto okamžiku musíme všechny řádky ukončovat procentem, jinak se TEX promění v zuřivého lva, který nám přeplní zásobník. Opět nebudeme provádět žádnou činnost v jiném než horizontálním režimu. Zvláštní ošetření vyžaduje posloupnost dvou konců řádků, proto se opět podíváme na následující token. Následuje-li konec řádku, místo sazby podtržené mezery ukončíme odstavec. Za poslední pravou závorkou definice makra `\maybepar` již procento nepotřebujeme. Zbývá už jen změnit kategorii mezery pomocí `\catcode`\ 13` a můžeme tisknout podtrženě. Makro není zcela robustní. Nejsou ošetřeny mezery na začátku a na konci řádku. V anglické sazbě se nevloží delší mezera za tečku, protože nekontrolujeme hodnotu `\sfcode`. Dokonalejší makro si můžete udělat za domácí cvičení.

Další problém je v uvedených perlovských skriptech. Neimplementují plný parser souborů ve formátu property list, ale jsou pevně svázány s výstupem z programu `tftopl`.

Překódování fontu

Pro češtinu a slovenštinu se používá několik různých kódování. Může se tedy stát, že máme sice font s akcentovanými znaky, ale v jiném kódování, než vyžaduje TEX. Na úrovni PostScriptu lze font překódovat operátorem `ReEncodeFont`. Jinou metodu, která však není zcela ekvivalentní s předchozí[3], nabízí virtuální font. Pro příklad nemusíme chodit daleko. Představme si, že chceme používat mezinárodní LATEX, jehož fonty mají kódování T1, ale typograficky se nám více líbí $\mathcal{CS}$ fonty s kódováním IL2. Můžeme si proto vytvořit virtuální font, jehož kódování bude T1, ale bude odkazovat na znaky z $\mathcal{CS}$ fontu. Provedeme to v několika krocích. Můžeme začít podobným perlovským skriptem, jaký byl popsán v předchozí kapitole. Nebudeme upravovat parametry fontu ani rozměry znaků, jen jejich kód. Například písmeno Č má v IL2 oktalový kód 310, v T1 má oktalový kód 203. Při konverzi tedy všechny výskyty řetězce `O 310` nahradíme řetězcem `O 203`. Do popisu znaku Č pak vložíme instrukci `(SETCHAR O 310)`. Podobně naložíme s ostatními znaky, jejichž kód se liší. Pokud chceme psát pouze česky a slovensky, můžeme být s výsledkem spokojeni. Kódování T1 však obsahuje znaky, které v $\mathcal{CS}$ fontech nejsou. Při jejich vytváření nám pomůže qdTEXvpl. Znaky s diakritickými znaménky, např. španělské ñ, vytvoříme snadno pomocí `\~n`. Některé znaky však v $\mathcal{CS}$ fontech nemáme, příkladem je Đ. Vezmeme si jej z odpovídajícího DC fontu. Oba VPL nyní zkombinujeme. Ve třetím kroku musíme doplnit kerningy. Můžeme je zkopírovat z $\mathcal{CS}$ fontu od podobně vypadajících znaků, nebo je převezmeme z DC fontu. Čtvrtým krokem bude doladění polohy akcentů a kerningů. Návod vypadá na první pohled jednoduše, ale mohou nás potkat problémy, které popisuje Petr Sojka [4].

Literatura

V této kapitole uvádíme seznam základní české i cizojazyčné literatury, která se problematice virtuálních fontů zabývá. Kniha [3] se sice virtuálním fontům nevěnuje, ale ohsahuje popisy formátů souborů TFM a DVI.

1. Donald Knuth: *Virtual Fonts: More Fun for Grand Wizards.* TUGboat (11) 1990, No 1 (April), pp. 13–23.

[3] PostScriptový font může obsahovat více než 256 znaků. Pomocí `ReEncodeFont` lze přidělit kódy i těm znakům, které v původním fontu žádný číselný kód neměly. Pomocí virtuálního fontu to možné není.

2. Petr Olšák: **Typografický systém TEX**, 2. vydání. Konvoj, Brno 2000. ISBN 80-85615-91-6.
3. Petr Olšák: **TEXbook naruby.** Konvoj, Brno 1997. ISBN 80-85615-64-9.
4. Petr Sojka: *Virtuální fonty, accents a přátelé.* Zpravodaj Československého sdružení uživatelů TEXu, **4** (2), 56–69 (1994).
5. Jiří Zlatuška: *Automatic generation of virtual fonts with accented letters for TEX.* Cahiers GUTenberg No. 10, září 1991.

Summary: Anatomy of Virtual Fonts

The article is a brief introduction to the concept of virtual fonts. It is first explained how TEX works with fonts. Afterwards a simple tool for building a virtual font, namely qdTEXvpl, is presented. Finally usage of virtual fonts is demonstrated by typesetting spaced and underlined text. The macros and Perl scripts described in this article are available from the web page of the Bulletin.

Wordové plug-iny související s TEXem aneb Možnosti a schopnosti produktů Word2TeX a TeX2Word

Aleš Pavelka

Úvod

V současné době kdy většina uživatelů používá produkty firmy *Microsoft*, vzniká tím vedlejší efekt nutící uživatele k vzájemné dohodě na formátu, ve kterém zpracovávají své dokumenty. Je smutné, že ne vždy se lidský vývoj ubírá tím „nejlepším“ směrem a nevybírá to nejlepší pro člověka. Musíme si otevřeně přiznat, že pro nezkušeného nebo začínajícího uživatele počítače je poněkud jednodušší napsat a upravit nějaký dokument v prostředí *MS Wordu* než v TEXu či LATEXu. Tento problém je však daleko širší a kdo ví, kdy bude vyřešen. Se stavem takového dvojího světa vzniká i otázka: „Existuje kvalitní a rychlá možnost přechodu mezi těmito formáty?“ Při hledání odpovědi na tuto otázku nám pomohou konvertory, pluginy či dodatky *MS Wordu* – produkty Word2TeX a TeX2Word.

Článek původně vznikl jako semestrální práce předmětu „Publikační systém TEX“

Požadavky pro instalaci a podmínky testování

Pro používání produktů Word2TeX™ a TeX2Word™ firmy *Chikrii Softlab* potřebujeme mimo operačního systému *Microsoft Windows* a textového editoru *Microsoft Word* také speciální editor rovnic MathType™ firmy *Design Science*.

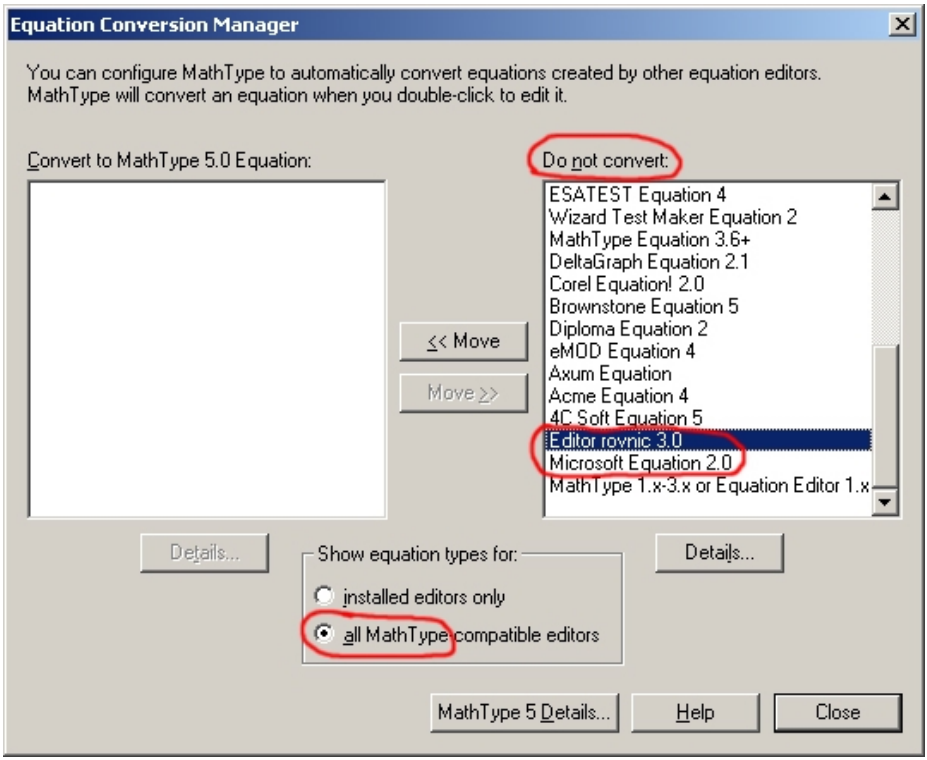
Testování vlastností konvertorů probíhalo na operačním systému *Windows XP*, s textovým editorem *Microsoft Word 2002*. Z domovské stránky *Chikrii Softlab* byly staženy 30denní demo verze programů Word2TeX 3.0 a TeX2Word 2.0. Stejné licenční podmínky mělo i demo editoru rovnic MathType 5.1.

Vlastnosti plug-inů

MathType

Pokud chceme psát matematický text, vzorce a rovnice v prostředí *MS Wordu*, tak si při jeho instalaci můžeme zvolit podporu pro psaní rovnic *Equation Editor* (v české lokalizaci *Editor rovnic*). Tento editor v mnoha případech zcela dostačuje. Náročnější uživatelé mohou zvolit produkt MathType, který je pro správný chod pluginů Word2Tex a TeX2Word nutný. Pro ilustraci rozdílů mezi oběma matematickými editory uvádím jejich porovnání v tabulce 1 převzaté z oficiální domovské stránky `http://www.mathtype.com/en/products/mathtype/win/mt_vs_ee.htm`. Pokud si uživatel tento produkt nezakoupí za $129 (pro školy $99), pluginy budou fungovat i po uplynutí 30denní testovací lhůty. Po této době nezakoupený program můžeme přepnout do *Lite* módu, který však podle mne má o něco méně funkcí než původní *Editor rovnic*. Pokud bychom chtěli používat současně *Editor rovnic* a *MathType*, bylo by třeba změnit nastavení v *Equation Conversion Manageru* (obr. 1), kde máme možnost ovládat konverzi rovnic z formátu původního matematického editoru na formát (objekt) MathType. Konverze již napsaných rovnic v jiném matematickém editoru do formátu MathType je jednostranná. Rovnice napsaná nebo pozměněná v prostředí MathType je po jeho odinstalovaní prakticky již needitovatelná, je zní jen pouhý objekt. Určitou zajímavostí je odkaz na produkt MathType přímo v nápovědě *Equation Editoru*. Vlastní instalace tohoto doplňku je velmi snadná, typicky taková na jakou jsou uživatelé prostředí Windows zvyklí.

Na obrázcích 2 a 3 můžeme srovnat vzhled obou matematických editorů. Prostředí MathType je lépe vizuálně zpracované, je bližší uživateli. Logika psaní vzorců je však u obou produktů stejná. Díky lepšímu grafickému zpracování je rovnice v prostředí MathType napsaná přece jen o něco efektivněji.

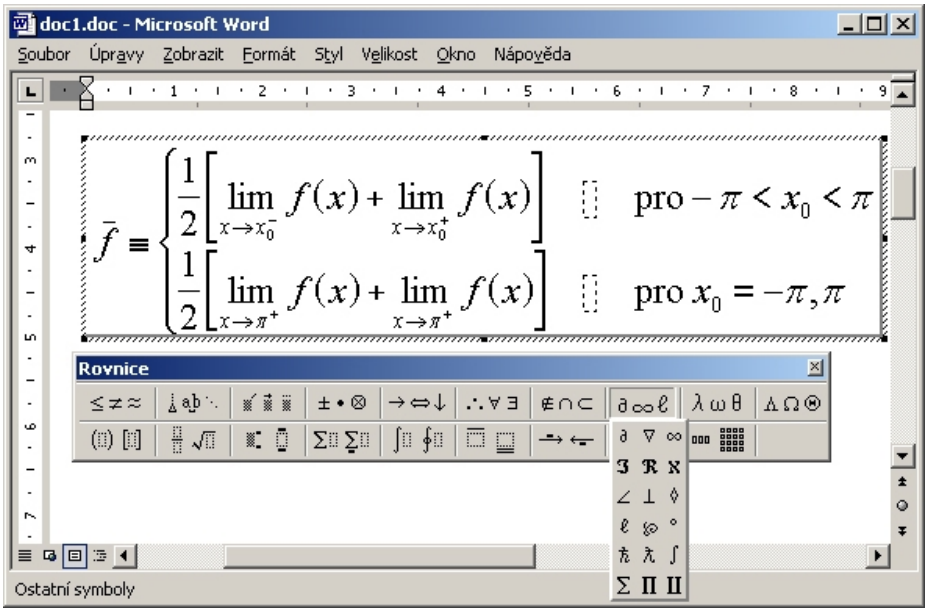


Obrázek 1: Změna nastavení v *ECM* pro používání více matematických editorů současně

Word2TeX

Pro testování konvertoru Word2TeX byl použit reálný dokument, existovala i možnost vytvořit speciální testovací dokument, ale pro jeho „umělost“ a nedokonalost (určitě by nebyl schopen postihnout paletu možností textového editoru, TEXu či MS Wordu) byla tato možnost zavrhnuta. Převáděný dokument byl původně napsán v LATEXu (je použit při testování produktu TeX2Word), ale z určitých důvodů již před časem „ručně“ přepsán do Wordu. Nyní je automaticky převáděn zpět do LATEXu. Pro možnosti osobního porovnání jsou všechny použité soubory přiloženy, včetně `*.doc` souborů, které jsou převedeny pomocí Adobe Acrobatu do formátu PDF.

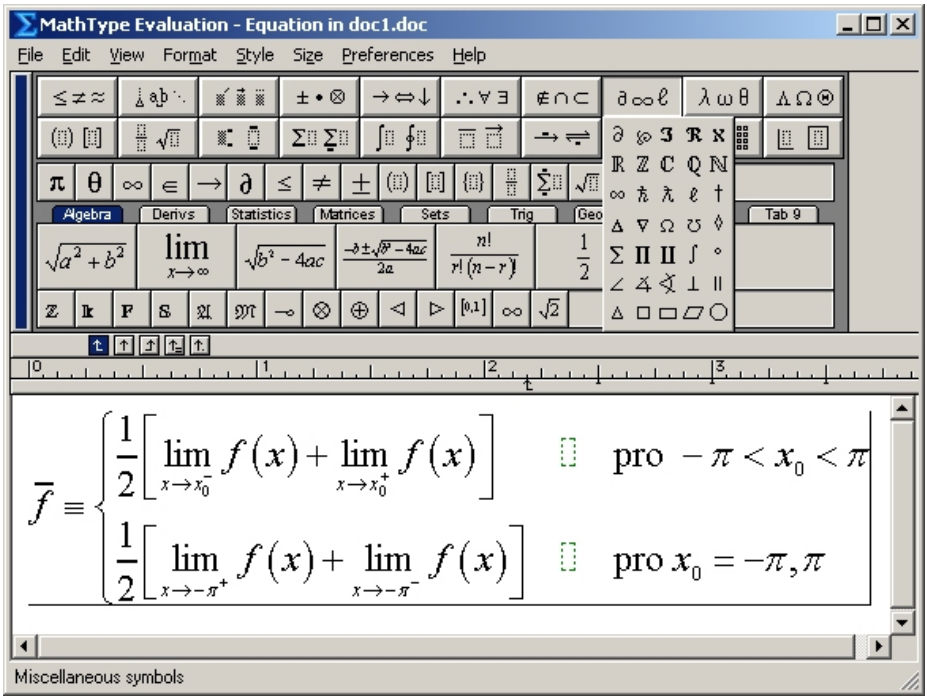
Vlastní ovládání konverze je velmi jednoduché – dokument ve Wordu je uložen jako `*.tex`, ale zároveň je zde ještě možnost upravit parametry převodu (viz. obr. 4 až 8). Převod netrvá dlouho, podle nastavení lze dostat `*.tex` sou-



Obrázek 2: Editor rovnic v prostředí MS Wordu

bor a případně i `*.eps` soubory představující obrázky. Testovaná zkušební verze programu Word2TeX je omezena na převod jednoho (prvního) obrázku, jedné (první) tabulky a prvních sedmi rovnic v každém převáděném dokumentu.

Z přiložených dokumentů může čtenář sám posoudit kvalitu převodu. Dle mého názoru je kvalita převodu textu velmi dobrá. Jsou dodrženy velikosti písem, jejich řez, hierarchické členění dokumentu, trochu pokulhává jen číslování sekcí. Složitější typ formátu (dvousloupcová sazba) nebyl dodržen vůbec. Tabulka a rovnice byly převedeny vcelku dobře, za výborné bych označil dodržení automatického číslování rovnic. Velmi špatná je situace s literaturou, která byla převedena, řekněme pouze „otrocky“. Na druhou stranu literatura v prostředí MS Wordu je vlastně jen obyčejný zformátovaný text, takže ji konvertor nemá šanci odlišit od ostatního textu. Není zdaleka tak automatizovaná jako v TEXu prostřednictvím `*.bib` souborů. Jedinou automatizací je relativní odkazování na literaturu v textu. Bohužel právě toto odkazování převede Word2TeX bez vazeb, odkazy jsou jen čísla. Lehce komplikovaná je situace kolem převodu češtiny. Do finálního dokumentu LATEXu jsou české znaky převedeny dobře, ale překlad do DVI pomocí příkazu `latex` či `cslatex` bohužel vynechá ě, č, ř, ů, ň, ť a ď. Pro zobrazení těchto znaků musí být doplněn ve zdrojovém `*.tex` souboru v příkaz `\usepackage{czech}` a přeložíme `cslatex`em.



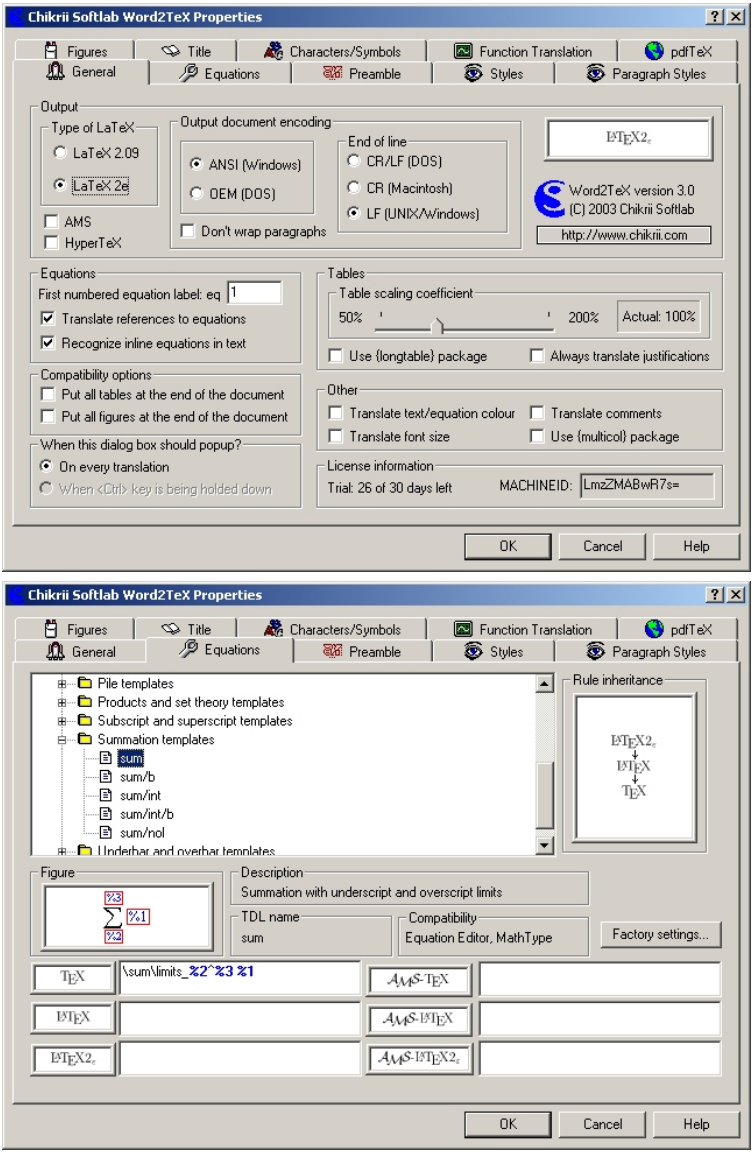
Obrázek 3: Prostředí editoru MathType jako pluginu MS Wordu

TeX2Word

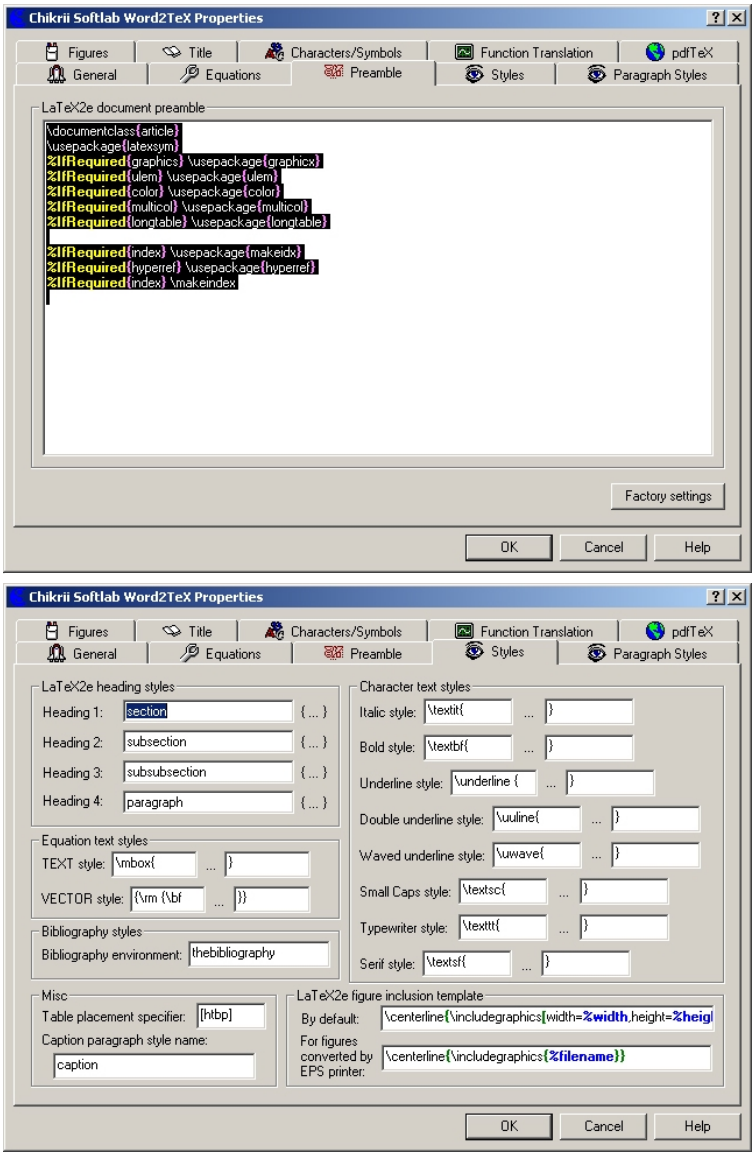
Pro testování konvertoru TeX2Word byl použit reálný dokument napsaný v prostředí LaTeXu. Pro konverzi je nutné, aby konvertovaný dokument byl jen jeden. Nelze proto používat praktický systém do sebe vnořených dokumentů pomocí příkazu `\input{file}`. Byl proto vytvořen jeden velký dokument, který zahrnuje i seznam literatury. Výsledný `*.doc` soubor byl opět převeden pro účely jednoduššího náhledu pomocí Adobe Acrobatu do formátu PDF.

Konverze TeXu do Wordu je uživatelsky velmi snadná, stačí jen požadovaný `*.tex` soubor otevřít ve Wordu, proběhne konverze (obr. 9) a její průběh (resp. chybové hlášky) je zaznamenán do souboru `*.plog`.

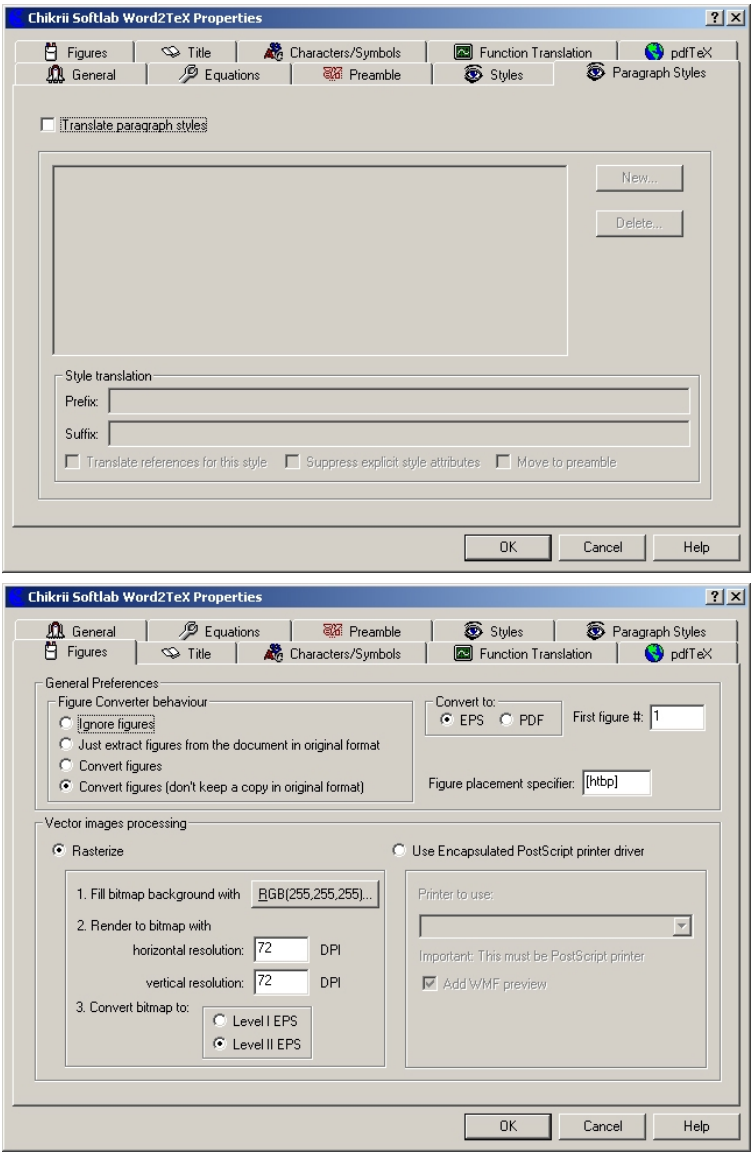
Kvalita převodu i v tomto směru je velmi dobrá, i když má některé nedostatky. Jako první nás samozřejmě upoutá opětovná nedokonalost v převedení dvousloupcové sazby, absence obrázků, které nepřevádí ani plná verze (30-ti denní demo má omezení v převodu prvních 100 rovnic v každém převáděném dokumentu). Ale vkládání obrázků do Wordu je zase jiná kapitola, dobré je, že místo pro obrázky je dobře viditelné a je zde zaznamenán i název příslušného



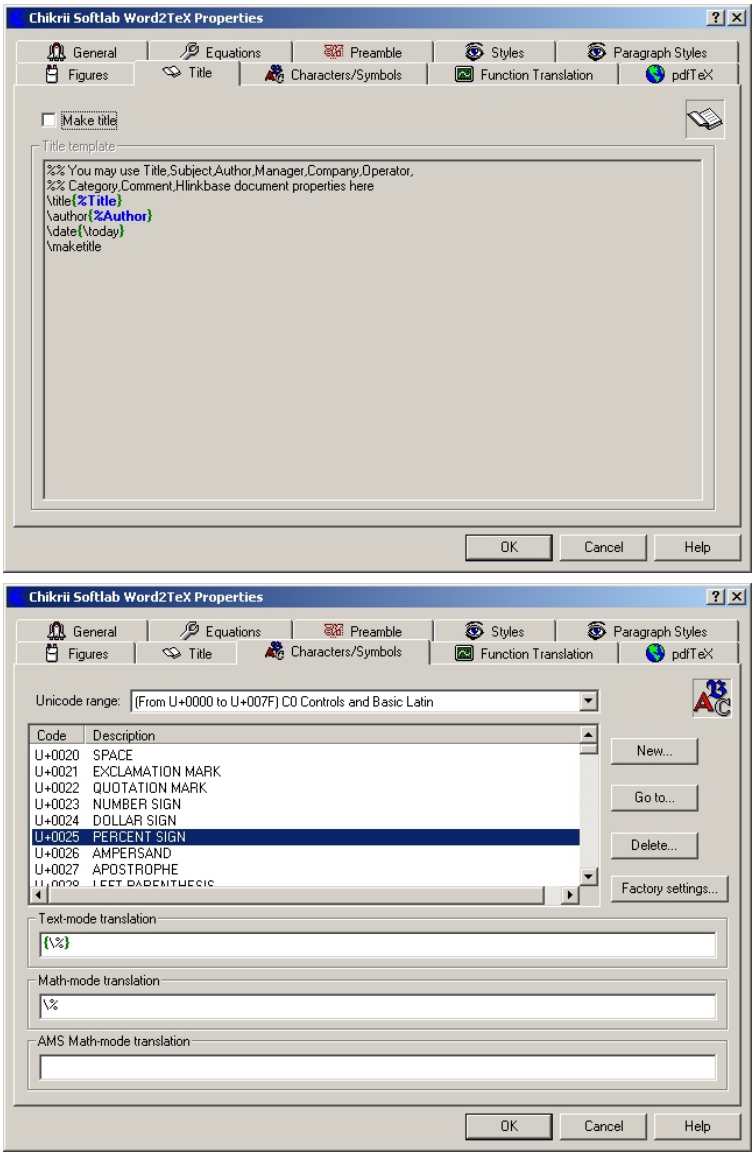
Obrázek 4: Možnosti nastavení parametrů pluginu Word2TeX



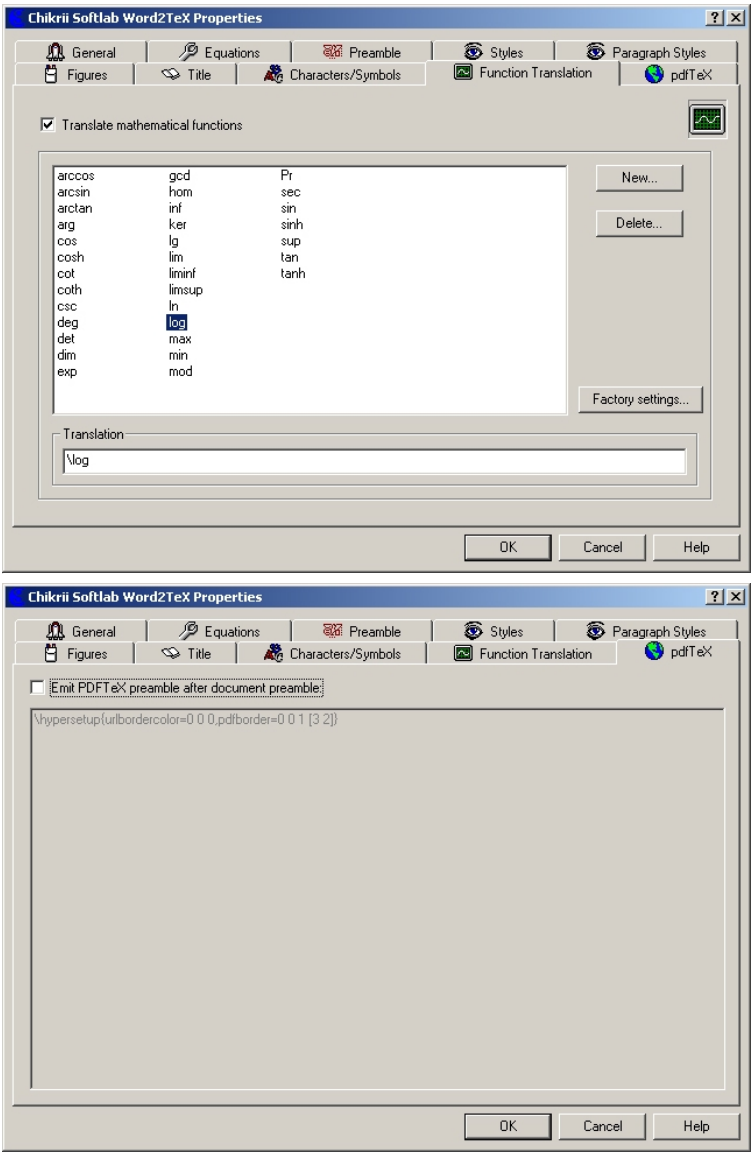
Obrázek 5: Možnosti nastavení parametrů pluginu Word2TeX (pokračování)



Obrázek 6: Možnosti nastavení parametrů pluginu Word2TeX (pokračování)



Obrázek 7: Možnosti nastavení parametrů pluginu Word2TeX (pokračování)



Obrázek 8: Možnosti nastavení parametrů pluginu Word2TeX (pokračování)

Tabulka 1: MathType vs. Equation Editor

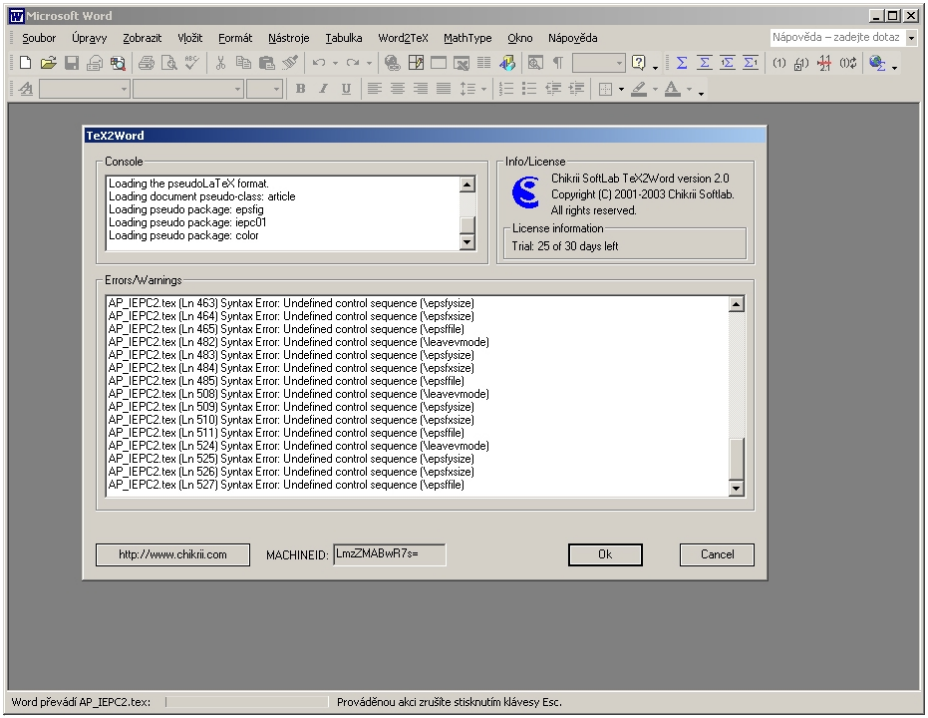
Vlastnost	MathType	Equation Editor
Počet matematických symbolů a šablon	500+	275
Tvorba matematických WWW stránek	Ano	Ne
Vlastní tlačítková lišta rovnic	Ano	Ne
Podpora CMYK, RGB barev	Ano	Ne
Automatická změna mezer, velikosti fontu a stylu rovnic v dokumentu MS Word	Ano	Ne
Číslování rovnic a odkazy v MS Wordu	Automatické	Ruční
Vlastní klávesové zkratky	Ano	Ne
TeX, LaTeX, AMS-TeX, AMS-LaTeX výstup	Ano	Ne
MathML 1.0 a 2.0 výstup	Ano	Ne
Font euclidovské matematiky	Ano	Ne
Dávkový export rovnic v MS Word doc do souborů GIF, EPS a WMF	Ano	Ne
Uložení rovnic jako souboru GIF s nebo bez anti-aliasingu	Ano	Ne
Uložení rovnic v EPS souboru	Ano	Ne
Uložení fontu, stylu a velikosti nastavení do souboru	Ano	Ne
Podpora mezinárodních znaků a klávesnic	Ano	Ano
Přidání/smazání řádku a sloupce v matici	Ano	Ne
Nelimitované undo and redo	Ano	Ne
Popis znaku založený na unikódu	Ano	Ne
Formátovací pravítko s tabulátorovými značkami	Ano	Ne
Kontrola vyrovnání závorek	Ano	Ne
Modifikovatelné rozeznávání funkcí (sin, cos, atp.)	Ano	Ne
Ovládání velikosti tlačítkové lišty a pracovního prostředí	Ano	Ne
Programovatelné překlady pro vlastní SGML/XML řešení	Ano	Ne
DLL rozhraní pro speciální úlohy	Ano	Ne
Hierarchický pohled na strukturu rovnice	Ano	Ne
Online návody	Rozsáhlé	Ano
Uživatelský manuál (tištěný a v PDF)	Ano	Ne
Technická podpora	Ano	???

Zdroj: Internet – `http://www.mathtype.com/en/products/mathtype/win/mt_vs_ee.htm`

souboru s obrázkem. U převáděných rovnic jsem nenašel mnoho chyb, snad jen u převodu znaků pro absolutní hodnotu a občas, když byla pozice rovnice nějak speciálně upravována, je do rovnice zahrnuta i hodnota této vzdálenosti. Formát literatury i odkazy jsou převedeny dobře, odkazy v textu i v seznamu literatury jsou opět jen čísla, stejně tomu je i při odkazech na čísla rovnic a obrázků. Dobrý výsledek jsme dostali i při převodu tabulky a českých znaků.

Závěr

Oba dva konvertory si počínají velmi dobře, jejich výsledky sice nejsou stoprocentní, ale v mnoha případech mohou uživatelům ušetřit spoustu času a práce při převodu z jednoho formátu na druhý. Za nejvýznamnější bych považoval kvalitní převod matematických rovnic, jejichž psaní zabírá relativně dlouhý čas a je velmi náchylné na překlepy. Oba konvertory také dobře dodržují hierarchické uspořádání textu (chapter, section, subsection, paragraph, ...). Problémy s převodem obrázků jsou významné a tato část by si určitě vyžadovala zlepšení. Na



Obrázek 9: Prostředí editoru MathType jako pluginu MS Wordu

druhou stranu to není zase až tak závažný problém. Závěrem musím přiznat, že mě kvalita obou převodů příjemně překvapila a že testované produktu jsou na vysoké úrovni.

WWW odkazy

http://www.word2tex.com Word2Tex, TeX2Word, Chikrii Softlab
http://www.mathtype.com MathType, Design Science

Přiložené soubory

tex/	...	převod z TEXu do Wordu
tex/IEPC2.tex	...	zdrojový dokument převodu z TEXu do Wordu
tex/IEPC2.doc	...	cílový dokument převodu z TEXu do Wordu
tex/IEPC2.doc.pdf	...	cílový dokument převodu z TEXu do Wordu zkonvertovaný do PDF
tex/IEPC2.plog	...	log soubor vytvořený při převodu z TEXu do Wordu
tex/iepc01.sty	...	styl dokumentu IEPC2.tex
tex/*.eps	...	EPS obrázky z dokumentu IEPC2.tex
	...	
word/	...	převod z Wordu do TEXu
word/ACHEMA.doc	...	zdrojový dokument převodu z Wordu do TEXu
word/ACHEMA.doc.pdf	...	zdrojový dokument převodu z Wordu do TEXu zkonvertovaný do PDF
word/ACHEMA.tex	...	cílový dokument převodu z Wordu do TEXu
word/*.eps	...	EPS obrázek vytvořený při převodu z Wordu do TEXu

Soubory jsou dostupné z WWW stránky Zpravodaje.

Summary: Word Plug-Ins for TEX: Possibilities and Abilities of the Word2TeX and TeX2Word Products

The article describes two MS Word plug-ins which allow conversion from and to TEX. The documents illustrating the results of conversion are available from the web page of the Bulletin.

TEX a PostScript v grafike programovacích jazykov

Ladislav Bittó

Na svete je viacero programovacích jazykov a väčšina z nich má aj grafickú knižnicu. Máloktorý z nich však umožňuje zápis LATEXovských a PostScriptových príkazov priamo do zdrojového kódu programu napísaného v tomto jazyku. Podľa ankety *Ako TEXujeme* je jasné, že okolo 80 % TEXistov svoje obrázky aj

Ukázky obrázků vytvořených popisovaným programem jsou dostupné z WWW stránky Zpravodaje. Pozn. red.

programuje – čo je potešiteľné – tak ma napadlo opísať Vám spôsob, ako sa mi to podarilo pre programovací jazyk FORTRAN. Vytvoril som balík programov, ktoré z FORTRANovského kódu obsahujúceho LaTeXové a PostScriptové príkazy vytvoria PostScriptové obrázky a LaTeXový zdrojový kód popisu týchto obrázkov spolu s príkazmi na zaradenie obrázkov na príslušné miesta v LaTeXovom kóde. Tesne pred dokončením tohto článku som sa dozvedel, že existuje makro (plaincad.tex), ktoré by umožnilo prekladať tieto obrázky aj plainTeXom.

S týmto článkom Vám chcem priblížiť moje starosti a radosti pri hľadaní spôsobu, ako využiť TeX aj na úplne iný účel než bol určený – na kreslenie obrázkov. K napísaniu nasledujúcich pár strán ma viac-menej vyprovokovala *Anketa užívatelů TEXu* od pána Polácha. Osobne používam TeX na všeličo možné, ale tento grafický balík mi narobil najviac problémov. Vďaka TeXu a PostScriptu ho používam do dnešných dní a fakticky s ním vytváram všetky svoje obrázky.

GPVTV

Ešte dakedy na začiatku 80. rokov minulého storočia vznikol na pôde Výpočtového strediska Slovenskej akadémie vied (VS SAV) súbor podprogramov napísaných v jazyku FORTRAN-77, vhodný pre grafické vyhodnotenie vedecko-technických výpočtov. Dostal názov GPVTV (zrejme skratka **G**rafické **P**odprogramy pre **V**edecko-**T**echnické **V**ýpočty). Celkový počet podprogramov je okolo 150 – od jednoduchých čiar až po konštruovanie a zobrazenie trojrozmerných scén. Posledné prídavky/úpravy sa datujú na začiatok 90. rokov. Pamätám sa, že väčšina programátorov, ktorí chodili počítať do VS SAV tento grafický balík používala. Po objavení sa rôznych grafických programov balík rýchlo strácal na popularite. Bolo to predovšetkým preto, lebo nepoužíval žiadne fonty – obrázky i text boli vektorovo zapísané dvoma príkazmi (MOVE a LINE). Dalo sa síce všetko otáčať, zmenšovať,zväčšovať, skláňať – ale všetko pozostávalo len z jednoduchých čiar. Ja som sa ho tak ľahko nechcel vzdať, lebo som veľmi oceňoval (a dodnes oceňujem) jeho programovateľnosť.

Prekódovanie do LaTeXu

Na prekódovanie výstupu z GPVTV do LaTeXového kódu – s pomocou EMLINE – stačil celkom jednoduchý program. Po preštudovaní štruktúry grafického súboru som bol schopný napísať program, ktorý správne prečítal jeho obsah. Potom stačilo príkazy MOVE a LINE správne rozložiť do príkazov EMLINE v prostredí PICTURE. Aby som vyrobil krajšie obrázky, začal som ich programovať vo FORTRANe bez popisiek, ktoré som potom ručne dopisoval v LaTeXu.

Bolo to trošku nepraktické a hneď som myslel nato, že by bolo výborné, keby som mohol LaTeXové príkazy písať priamo vo FORTRANovskom zdrojáku. Lenže zdrojové kódy balíka GPVTV neboli k dispozícii a ja som chcel, aby grafika LaTeXové príkazy zaregistrovala ako súčasť balíka. Chcel som, aby tie príkazy fungovali ako FORTRANovské premenné, na ktoré môžem aplikovať matematické operátory a mohol ich aj ukladať do indexovaných dátových polí (napr. aby som mohol bez problémov vypísať LaTeXom riadiacu premennú FORTRANovského cyklu).

Pridávanie LaTeXových príkazov

Skúšal som ťažkopádne a krkolomné veci, až nakoniec som prišiel nato, že to pôjde celkom jednoducho. Využijem jeden veľmi krátky a už nepotrebný podprogram z balíku GPVTV, ktorý mal jediný parameter na výber pera pre kresliace zariadenie (DIGIGRAF). Každý nový podprogram označím na začiatku s týmto nepotrebným podprogramom (pre každý nový podprogram iné číslo) a dám ho zapísať do grafického súboru. V prekódovacom programe ich nájdem podľa rôznych čísiel tej istej inštrukcie (výber pera). Potom to už išlo pomerne rýchlo. Vytvoril som si okolo desať podprogramov na zápis textu, čísla, na popis osí (aj logaritmických, dátumových, popisy s opačným smerom, atď.). Obmedzenia DOSu a EMLINE som dokázal obísť a mohol som vytvárať aj obrovské grafické súbory.

Bol som celkom spokojný až nato, že som nemohol pohodlne otáčať popisky. Písal sa rok 1994 a o PostScripte som vedel iba to, že to je v tlačiarňach. Musel som vtedy pristúpiť na vetvenie súboru. Vo FORTRANe som si vytvoril podprogram na vertikálny zápis (stačili mi 90 a 270 stupňové otočenia). Bol taký istý ako ten na horizontálny zápis (veľmi krátky – iba štyri riadky), len mal iné číslo. Vetvenie bolo zariadené až v prekódovacom programe. Keď sa v grafickom súbore našla inštrukcia na výpis vertikálneho textu, tak do LaTeXového súboru sa zapísal príkaz na vloženie obrázka (text1 – 01.pcx, text2 – 02.pcx, text3 – 03.pcx…) a do pomocného súboru sa zapisovali jednotlivé zvislé texty (každý text na samostatnú stranu). Po prekódovaní GPVTV súboru do LaTeXového sa muselo pristúpiť na preklad pomocného súboru a hneď po ňom na výrobu PCX obrázkov s pomocou E. Mattesových (EM) ovládačov. Až po týchto úkonoch prišiel na rad preklad LaTeXového súboru, ktorý sa medzitým vygeneroval. Išlo to celkom svižne aj s viacerými obrázkami v jednom súbore (samostatné strany), hoci cesta z editora FORTRANovského zdrojáku po DVISCR bola celkom kľukatá. Vyzerala takto: editor, preklad FORTRANovského kódu, zlinkovanie a vytvorenie spustiteľného programu, spustenie programu, prekódovanie GPVTV výstupu do LaTeXového kódu, prípadná výroba vertikálnych popisiek, preklad TeXom, DVISCR. Napriek tomu som videl drvivú väčšinu obrázkov v priebehu 2–4 sekúnd.

Takto som teda vytváral svoje obrázky od začiatku roku 1995. Niektoré som vkladal priamo v LaTeXovom kóde (cez príkaz `\input`), ale väčšinou som ich prekódovával do grafického formátu PCX cez EM ovládače. Bol som vcelku spokojný – išlo to celkom svižne, diakritika mi fungovala, mal som v LaTeXovom súbore vložené obrázky s LaTeXovými popiskami, mohol som už zmeniť hrúbku čiary. Aj v GPVTV boli nato príkazy, lenže nikdy sa mi to nepodarilo – iba som to imitoval s viacerými čiarami. Hoci GPVTV umožňoval aj prácu s farbami, do LaTeXu som to vtedy nemohol vložiť (a vtedy neboli ani také požiadavky).

Prekódovanie do PostScriptu cez DVIPS

Prišiel rok 1997 a na TeX konferencii sa množili otázky a odpovede o PostScripte. Dozvedel som sa veľa zaujímavých vecí – ako dostať DVI do PostScriptu, ako vkladať PostScriptové obrázky do LaTeXu, keď chcem dostať do LaTeXu farebné obrázky, tak cez PostScript, atď. Nainštaloval som si DVIPS a mohol som mať aj obrázky v PostScripte. Niekedy v septembri 1998 som si stiahol zo siete krátku príručku – Rudolf Musil: *Základy programování v PostScriptu*. Aby som trošku pochopil PostScript, začal som skúšať jednotlivé základné príkazy, ktoré som chcel zamontovať do obrázkov tak, aby to DVIPS vedel správne vyrobiť. Keď mi to už ako tak išlo, skúsil som vyrábať FORTRANovské podprogramy na PostScriptové príkazy. Tu ma čakalo sklamanie. Okrem nastavenia a zmeny šedosti a farby sa mi nepodarilo donútiť DVIPS, aby správne interpretoval zapísané príkazy cez EMLINE. Snažil som sa, ale po čase som pochopil, že to nepôjde. Videl som nemalé možnosti PostScriptu a tak som si povedal, že to skúsim priamo prekódovať do PostScriptu. Lámal som si dlho hlavu, ako potom tam dostanem LaTeX, ako zosúladím grafiku FORTRANu a PostScriptu s LaTeXom. Lákalo ma to aj preto, lebo by potom DVIPS prekódovával iba LaTeXovú časť obrázkov – bolo by to teda rýchlejšie.

Prekódovanie priamo do PostScriptu

Grafika FORTRANu a LaTeXu sa tvorí z ľavého horného rohu, PostScriptu z ľavého dolného – čo mi veľmi sťažilo situáciu. Grafický kód GPVTV nebol až taký problém – tam sú len dva vykonávajúce príkazy, MOVE a LINE. Horšie bolo to všetko zosúladiť s LaTeXom. Najprv som začal skúšať prekódovanie do príkazov `\special{ps: ...}`. Obrázky sa mi rozladili a ani na viaceré pokusy sa mi nepodarilo týmto spôsobom ich vytvárať. Zase som bol sklamaný, že som to nedokázal. Po dlhšom čase ma napadlo (niekedy pred Vianocami 1999), že čo sa stane keď to skúsim oddeliť – PostScript do jedného súboru a LaTeX do druhého a potom ich dajako spojím. Takže všetky MOVE a LINE

do jedného PostScriptového súboru a výstup z LaTeXových podprogramov do druhého, v ktorom bude aj príkaz na vloženie PostScriptového obrázka. Trošku mi to trvalo kým som to zosúladil. Potom som už mohol pridávať PostScriptové príkazy do FORTRANovského zdrojáku. To už išlo rýchlo a konečne som mohol obdivovať krásu a presnosť PostScriptových obrázkov, ktoré boli zrazu omnoho menšie – zaberali menej miesta na disku.

Hneď som si uvedomil, že ušetrím ešte ďaleko viac, keď sa mi podarí implementovať priame PostScriptové príkazy, ktoré vykonáva až samotný PostScript. Najlepšie to vysvetlím na kruhu. Vo FORTRANe si vyrobím kruh, ktorý vytvorí jeden príkaz MOVE a okolo 360 až 720 (niekedy aj viac) príkazov LINE. Keď sa mi podarí kruh zapísať PostScriptovým príkazom, tak bude stačiť jeden jediný príkaz a to je obrovská úspora. Takisto rôzne trhané čiary (čiarkovaná, bodkovaná., atď). Keď som vo FORTRANe ťahal čiarkovanú čiaru z jedného bodu do druhého, tak to bolo zapísané viacerými príkazmi (podľa dĺžky ich mohlo byť aj viac než 100) MOVE LINE, MOVE LINE. Keď nastavím typ čiary priamo PostScriptovým príkazom, tak to bude len jediné MOVE LINE. Tak som postupne pridával podprogramy na PostScriptové príkazy. S potešením som potom zisťoval, že niektoré súbory sa mi zmenšili až neskutočne, teda aj výroba obrázkov bola rýchlejšia. Nakoniec som pochopil PostScript natoľko (preštudoval som aj článok vo Zpravodaji 1–3 2001 – Arnošt Štědrý: *PostScript pro programátory, vědce i inženýry*), že som mohol naprogramovať aj rotujúci LaTeXový text. V súčasnosti mám vo FORTRANe okolo 20 PostScriptových podprogramov. Môžem zapísať všetky PostScriptové príkazy (niečo ako `\special{....}` v LaTeXu), ale to je pre užívateľa zložitejšie, lebo tento spôsob vyžaduje určité znalosti PostScriptu.

Rýchlosť vytvárania obrázkov

Ako rýchlo vytvoríme obrázok(y) záleží na viacerých faktoroch. Aký máme editor, ako vieme programovať, aký rýchly máme počítač, aká rýchla je tá aplikácia, či je vhodná na náš obrázok, atď. FORTRANovský kód môžeme zapísať ľubovoľným editorom, ale pokladal som za veľmi dôležité, aby bol trošku prispôsobený k FORTRANu. Hneď ako som dostal prvé PC (1988), naprogramoval som si taký editor a zabezpečil, aby som mohol priamo z neho pustiť preklad, link a samotný beh programu. Svoje programy si píšem priamo z hlavy, takže môžem s ním za 1–3 minúty vyprodukovať aj 100riadkový kód, ktorý na drvivú väčšinu obrázkov stačí.

Druhá vec je vymyslieť ten kód. V tomto nám nepomôže sebelepší editor, najrýchlejší počítač, sebelepšia aplikácia – musíme to jednoducho naučiť a skúšať, aby to druhý krát išlo rýchlejšie. Nadarmo budem schopný za minútu napísať 50 riadkov, keď tých 50 riadkov zfunkčím a odladím za tri dni. Niekedy sa nám

podarí za minútu vyprodukovať 100 obrázkov a niekedy sa potrápime s jedným obrázkom aj 100 minút a viac.

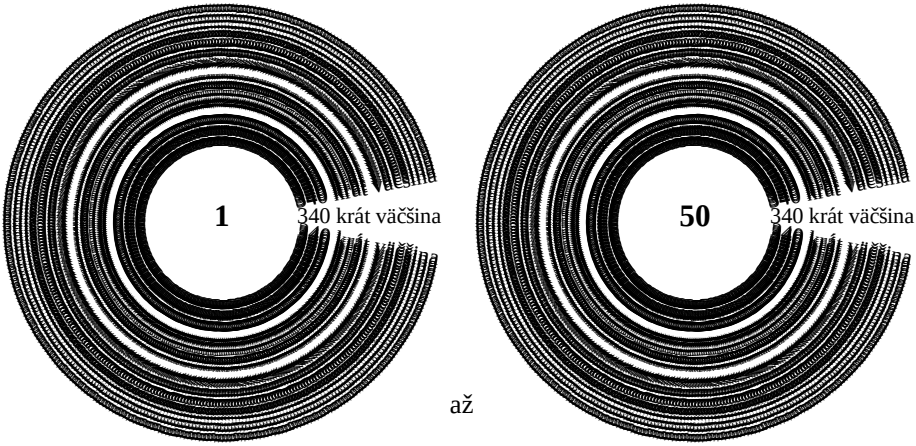
Hoci výroba obrázkov FORTRANom od editora až po GhostView (GV) je dlhá – 1. editor, 2. preklad, 3. link a vytvorenie spustiteľného programu, 4. spustenie a výroba GPVTV súboru, 5. prekódovanie do LaTeXu a PostScriptu, 6. preklad TeXom, 7. DVIPS, 8. GV – zdá sa mi však celkom svižná. Napadlo ma, že skúsim porovnať tento môj postup s rýchlosťou METAPOSTu. Vedel som, že aj v METAPOSTe sa dá niečo zapísať, čo prejde TeXom. Tu je cesta z editora po GV kratšia – 1. editor, 2. preklad METAPOSTom do PostScriptu a do TeXu, 3. preklad TeXom, 4. DVIPS, 5. GV. Krok číslo 2 je podrobne vysvetlený v nižšie spomenutom článku o METAPOSTe.

Malé porovnanie s METAPOSTom

Najviac spomaľujú výrobu obrázkov TeXovské popisky a z nich hlavne rotujúce. Vymyslel som si jednoduchý obrázok, kde bude 340krát vypísaný rotujúci text. S týmto postupom sa vyrobí 50 obrázkov tak (každý na novej strane), že v strede bude vypísané číslo obrázka (tiež LaTeXom). Rotujúci text *(340 krát väčšina)* sa zapísal teda $50 \times 340 = 17000$ krát. Obr. 1 znázorňuje prvý a posledný obrázok, pod ktorým je FORTRANovský zdrojový kód s krátkym komentárom. V editore som vygeneroval dva cykly a do vnútorného som zapísal text s otočením o riadiacu premennú cyklu. Do vonkajšieho cyklu som zapísal text s nulovou rotáciou a do stredu som zapísal riadiacu premennú vonkajšieho cyklu. Behom pár minút som to mal napísané, odladené, vyskúšané.

Nemám nainštalovaný METAPOST a mám s ním nulové skúsenosti. Z TeX konferencie a aj zo ZPRAVODAJa 1–3/2001 (Miroslava Krátká: METAPOST a mfpic) viem, že je to veľmi mocný nástroj. Kým by som sa dopracoval k METAPOSTovým kódom, tak by možno prešiel aj mesiac. Napísal som preto pánovi Kubenovi a potom aj pánovi Klocovi, či by neboli ochotní vyrobiť podobný súbor s pomocou METAPOSTu. Obidvaja sa zachovali ako praví TeXisti a prisľúbili, že mi pošlú riešenia. Nevedel som, že METAPOST nemá nástroje na stránkovanie (nedá sa jednoducho vyrobiť viac obrázkov do jedného súboru – METAPOST vytvára bez problémov aj viac obrázkov, ale vždy ich dáva do samostatných súborov) a tak nechtiac som ich obral o viac času, než som si myslel (za ich námahu a pochopenie im aj touto cestou ďakujem). Pán Kuben mi poslal tri riešenia na tom istom stroji a pán Kloc jedno riešenie na troch rôznych strojoch – čiastočné preto, lebo som mu napísal, aby zbytočne nestrácal čas a v tom nepokračoval. Výsledky som zhrnul do Tab. 1.

Ťažko sa tieto postupy porovnávajú, bol som len zvedavý, či môj postup nezaostáva rádovo od postupu cez METAPOST. Z tej tabuľky dobre vidieť, že počítačový čas za chvíľu nebude hrať žiadnu rolu – procesory nad 2 GHz to



Zdrojový kód	**Komentár**
`cgtpc`	cesta od 1(editor) po 8(GV) zabezpečená
`character a*20,t*200`	definovanie premenných pre text a číslo
`call openg`	otvorenie grafického súboru
`a='\bf'`	font pre číslo
`t='\hspace*{17mm}340 krát väčšina'`	LaTeXový text do premennej `t`
`do 2 j=1,50`	otvorenie cyklu pre obrázky
`call begpic(11,10,3)`	otvorenie obrázka v grafickom súbore
`do 1 k=12,350`	otvorenie cyklu pre rotáciu
`r=k`	do premennej `r` (real) sa vloží `k` (integer)
`1 call larget(0.,0.,'l',t,r)`	výpis textu s rotáciou `r` a koniec cyklu `1`
`call larget(0.,0.,'l',t,0.1)`	výpis textu s rotáciou `0.1`
`call latexi(0.,0.,'c',a,j)`	výpis čísla `j` (integer)
`call move(-7.,4.)`	posunutie obrázka
`2 call eopic`	koniec obrázka a koniec cyklu `2`
`end`	koniec programu

Obr. 1. porovnávacie obrázky

Tab. 1. Porovnávacia tabuľka TEXovej rýchlosti METAPOSTu a FORTRANovského grafického balíka GPVTV.

N	PC MHz/RAM	sys	2-5/2 [s]	TEX [s]	DVIPS [s]	GV [s]	Total [s]	zdroj [bajt]	PS [MB]	ZIP [kB]
LB	590/192	W98/D	4	8	2	6	20	265	4.2	99
LB	1000/256	W00/D	3	5	2	4	14	265	4.2	99
LB	2200/256	WXP/D	2	3	1	2	8	265	4.2	99
JK	2400/512	eCS1.1	2	2	2	2	8	2615	8.2	1117
JK	2400/512	eCS1.1	2	2	2	2	8	545	8.2	1117
JK	2400/512	eCS1.1	2	2	2	2	8	950	8.2	1117
LK	2400/512	eCS1.1/D	4					*500	21.9	2836
LK	900/256	W00	7					*500	21.9	2836
LK	90/64	Linux	55					*500	21.9	2836
LB	133/16	W98/D	17					265	4.2	99
AŠ	590/192	W98/D		2	2	5	9	300	3.2	130
LB	590/192	W98/D	2	4	2	4	12	265	2.5	94

Poznámky k jednotlivým stĺpcom:
N (name) – LB-Ladislav Bittó, JK-Jaromír Kuben, LK-Luboš Kloc, AŠ-Anton Šurda
PC – rýchlosť procesora/operačná pamäť
sys – operačný systém, /D znamená DOS okno (okrem GV všetko DOSové programy)
2–5/2 – kroky 2–5 cez FORTRAN/krok 2 cez METAPOST
TEX, DVIPS, GV, Total nepotrebuje komentár
zdroj – veľkosť zdrojového kódu, * znamená neúplné riešenie
PS – veľkosť výsledného súboru 50 obrázkov
ZIP – veľkosť archivovaného súboru 50 obrázkov.

už zvládnu bleskovo. Pri tých pomalších je situácia iná, tam záleží viac na veľkosti spracovaného súboru, operačnej pamäti, rýchlosti disku, zbernice, systému, od použitých verzií programov atď. Pán Šurda vytvoril čisto plainTEXový kód, ktorý som bez problémov preložil na svojom počítači (predposledný riadok tabuľky). Preklad TEXom v tomto prípade je veľmi rýchly, lebo sa prekladá malý súbor a kvôli zaujímavosti som ho zaradil do tabuľky. Na poslednom riadku uvádzam výsledok môjho snaženia optimalizovať jednotlivé kroky výroby obrázka. V tomto prípade však to nemá príliš veľký význam, lebo obyčajne potrebujeme malé množstvo rotujúceho textu. Vyskúšal som otvorenie všetkých piatych PostScriptových súborov v GV s nasledujúcim výsledkom: 4s, 5s, 6s, 7s, 13s.

Možnosti a použitie

Skúsim v krátkosti naznačiť možnosti a použitie grafického balíka GPVTV obohateného o LaTeXové a PostScriptové príkazy.

1. balík umožňuje zapísať vo FORTRANovskom zdrojáku LaTeXové a PostScriptové príkazy. LaTeXové sú implementované ako FORTRANovské textové a číselné premenné, teda sú aj FORTRANovsky programovateľné, čo môžeme využiť nielen na popisky. Napríklad: pracujem na obrázku, kde by som chcel vykresliť 10 rôznych TeXovských tabuliek do 10tych rôznych bodov a to tak, že ani neopustím FORTRANovský zdroják. Údaje ešte ani nepoznám, budú vyrátané. Dám ich vygenerovať do 10tych rôznych súborov a v obrázku ich vložím do príslušných bodov cez `\input`, alebo vygenerujem do jedného súboru, ako rôzne makrá. Môžem ich dokonca skôr vykresliť, než vygenerovať – na poradí nezáleží, lebo sa TeXuje až po prekódovaní grafického výstupu z GPVTV. PostScriptové príkazy sú generované vo FORTRANovských podprogramoch, ktorých parametre sú FORTRANovsky programovateľné.
2. pohodlná tvorba viacerých obrázkov do jedného súboru (obrázky sú na samostatných stranách)
3. pohodlná tvorba viacerých obrázkov na jednu stranu – umožňuje to grafický systém GPVTV s posuvom súradnej sústavy
4. pohodlná práca s množstvom dátových súborov
5. Vyhodnotenie vedecko-technických výpočtov. Tento pojem je veľmi široký. GPVTV má okolo 150 podprogramov, z toho je väčšina už nepotrebná (rôzne ovládače a iné), ale okolo 50–60 môžem aj teraz využívať. Sú tam podprogramy na 2-D a aj na 3-D grafiku. Ja osobne využívam z GPVTV iba 4–5 výkonných podprogramov a okolo 10, ktoré sú potrebné na definovanie grafického systému a súradnej sústavy.
6. obrázky môžme vytvárať aj priamo vo výpočtových programoch
7. grafické vyhodnotenie rôznych registračných (dátových) záznamov
8. kreslenie diagramov (aj živých - kde čiary idú s objektom), elektrických schém, pohyby planét, planétok, komét po oblohe, hviezdne mapy, ročné priebehy časovej rovnice, atď.
9. balík je súčasťou programovacieho jazyka, teda možno vytvoriť aplikáciu (vykonávateľný súbor s príponou EXE). Takto s ním môže vyrábať obrázok aj ten, kto nepozná ani FORTRAN, ani LaTeX a ani PostScript (s pomocou dátových súborov).
10. presné ručné kreslenie. Trošku som si prispôsobil editor a vytvoril pohyblivý milimetrák, aby som mohol pohodlne kresliť aj také obrázky, ktoré sa nedajú programovať. Teraz už ani nemusím väčšinou nakresliť obrázok na milimetrový papier, ale rovno na obrazovku. Začínam tak, že si vykreslím milimetrák na obrazovku (v jej hornej časti) a postupne kreslím objekty (v dolnej časti obrazovky) vo FORTRANovskom zdrojáku. K úplnej spo-

kojnosti mi chýba už len to, aby GV pracoval aj vtedy keď nie je aktívny – naskytla by sa možnosť vidieť pridávanie príkazov okamžite (čo by som si doprogramoval do editora).

11. umožňuje proste všetko, čo sa dá vo FORTRANe s pomocou GPVTV a s mojimi LaTeXovými a PostScriptovými podprogramami naprogramovať. Môžem napísať a ľahko pridať ďalšie podprogramy, keď to bude potrebné.

Osobne týmto balíkom vytváram 99.5% svojich obrázkov a za 20 rokov som s ním nakreslil (a pomohol nakresliť) viac než dvestotisíc. Niekomu sa môže zdať tento spôsob zložitý, ale snáď z kódu tých porovnávacích obrázkov je zrejmé, že pre užívateľa to nie je až také zložité. Pracujú s ním aj kolegovia, ktorí ináč nepoužívajú TeX (paradoxne je ich viac) a PostScript je pre nich neznáma veličina. FORTRAN bol vymyslený pre vedcov a patrí medzi najjednoduchšie programovacie jazyky (výsledný kód medzi najrýchlejšie). Moje vedomosti o FORTRANe sú slabé, z GPVTV využívam len malú časť, z LaTeXu ovládam minimum a v PostScripte som na samom začiatku – čo mi však nebránilo vo vytváraní podporných programov pre svoj postup. Aj z mála sa niekedy dajú robiť zložité veci – treba len chcieť a to málo ovládať, ostatné príde.

Záverom

Teraz by sa patrilo napísať, kde je ten balík k dispozícii (aj keď sme presvedčení, že oň by nebol veľký záujem). Skúsil som zohnať zdrojové kódy podprogramov spolu s textom manuálu k GPVTV, ale nepodarilo sa mi to (nenašli sa). Ja mám len zlinkovanú DOSovú verziu – čo môžem s povolením VS SAV vystaviť – a jednu príručku o GPVTV (má 170 strán). Berte preto tento článok ako propagáciu TeXu. Keď sa niečo podarí s TeXom vytvoriť, treba sa o tom podeliť. Viem ako ťažké je v súčasnosti k nemu niekoho pritiahnuť. Okolo 1990 to bolo ešte niečo iné. Vytvoril som vtedy špeciálny editor pre TeX a oboznámil som s tým svoje okolie. Bol som rád, že som s tým pritiahol niekoľkých ľudí k práci v tomto sádzacom systéme. Zastavovali ma úplne neznámi ľudia, či by som im nenainštaloval TeX s tým svojím editorom. Sem tam som musel robiť aj školenia. Pár krát sa mi stalo, že som nevedel kam skôr ísť. Ľudia však idú s menším odporom, prišiel WORD a ostatné poznáme.

Som od prírody veľmi lenivý, takže stále vymýšľam, že ako by som si mohol ušetriť prácu. Niekedy sa natrápim a je to k ničomu, ale nesťažujem sa. Naučil som sa, že chyby mám hľadať predovšetkým u seba a keď ich vyriešim vlastnými silami, tak som spokojný. Keď sa chceme pustiť do rôznych projektov, tak najdôležitejší je vždy (a obyčajne aj najťažší) prvý krok s ktorým pustíme na seba tú lavínu problémov. Potom už je len na nás, či to ľahko vzdáme alebo budeme riešiť jeden problém za druhým.

Niekoho môže napadnúť, že načo sa s tým trápim, keď na svete je mnoho programov rôzneho druhu – treba si len vybrať, nainštalovať (niekedy poriadne zaplatiť) a už to aj funguje. Lenže všetko čo je krásne pre človeka, rastie pomaly. Je to trošku aj moje dieťa, takže ho mám rád. Do jeho výchovy mi nikto nezasahuje a teším sa zo všetkého, čo som schopný ho naučiť. Pár rokov ešte potrvá, kým dorastie (ak vôbec). Dôležité je, aby človek mal s čím bojovať, mal čo objavovať pre seba, potom sa môže postarať aj o to najcennejšie čo má – o svoje zdravie.

Poďakovanie

Autor článku ďakuje pánovi Kubenovi a pánovi Klocovi za METAPOSTové riešenie porovnávacích obrázkov, za poskytnutie ich zdrojových kódov a výsledného PostScriptového súboru a za ich pripomienky k článku. Moja vďaka patrí aj pánovi Šurdovi, ktorý prejavil značnú dávku trpezlivosti a ochoty pri opravovaní mojej nie celkom spisovnej slovenčiny (taká ľudová made in Hungary) a hlavne poradil čo pridať/vypustiť, aby článok bol zrozumiteľný pre každého.

Summary: TEX and PostScript in Graphics of Programming Languages

The article describes possibility of generating PostScript graphics by means of a library of subroutines written in FORTRAN. The speed of the program is compared to that of METAPOST. Examples of pictures created by the mentioned program are available from the web page of the Bulletin.

BachoTEX — zpráva o setkání TEXistů v Polsku

Petr Olšák

Polské sdružení uživatelů TEXu GUST pořádá každoročně koncem dubna a začátkem května TEXovou slezinu v polském Bachotku. Jde o rekreační středisko u rybníka na cestě mezi Olsztynem a Brodnicou. Ubytování je ve dřevěných chatičkách se dvěma maličkými patry a šikmou špičatou střechou. V každém

Fotografie: Marcin Sochacki, viz `http://www.gust.org.pl/BachoTeX/2004/zdjecia.html`

patře jsou dvě postele a zřejmě nic více. Do prvního patra se leze po takovém zařízení, které připomíná napůl schody a napůl žebřík. Středisko úzce spolupracuje s universitou v Toruńi. Odehrávají se tam i mimo TEXový termín semináře a konference, které pořádají lidé z Toruńské university. Pokud jsem to správně pochopil, pak tato universita je hlavním stanem GUSTu, takže je přirozené, že se TEXovská setkání odehrávají v Bachotku.

Dřevěný srub v Bachotku. Takových tam je kolem třiceti. Komě toho jsou ve středisku dvě větší přízemní budovy: v jedné je jídelna a v druhé společenská místnost, kde se konaly přednášky

Na BachoTEXu jsem byl naposledy v roce 1998. Dospěl jsem k názoru, že to už bylo docela dávno a že se tam tedy letos podívám zase. Vlakové spojení mezi Prahou a Brodnicí (poslední vesnice před Bachotkem) bez zbytečných oklik

vyžadovalo mnoho přestupů. Vycházelo to na celý den a noc. Podstatně rychleji jsem se na místo dostal přes Ostravu a Warszawu, což ale pohledem na mapu vypadá jako docela velká oklika. Domluvil jsem se, že mě ve Warszawě vyzvedne ráno jeden účastník konference Mariusz Olko a vezme mě do svého auta. Do hlavního města Polska jsem tak mohl jet nočním vlakem.

Asi 14 dní před odjezdem mě kontaktoval e-mailem pan Janusz Nowacki a poslal mi vzorek tzv. LM fontů, na kterých pracuje s ostatními kolegy. Jedná se o projekt „Latin Modern“ fontů, které mají stejnou kresbu jako Computer Modern, ale jsou vybaveny všemi znaky, které používají různé latinkou píšící jazyky, včetně všech písmen s akcenty. Fonty jsou distribuovány v Type1 formátu, do kterého se, jak známo, vejde libovolné množství znaků. Nejsme tedy omezeni číslem 256. K jednomu takovému fontu pak může existovat více TEXových metrik a mapovacích souborů, které z fontu vybírají jen ty znaky, které jsou v daném jazyku potřebné. Principiálně by mohly LM fonty (mimo jiné) nahradit naši současnou Type1 implementaci CSfontů. Samozřejmě při zachování metrik CSfontů beze změny.

Panu Nowackému jsem poslal zpět připomínky k LM fontům. Zejména jsem ho upozornil na skutečnost, že v CSfontech máme různé kresby stejného akcentu pro malé a velké písmeno. Například háček je nad velkým písmenem mělčí než nad písmenem malým. Dostal jsem od něj před konferencí ještě jeden e-mail, ve kterém psal, že provedl průzkum naší Type1 implementace CSfontů a shledal, že při použití distilleru z Acrobatu 5 jsou tyto fonty totálně nefunkční. Akcenty ulítnou do zcela nepatřičných míst. Podivil jsem se, že tuto chybu nehlásil nikdo z českých nebo slovenských uživatelů. Pan Nowacki mi navíc poslal vzorek Type1 formátu jednoho CSfontu, kde chybu opravil za použití programu METATYPE1. Tento program používá jejich skupina nejen na LM fonty, ale i na údržbu dalších fontů, které jsou původem z Polska (viz níže). Zaslaný vzorek CSfontu byl velmi dobrý, byl dokonce úspornější než moje implementace Type1 CSfontů, takže jsem se o jejich program začal zajímat. Také jsem na listu `cstex@felk.cvut.cz` konzultoval otázku, proč CSfonty nefungují s Acrobatem 5. Dozvěděl jsem se, že to je chyba jen této verze a že ve verzi 6 už zase ty fonty fungují. Je tedy sporné, zda to máme považovat za chybu Type1 implementace CSfontů nebo za chybu Acrobatu 5, která se projeví jen při distillování z PostScriptu na PDF. Panu Nowackému jsem poslal výsledky této diskuse a projevil jsem zájem o METATYPE1, o opravu CSfontů i o LM fonty. Dostal jsem od něj zhruba tuto odpověď:

```
Zrobimy tak:
1. Przegenerujemy cs*.pfb, aby byly dobre subrutyny.
2. Poprawimy akcenty w Latin Modern (Czesi uzywaja caron,
   wiedza lepiej jak ma wygladac)
3. Poprawimy metryki lcaron, dcaron i tcaron w Latin Modern
4. Wypijemy polskie piwo w Bachotku i zatwierdzimy poprawki
5. Bedzie pieknie
```

Jistě uznáte, že mě velmi zaujala první část bodu 4, ale i ostatní body stály za pozornost. S tímto „programem konference“ jsem se tedy odebral na BachoTEX 2004.

Oficiální program konference včetně programů minulých ročníků je k dostání na adrese `www.gust.org.pl/BachoTeX/2004/`. Nebudu se tedy ve svém článku k jednotlivým přednáškám vracet. Přiznávám ale, že docela nechápu, jak se daří našim polským sousedům uspořádat každoročně třídenní konferenci věnovanou výhradně TEXu a naplnit ji bohatým programem. Pravda, Poláků je daleko více než Čechů, ale členů GUSTu je zase podstatně méně než členů CSTUGu. Na druhé straně skoro každý člen GUSTu je nějakým způsobem aktivní a měl aspoň na nějakém BachoTEXu příspěvek. Měl jsem pocit, že v Bachotku se sešla většina členů GUSTu a z toho většina z přítomných měla nějaký příspěvek. Na druhé straně se nemohu zbavit pocitu, že CSTUG se skládá většinou z pasívních členů, které akorát zajímá, kdy dostanou časopis nebo CéDéčko.

Další věc, která mě v Bachotku zaujala, byla velmi vysoká účast zahraničních hostů. Letos jich tam bylo kolem dvaceti. Kromě pravidelného účastníka Kees van der Laana tam byli v hojném počtu zastoupeni členové německého DANTE v čele se svým předsedou Volkerem Schaa. Dále tam byl Hans Hagen, kterého jistě znáte jako tvůrce ConTEXtu. Když jsem se díval na fotografie minulých ročníků z BachoTEXu, shledal jsem, že jmenovaní pánové jezdí na polské jarní konference asi docela pravidelně. Letos tam byl také jeden z hlavních autorů Omegy John Plaice.

Se svými příspěvky vystoupili i zahraniční hosté, takže jeden blok přednášek na konferenci byl v angličtině. Jinak byla hlavním rokovacím jazykem polština a přihlížejícím zahraničním hostům vždy někdo z diváků šeptem překládal to podstatné do angličtiny. Mě osobně polština nevadila a shledal jsem, že se jí dá aspoň částečně rozumět, pokud se jedná o prezentaci. Když se ale Poláci baví rychle mezi sebou, je porozumění velmi obtížné. Z České republiky tam kromě mě přijel ještě Tomáš Hála.

Pro zahraniční účastníky letos Poláci připravili zábavné lekce polštiny, které mě osobně připadaly jako známé parodie „alles gute“. Nicméně se všem těmto účastníkům lekce líbily. Navíc byly na závěr rozdány účastnické diplomy. Nejnadějnější student kursu byl zřejmě John Plaice.

Ačkoli jsem uvedl, že nebudu popisovat jednotlivé přednášky, přeci jenom udělám výjimku. Zaujala mě přednáška o historii GUSTu. Dozvěděl jsem se plno zajímavostí o počátcích polského sdružení uživatelů TEXu, o problémech, které na počátku řešili, a lidech, kteří to měli ze začátku na svědomí. Přednášející Andrzej Odyniec téma pojal velmi podrobně, zajímavě a navíc do toho zapojil své nevšední herecké schopnosti a v určitých pasážích se jal recitovat na téma historie GUSTu básně. Když jsem s ním později mluvil a ocenil ho za jeho herecký výkon, řekl, že to nešlo udělat jinak, protože nudné téma, jako je historie, je potřeba něčím okořenit.

BachoTEX se pravidelně koná koncem dubna (polsky kwiecień) a začátkem května (polsky maj). To je období pálení čarodějnic. Proto taky nedílnou součástí programu konference na večer posledního dubna je velký táborový oheň. Letos byl tento oheň obohacen o další událost: o půlnoci vstoupilo Polsko s některými dalšími evropskými státy do Evropské unie. V propozicích konference stálo, že zahraniční účastníci přijedou do Polska a odjedou z Evropské unie. U táborového ohně se shodou okolností sešli tři „presidenti“ lokálních TEXových LUGů, kteří už v Evropské unii jsou (Holandsko, Německo a Dánsko) a tři „presidenti“, jejichž státy do Unie přistupují (Polsko, Česko, Litva). Polští přátelé u této příležitosti udělali o půlnoci u ohně důstojnou oslavu: pustili Beethovenovu Ódu na radost a pozvali přítomné „presidenty“ na přípitek (viz obrázek) a mávali u toho modrými prapory se žlutými hvězdičkami. Jen se mi nepovedlo prokouknout, zda to veselí kolem vstupu do Unie myslejí Poláci upřímně a vážně nebo to berou jako parodii. Bylo to uděláno tak, aby to každý mohl vnímat po svém. Poláci každoročně u táboráku vytáhnou kytary a zpívají dlouho do noci. Letos po přípitku tomu nebylo jinak.

Přípitek presidentů při vstupu do Unie. Zleva: Petr Olšák (cz), Andrzej Borzyszkowski (pl), Vytas Statulevičius (lt), Volker Schaa (de), Hans Hagen (nl), Kaja Christiansen (dk)

Samozřejmě jsem na konferenci vyhledal Janusze Nowackého a probral s ním osobně otázku fontů. Na setkání jsem se náležitě připravil, tj. prostudoval jsem všechny dostupné materiály k programu METATYPE1. Bohužel, ukázalo se, že mi to není moc platné. Moc technických informací jsem se tam nedozvěděl. Byl jsem tedy rád, že byl Janusz ochoten mi program ukázat na svém počítači. „Zdrojové texty“ k fontům jsou uloženy ve formátu METAPOSTu a v komentářích jsou další dodatečné informace. Tahy pera navazují a kopírují přesně obrys písmene, takže konverze do Type1 formátu nenarazí na vážnější problémy. Zdrojové texty se proženou programem METATYPE1, tj. přesněji METAPOSTem, awkem a t1utils. Tím dostaneme TEXovou metriku a Type1 font. Nebo to udělá PostScriptovou dokumentaci k fontu, jako dělával kdysi Knuth přímo METAFONTem s následným `gftodvi`. Pokud máme font k dispozici jen v Type1 formátu, pak můžeme použít nástroje, kterými font převedeme do METAPOSTových „zdrojových souborů“.

Takovým způsobem udělal Janusz opravu jednoho vzorku z CSfontů. Převedl to do METAPOSTu a následně pomocí programu METATYPE1 zpět do Type1. Ukázalo se ale, že moje Type1 implementace CSfontů, která vychází z Bakoma fontů, má na některých místech drobné chyby (nepatrné překrývnání tahů atd.). Program METATYPE1 chyby vychytá, ale neumí je automaticky opravit. Ono to někdy z charakteru té chyby ani moc nejde. Janusz mi ukazoval, jak by věci byl schopen opravit manuálně, ale ukázalo se, že by to byla při tom množství CSfontů šílená práce. Musel se totiž po odhalení chyby znovu vrátit do editoru, najít číslo kóty, které se chyba týká, cosi poeditovat v METAPOSTovém zdrojáku a spustit proces konverze znovu. Společně jsme to tedy zavrhli a shledali, že oprava Type1 implementace CSfontů se nevyplatí a že tedy uživatelé Acrobatu 5, kteří jím distillují dokumenty s CSfonty, budou muset povýšit svůj Acrobat na verzi aspoň 6.

Když mi Janusz svůj program METATYPE1 ukazoval, zdálo se mi to poměrně složité. Proto jsem se zeptal, kde je k tomu technická dokumentace. Odpověděl, že dokumentace je, bohužel, ve vývoji. Program METATYPE1 je šířen zdarma a je k dostání například na `ftp://bop.eps.gda.pl/pub/metatype1/`, ale bez aktuální dokumentace. On a jeho kolegové, kteří program vyvíjejí, jsou zároveň jeho uživatelé. Dělají tím Antykwu Toruńskou, Antykwu Poltawskiego a již zmíněné LM fonty. Zdroje k těmto fontům mají v METAPOSTovém formátu vhodném pro zpracování programem METATYPE1, ale veřejně jsou šířeny jen výstupy tohoto programu ve formě TEXových metrik a Type1 fontů. Když jsem se ptal na důvod, bylo mi odpovězeno, že zdroje zatím nejsou tak pročištěny, aby mohly být zveřejněny, a přitom se za ně autoři nemuseli stydět.

Pokud jde o LM fonty, dohodli jsme se na tom, že v budoucnu mohou nahradit Type1 implementaci CSfontů. Metriky CSfontů samozřejmě zůstanou beze změny, ale mapovací soubory se budou odkazovat na LM fonty. Janusz ochotně opravil akcenty podle mých námětů (dvojí druh akcentů pro malá a velká

písmena) a je nakloněn další spolupráci. Jsme si vědomi, že pro možnost použít stávající metriky CSfontů proti novým LM fontům je bezpodmínečně nutné, aby šířky všech znaků byly zcela stejné. Protože jsou šířky v obou případech odvozeny z CM fontů, neměl by to snad být takový problém. Tuto otázku je ještě potřeba podrobněji prozkoumat. Jiné metrické údaje (výšky, kerning) nejsou v `pfb` formátu zapsány, takže zde konflikt s metrikami CSfontů neočekáváme.

Byl jsem i na valném shromáždění GUSTu, které proběhlo jeden večer během konference. Mezi členy GUSTu patří i někteří zahraniční účastníci. Já osobně členem nejsem, jen jsem to tam tak trochu po očku sledoval. Nejprve odříkali nějaké zprávy o činnosti a o finanční situaci. To bylo podobné, jako u nás. Pak začaly volby do nového výboru. Zaujalo mě, že lidé se vzájemně navrhovali jako kandidáti do výboru na místě a že se jim podařilo během chvíle mezi přítomnými účastníky najít dostatek kandidátů, aby volby byly skutečně volbami, tj. aby kandidátů bylo více než je potřebných členů výboru. Dále proběhly volby předsedy. Původní předseda Andrzej Borzyszkowski ukončil zrovna funkční období a bylo potřeba nají předsedu nového. I to se bez problému na místě podařilo a dokonce se volilo mezi dvěma kandidáty. Novým předsedou se stal Jerzy Ludwichowski. Osobně si myslím, že v tomto ohledu se jeví GUST daleko životaschopnější než ČSTUG. I my budeme letos volit výbor a ten nového předsedu. Jsem velmi zvědav, zda se nám podaří také tak lehce získat dostatek kandidátů.

Ke konci konference se v Bachotku sešla silná skupina uživatelů TEXlive s některými tvůrci a přispěvateli a diskutovali další směřování této TEXové distribuce. Já jsem tam už nebyl, neboť v té době jsem byl už na cestě do Prahy. Výsledky z tohoto jednání byly pak zveřejněny na `tex-live@tug.org`. Kdo čte tento diskusní list, ten ví, že Sebastian Rahtz se poněkud naštval, že se jednalo o něm bez něj. Vyhrožoval, že za dané situace přestane TEXlive dělat. Horké hlavy musel nakonec urovnávat Karl Berry a naštěstí se mu to podařilo.

V závěru tohoto článku bych chtěl ještě jednou vyjádřit obdiv nad životaschopností GUSTu, nad tím, že jsou schopni každoročně pořádat konference o TEXu s bohatým programem, že jim tam jezdí zahraniční účastníci (a to ne jen tak ledajací) a že mají mezi svými členy výrazně vyšší procento aktivních lidí než v našem ČSTUGu. Přiznávám, že jim tak trochu závidím. Nedalo by se s tím v ČSTUGu taky něco udělat? Když jsme za měsíc poté pořádali s ČSTUGem a CZLUGem konferenci SLT, přijelo tam kvůli TEXové části asi dvanáct lidí. Poláků bylo v Bachotku kolem stovky. V čem je rozdíl?

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661 (tištěná verze), ISSN 1213-8185 (online verze)

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Antonín Strejc
Počet výtisků:	650
Uzávěrka:	7. července 2004
Odpovědný redaktor:	Zdeněk Wagner
Redakční rada:	Petr Aubrecht, Matěj Cepl, Jiří Demel, Jana Chlebíková, Jiří Kosek, Jaromír Kuben, Petr Sojka, Martin Tkadlčík
Tisk:	WOW, s. r. o., Praha 1, Washingtonova 25
Adresa:	CSTUG, c/o FEL ČVUT, Technická 2, 166 27 Praha 6
Tel:	+420 224 353 611
Fax:	+420 233 332 938
Email:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
korespondence předsedovi sdružení

`gacstug@cstug.cz`
grantová agentura CSTUGu

`secretary@cstug.cz`, `orders@cstug.cz`
korespondence administrativní síle sdružení, objednávky CD-ROM

`cstug-members@cstug.cz`
korespondence členům sdružení

`cstug-faq@cstug.cz`
řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

ftp server sdružení:
`ftp://ftp.cstug.cz/`

www server sdružení:
`http://www.cstug.cz/`

Uzávěrka příštího čísla: 2. září 2004